

Gnus Manual

by Lars Magne Ingebrigtsen

Copyright © 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007
Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with the Front-Cover texts being “A GNU Manual”, and with the Back-Cover Texts as in (a) below. A copy of the license is included in the section entitled “GNU Free Documentation License” in the Emacs manual.

この文書を、フリーソフトウェア財団発行の GNU フリー文書利用許諾契約書第 1.2 版またはそれ以降の版が定める条件の下で複製、配布、あるいは変更することを許可します。変更不可部分は指定しません。“A GNU Manual”は表表紙テキスト、以下の (a) は裏表紙テキストです。この利用許諾契約書の複写は“Emacs manual”の「GNU フリー文書利用許諾契約書」という章に含まれています。

(a) The FSF’s Back-Cover Text is: “You have freedom to copy and modify this GNU Manual, like GNU software. Copies published by the Free Software Foundation raise funds for GNU development.”

(a) FSF の裏表紙テキスト:「あなたにはこの GNU Manual を GNU ソフトウェアのように複製したり変更する自由があります。複製はフリーソフトウェア財団によって出版されました。(フリーソフトウェア財団は) GNU の開発のために必要な資金を集めています。」

This document is part of a collection distributed under the GNU Free Documentation License. If you want to distribute this document separately from the collection, you can do so by adding a copy of the license to the document, as described in section 6 of the license.

この文書は「GNU フリー文書利用許諾契約書」に基づいて配布された収集著作物の一部です。もしあなたがこの文書を収集著作物から分離して配布したいときは、契約書の第 6 章に記述されているように、文書に契約書の複写を付加することによって、行なうことができます。

The Gnus Newsreader

Gnus は先進的で、説明不用で、カスタマイズ可能で、拡張可能な、リアルタイムでない GNU Emacs のニュースリーダーです。

おおっと。不思議なことに以前にも似たようなことを聞いたことがあるような気がします。真似をしたと非難されないうちに説明を始めましょう:

Gnus はメッセージを読むことに関する実験場です。Gnus はすべてをニュースグループのように表示します。Gnus でメールを読み、ディレクトリーをブラウズし、ftp をすることができます。ああ、それに、ニュースを読むことさえできます!

Emacs が文章を編集する人に力を与えるように、Gnus はニュースを読む人に力を与えようとしています。Gnus は利用者が実行可能なことに制限を設けません。利用者が自分で望む動作をするように Gnus を拡張することを奨励しています。プログラムが人を操作するべきではありません。人がプログラムを使う (もしくは濫用する) ことによって、やりたいことをできるようになっているべきなのです。

1 Gnus の起動

Gnus を使う以前にあまり Emacs を使っていないのならば、最初に Section 10.9 [Emacs for Heathens], page 355 を読んで下さい。

システム管理者が適切な設定をしていたならば、Gnus を起動してニュースを読むのは非常に簡単です。そう、Emacs で *M-x gnus* と打つだけです。さもなければ、変数 `gnus-select-method` をカスタマイズしなければなりません。これは Section 1.1 [Finding the News], page 3 で説明されています。また、投稿するための最低限の設定を行なうために、変数 `user-full-name` および `user-mail-address` もカスタマイズしなければなりません。

別のフレーム (frame) で Gnus を起動したいときは、*M-x gnus-other-frame* 命令を使うことができます。

開始時に何かがうまくいかないときは '`~/gnus.el`' ファイルの中で変数をいくつかいじくりまわさなければなりません。このファイルは '`~/emacs`' と似ていますが、こちらは Gnus が起動するときに読み込まれます。

この説明書でよくわからない用語がでてきたときは、用語の章 (Section 10.5 [Terminology], page 328) を参照して下さい。

1.1 ニュースを見つける

変数 `gnus-select-method` は Gnus がどこでニュースを探すべきかを示します。この変数ははじめの要素が「方法」、二番目の要素が「場所」を表すリストである必要があります。この方法はあなたの基本方法 (native method) になります。この方法で取ってこないグループはすべて外部 (foreign) グループです。

例えば NNTP サーバー '`news.somewhere.edu`' から毎日 (薬のように) 一定の量のニュースを摂取したいのであれば、

```
(setq gnus-select-method '(nntp "news.somewhere.edu"))
```

のようにすることができます。

ローカル・スプールのディレクトリーを読み込みたい場合は、

```
(setq gnus-select-method '(nnspool ""))
```

のようにできます。

ローカルのスプールを使えるのであれば、かなりの確率でその方がずっと速いでしょうし、それを使うべきでしょう。でも、もしあなたのサーバーが Leafnode (それは簡単な個人用のニュースサーバーです) であるならばローカルスプールを使ってはいけません。この場合は `(nntp "localhost")` にしましょう。

もしこの変数が設定されていなければ、Gnus は `NNTPSERVER` 環境変数を読みに行きます。もしその変数が設定されていなければ、Gnus は `gnus-nntpserver-file` (設定されていない場合は '`/etc/nntpserver`') がこの件に関して何かを言っていないかを調べます。もしそれも失敗したなら、Gnus は Emacs が動作しているサーバーを NNTP サーバーとして使おうとします。随分な当て推量ですけどね。

`gnus-nntp-server` が設定されていると、この変数は `gnus-select-method` よりも優先されます。ですから `gnus-nntp-server` は `nil` に設定するべきで、それがデフォルトです。

Gnus に NNTP サーバーの名前を対話的に指定することもできます。gnus に数値でない接頭引数を渡すと (例: *C-u M-x gnus*)、Gnus は `gnus-secondary-servers` リスト (もし存在するならば) からサーバーを選ぶことができるようになります。ただ単に接続したいと思ったサーバーの名前を

打つこともできます。(これは `gnus-nntp-server` を設定し、これは後の Emacs のセッションで `M-x gnus` とすると、Gnus は同じサーバーに接続しようとするということです。)

しかし、普段日常的には一つの NNTP サーバーを使い、違ったサーバーには興味のあるグループが少ししかない場合、グループバッファで `B` 命令を使うことの方が良いでしょう。それは、選択可能なグループを表示し、その中からどれでも好きなものを購読することができます。これは `‘.newsrsc’` の保持をずっとやりやすくします。See Section 2.9 [Foreign Groups], page 21.

外部グループに対する少し違ったやり方は、変数 `gnus-secondary-select-methods` を設定する方法です。この変数に表されている選択方法は、多くの点で `gnus-select-method` サーバーの選択方法と同じように扱われます。起動中にアクティブファイルを探しにいき (もし要求されていれば)、これらのサーバー上にできた新しいニュースグループは元々のグループと同じように購読されません (もしくは、されません)。

例えばメールを読むために `nnmbox` バックエンド (back end) を使いたいときは、普通この変数を、

```
(setq gnus-secondary-select-methods '((nnmbox "")))
```

と設定します。

注: NNTP バックエンドは印ファイル (see Section 6.2.1.4 [NNTP marks], page 144) に印を保存します。この機能は Gnus がインストールされている複数のホスト間で印を共有することを容易にしますが、新着記事の取得をちょっと遅くするかもしれません。詳細については Section 6.2.1.4 [NNTP marks], page 144 を参照して下さい。

1.2 一番初め

起動用ファイルが存在しないときは (see Section 1.7 [Startup Files], page 8)、Gnus はどのグループがデフォルトで購読されているべきかを決定しようとします。

変数 `gnus-default-subscribed-newsgroups` が設定されていると、Gnus はそのリストの中のグループを購読し、残りを削除します。システム管理者はこの変数を何か役に立つものに設定しておくことが望まれます。

そうでないときは、Gnus は少数の適当なグループを購読します (例: `‘*.newusers’`)。 (「適当な」はここでは「あなたが読むべきであると Lars さんが考えるもの」というように定義されています)

また、たいいていの共通の問題の解決の手助けになるよう、Gnus に関する文書のグループも購読することになるでしょう。

`gnus-default-subscribed-newsgroups` が `t` のときは、Gnus は新しいグループを扱うのに普通の関数を使い、特別なことは何もしません。

1.3 サーバーが落ちている

デフォルトのサーバーが落ちているときは、当然 Gnus の起動にいくつかの問題が発生します。しかし、ニュースグループの他にいくつかメールのグループがあるのならば、それにもかかわらず Gnus を起動する必要があるかもしれません。

信頼できるプログラムの一つである Gnus は、サーバーと接続できないときは基本選択方法なしで続けるかどうかを尋ねます。これは実際にはサーバーが存在しないとき (例えば、アドレスを間違えた場合) やサーバーが何らかの理由で一時的に調子がおかしくなっているときに起こります。もしそのまま続行することにして、外部グループが一つも無い場合、実はグループバッファではほとんど何もできないということに気が付くでしょう。でも、ねえ、それはあなたの問題です。プブーッ!

サーバーが完全に落ちているのを知っているか、サーバーでわずらうことなくメールだけを読みたいときは、Gnus を起動するのに `gnus-no-server` 命令を使うことができます。急いでいるときにもぴったりでしょう。この命令は本来のサーバーには接続しません—その代わりに、レベル 1 と 2 にあるすべてのグループを活動状態にします（基本グループでないグループはその二つのレベルにしておくのが望ましいでしょう）。Section 2.6 [Group Levels], page 19 も参照して下さい（訳注: `gnus-no-server` は `gnus-group-use-permanent-levels` 変数の値を 2 に設定することに注意して下さい）。

1.4 Gnus をスレーブにする

あなたには二つ以上の Gnus をそれぞれ別の Emacs 上で同時に動かす必要が生じるかもしれませんが。違った `.newsrsrc` ファイルを使っているなら（例えば、二つの違ったサーバーから読み込むために、二つの違った Gnus を動作させている場合）、まったく問題はありません。それを行えば良いだけです。

問題は、同じ `.newsrsrc` ファイルを使う二つの Gnus を動かそうとしたときに起こります。

この問題に対処するために Gnus タワーのシンクタンクにいる私たちは、新しい概念にたどりつきました。「マスター」と「スレーブ」です。（私たちはこの概念に特許を申請しました。そして、その言葉の著作権を得ました。お互いに関連してこれらの言葉を使いたいなら、一回使う毎に、私に \$1 を送らなければなりません。もちろん「コンピュータアプリケーションのマスター/スレーブ関係」の使用料はもっと高くなります。）

とにかく、`M-x gnus`（もしくは、普段やっている方法）で Gnus を普通に起動します。その後のスレーブ Gnus はそれぞれ `M-x gnus-slave` で起動します。スレーブは普通の `.newsrsrc` は保存しませんが、代わりに「スレーブファイル」にスレーブの起動中にどのようなグループが読まれたかという情報だけを保存します。マスター Gnus が起動するとき、それはそれらのスレーブファイルを読み込み（そして消し）、それらからすべての情報を取り込みます。（スレーブファイルは、最終的な変更が優先されるようにそれらが作られた順番で読まれます。）

もちろん、スレーブファイルからの情報は普通の（すなわち、マスターの）`.newsrsrc` ファイルよりも優先されます。

スレーブを起動するときにもしマスターの `.newsrsrc*` ファイル群がセーブされていなかったら、自動保存されたファイルを読むかどうかを尋ねられるかもしれません。“yes”と答えると、マスターにセーブされていない変更はスレーブに反映されません。“no”と答えると、マスターで読まれたいくつかの記事が、スレーブでは未読であると見なされるかもしれません。

1.5 新しいグループ

新しいニュースグループをまったく見なくても満足ならば、`gnus-check-new-newsgroups` を `nil` に設定することができます。これを設定した場合、起動にかかる時間が短くなります。この変数が `nil` に設定されていても、グループバッファで `U` を押せばいつでも新しいグループを購読することができます（see Section 2.13 [Group Maintenance], page 32）。デフォルトではこの変数は `ask-server` です。この変数が `always` に設定されていると、`g` 命令を実行したときでも Gnus はバックエンドに新しいグループを探すことを求めます（see Section 2.17.1 [Scanning New Messages], page 40）。

1.5.1 新しいグループを調べる

Gnus は、普通はグループが新しいかどうかを、購読しているグループと削除されているグループのリストとアクティブファイルを比較することにより判定しています。この方法は特に速いというわ

けではありません。gnus-check-new-newsgroups が ask-server であると、Gnus はサーバーに、最後に接続してから新しいグループができていかどうかを尋ねます。この方法は速いし、安上がりです。これにより、削除されたグループのリストを保持しておくことから完全に開放されます。ですから、gnus-save-killed-list を nil にすることができるでしょう。そうすれば、起動、終了の両方、そして全体にわたって時間を節約できます。ディスク消費量も少なくなります。それなら、どうしてこれがデフォルトではないのでしょうか？ 残念ながら、すべてのサーバーがこの命令を理解するわけではないのです。

私は今あなたが何を考えているかを当てられます。どうすればサーバーが ask-server を理解するかがわかるのでしょうか？ え、違うのですか？ ああ、良かった。というのは、確実な答は存在しないのです。私に言えることは、この変数を ask-server に設定して、数日間新しいグループが現れるかどうかを調べて下さい、ということだけです。もしいくつかのグループが現れたなら、それで動作しています。一つも現れなければ、それは動作していません。私は、Gnus にサーバーが ask-server を理解するかどうかを推量させる関数を書くこともできますが、それは単に推量しているにすぎません。ですから、その関数を書くことはしないでしょう。他の方法としては、サーバーに telnet をして、HELP と打ち、サーバーが理解するコマンドの中に 'NEWGROUPS' があるかどうかを調べることもできます。もしあれば、おそらく動作するでしょう（しかし、適切に機能を提供することなく 'NEWGROUPS' をリストに含めるサーバーもあります）。

この変数は、選択方法のリストであることもできます。そのときは、Gnus は ask-server 命令をそれぞれの選択方法に対して実行し、普通の方法で購読します（もしくは、しません）。この副作用は、起動にかなり時間がかかるので、待っている間に瞑想できることです。永久の幸福を達成するために、マントラ“ dingnusdingnusdingnus ”を使って下さい。

1.5.2 購読方法

新しいグループに遭遇したときに Gnus が何をするかは、変数 gnus-subscribe-newsgroup-method によって決定されます。

この変数は関数を含んでいる必要があります。この関数は新しいグループの名前を唯一の引数として呼ばれます。

いくつかの手軽なブレハブ関数は、以下のようになっています。

gnus-subscribe-zombies

すべての新しいグループをゾンビ (zombie) にします。これがデフォルトになっています。後でゾンビを (A z によって) 概観したり、(S z によって) 適切にすべてを削除したり、(u によって) 購読したりできます。

gnus-subscribe-randomly

任意の順番ですべての新しいグループを購読します。実際には、すべての新しいグループはグループバッファの『一番上』に加えられます。

gnus-subscribe-alphabetically

すべての新しいグループをアルファベット順に購読します。

gnus-subscribe-hierarchically

すべての新しいグループを階層的に購読します。この関数と gnus-subscribe-alphabetically の違いは少ししかありません。gnus-subscribe-alphabetically は新しいグループを厳密にアルファベット順にならべますが、この関数はグループをその階層の中に入れます。ですから、'rec' の階層を 'comp' の階層の前に持てきたい場合、この関数はその配置をぐちゃぐちゃにはしません。もしくは、そのようなものです。

gnus-subscribe-interactively

新しいグループを対話的に購読します。これは Gnus が すべてのグループに対して尋ねることを意味しています。購読するグループは階層的に購読されます。

gnus-subscribe-killed

すべての新しいグループを削除します。

gnus-subscribe-topics

グループを、それに合致する subscribe トピックパラメーターを持っているグループに入れます (see Section 2.16.5 [Topic Parameters], page 37)。例えば、以下のような subscribe パラメーター

```
"nnslashdot"
```

は、その正規表現に合致するすべてのグループはそのトピックの下で購読されるということです。

グループに合致するトピックが無い場合、グループは最上位のトピックで購読されます。

上の変数と密接に関係する変数は、gnus-subscribe-hierarchical-interactive です。この変数が nil でないと、Gnus は階層的な方法で新しいグループを購読するかどうかを尋ねます。Gnus はそれぞれの階層で、それを下に降りるかどうかを尋ねます。

よくある間違いは、数段落前の (gnus-subscribe-newsgroup-method) 変数を gnus-subscribe-herarchical-interactive に設定することです。これは誤りです。これは動作しません。これはおめでたい人のすることです。ですから、絶対にしないで下さい。

1.5.3 新しいグループを選別する

どの新しいグループが購読 (もしくは、無視) されるべきかを管理する快適で手軽な方法は、`‘.newsrc’` ファイルの先頭に `options` 行を挿入することです。次は、例です。

```
options -n !alt.all !rec.all sci.all
```

この行は、明らかにまじめで理知的で科学的な人間 (あるいは彼女はどこにでもいる単につまらない人かもしれないけれど) が書いたものです。なぜなら、これは `‘alt’` と `‘rec’` で始まる名前を持つグループはすべて無視され、`‘sci’` で始まる名前を持つグループはすべて購読する、ということを表しているからです。Gnus はこれらのグループを購読するのに普通の購読方法を使いません。代わりに `gnus-subscribe-options-newsgroup-method` が使われます。この変数はデフォルトで `gnus-subscribe-alphabetically` になります。

`‘.newsrc’` ファイルをいじりたくない場合は、`gnus-options-subscribe` と `gnus-options-not-subscribe` の二つの変数だけを設定することもできます。この二つの変数は `‘.newsrc’` ファイルの `‘optinos -n’` 行とまったく同じことをします。どちらの変数も正規表現で、新しいグループは前者に合致すれば無条件に購読され、後者に合致すると無視されます。

さらにここでおせっかいをする変数は、`gnus-auto-subscribed-groups` です。それは `gnus-options-subscribe` とまったく同じように動作するので、本当は余分なものです。しかし、私はこの二つがあった方が良かったと思いました。もう一方の変数は利用者がいじくするのに使われるのに対して、この変数はいくつかの基本的な規則を設定するためのものです。デフォルトではこの変数はメールバックエンド (`nnml`, `nnbabyl`, `nnfolder`, `nnmbox`, `nnmh` および `nnmaildir`) からできるすべての新しいグループを購読するようになっています。それが嫌であれば、この変数を `nil` に設定して下さい。

この正規表現に合致する新しいグループは `gnus-subscribe-options-newsgroup-method` を使って購読されます。

1.6 サーバーを換える

ときどき、ある NNTP サーバーから別のサーバーへ移動しなければならないことがあります。このようなことはめったにおきませんが、おそらくあなたが仕事を変えたり、使っているサーバーがとても不安定で、別のものに移り換えたいというときに必要になるでしょう。

サーバーを変更するのはとても簡単ですね？ `gnus-select-method` を新しいサーバーを指し示すように変更すればいいだけですね？

違います！

記事の番号は違った NNTP サーバーでも (どうにかして) 同じにしてあるということはありません。そして、Gnus がどの記事を読んだかを記録する唯一の方法は、記事番号を記録することです。ですから `gnus-select-method` を変更したときは、`‘.newsrc’` ファイルは役に立たなくなります。

Gnus は `‘.newsrc’` ファイルのあるサーバー用から別のサーバー用に変換する関数を二、三用意しています。それらには一つ共通点があります—実行になが—い時間がかかることです。おそらく、どうしても必要になったとき以外にこの関数を使おうとは思わないでしょう。

もし両方のサーバーに接続できるなら、Gnus はあなたが読んだ記事すべてに対してヘッダー (headers) を要求して、Message-ID を比較し、読んだ記事と記事の印を新しく記録します。M-x `gnus-change-server` コマンドはこれをすべての基本グループに対して行ないます。そのコマンドは移動先の方法 (the method) を入力することを要求します。

個々のグループを M-x `gnus-group-move-group-to-server` 命令で移動することもできます。これはあるサーバーから別のサーバーへ一つの (外部) グループを移動したいときに役に立ちます。

古いサーバーと新しいサーバーの両方に接続することができないとき、印と読んだ範囲はすべて意味が無くなります。そのようなときは M-x `gnus-group-clear-data-on-native-groups` コマンドを使って、基本グループに関するデータをすべて消去することができます。このコマンドは注意して使って下さい。

`gnus-group-clear-data` コマンドは現在のグループのすべてのデータをクリアします—印と既読記事のリストを消し去ります。

サーバーを変更した後で、キャッシュ階層を移動させなければなりません。というのは、キャッシュ記事は間違った記事番号になっており、それは Gnus がどの記事を読んだとみなすかに影響します。`gnus-group-clear-data-on-native-groups` はそれを自動で行なってしまうかどうかを尋ねます。`gnus-group-clear-data` では M-x `gnus-cache-move-cache` が使えます (でも気を付けて、それはすべてのグループのキャッシュを移動してしまいますから)。

1.7 起動ファイル

最もありふれた Unix のニュースリーダーは、`‘.newsrc’` と呼ばれる共用の起動ファイルを使います。このファイルは、講読しているグループと、それらのグループにおいてどの記事が読まれたかの、すべての情報を持っています。

GNUS ではものが少々複雑になっています。`‘.newsrc’` ファイルを最新のものにするだけでなく、`‘.newsrc’` ファイルには合わない情報を保存しておくために `‘.newsrc.el’` と呼ばれるファイルを使います。(実際は `‘.newsrc’` ファイルのすべての情報を複製して保持しています。) GNUS はこれらの中で一番最後に保存されたものを使います。これをするにより、GNUS と他のニュースリーダーを切り替えて使うことができます。

これはちょっと間が抜けているので、Gnus はもっと良い方法を編み出しました。`‘.newsrc’` と `‘.newsrc.el’` ファイルに加えて、Gnus は `‘.newsrc.eld’` と呼ばれるファイルも持っています。Gnus はこれらの中で一番新しいファイルを読みますが、`‘.newsrc.el’` ファイルに書き込むことは

ありません。‘.newsrsrc.eld’ ファイルは絶対に消すべきではありません。—それは ‘.newsrsrc’ ファイルにはないたくさんの情報を保持しています。

gnus-save-newsrsrc-file を nil にすることによって ‘.newsrsrc’ ファイルに書き込むのを止めることができます。そうすれば、そのファイルを削除することができ、ディスク容量を節約することができ、Gnus の終了が速くなります。しかし、そうすると他のニュースリーダーを使えなくなります。でも、ちょっと、誰かそうしたい人がいるでしょうか。同じように gnus-read-newsrsrc-file を nil にすることによって、Gnus は ‘.newsrsrc’ ファイルとすべての ‘.newsrsrc-SERVER’ ファイルを無視するようになります。そのことは、あなたが時々違うニュースリーダーを使ったり、利用可能なグループの異なるサブセットをそれらのニュースリーダーで読みたい場合に、便利があります。

gnus-save-killed-list (デフォルトは t) が nil であると、Gnus は削除されたグループを起動ファイルに保存しません。これは (起動時と終了時の) 時間と、(ディスクの) 容量を節約します。こうすると Gnus がどのグループが新しいかの記録を持っていないことになるので、新しいグループの自動購読方法は意味がなくなります。この変数を nil にしたときは、gnus-check-new-newsgroups を常に nil か ask-server にしておくべきでしょう (see Section 1.5 [New Groups], page 5)。この変数は正規表現であることもできます。そのような場合は、ファイルを保存する直前にその正規表現に合致しないすべてのグループを消去します。これは、すべてのサーバーが ask-server を理解するわけではない、といったような、いくらかあいまいな状況のときに役に立つでしょう。

変数 gnus-startup-file は起動ファイルがどこにあるかを指定します。デフォルト値は ‘~/newsrsrc’ で、それがどのようなものであれ、末尾に ‘.eld’ を付けたものが Gnus (El Dingo) の起動ファイルになります。このファイルのバージョン制御をしたいときは gnus-backup-startup-file をセットして下さい。それは version-control 変数と同じ値を取ります。

gnus-save-newsrsrc-hook は各種の newsrsrc ファイルのどれかを保存する前に実行されるのに対し、gnus-save-quick-newsrsrc-hook は ‘.newsrsrc.eld’ ファイルを保存する前に実行され、gnus-save-standard-newsrsrc-hook は ‘.newsrsrc’ ファイルを保存する前に実行されます。後の二つは普通はバージョン制御を on/off するのに使われます。デフォルトでは、起動ファイルを保存するときにバージョン制御が行なわれます。バックアップファイルの作成を止めたいときは、次のようにして下さい。

```
(defun turn-off-backup ()
  (set (make-local-variable 'backup-inhibited) t))

(add-hook 'gnus-save-quick-newsrsrc-hook 'turn-off-backup)
(add-hook 'gnus-save-standard-newsrsrc-hook 'turn-off-backup)
```

Gnus が起動すると、gnus-site-init-file (デフォルトで ‘.../site-lisp/gnus-init’) と gnus-init-file (デフォルトで ‘~/gnus’) のファイルを読み込みます。これらは普通の Emacs Lisp ファイルで、‘~/emacs’ や ‘site-init’ ファイルを Gnus 関係のもので乱雑にしないようにするために使うことができます。Gnus はこれらと同じ名前のファイルに、接尾語 ‘.elc’ と ‘.el’ が付いているものも調べます。言い換えれば、gnus-init-file を ‘~/gnus’ に設定すると、Gnus は ‘~/gnus.elc’, ‘~/gnus.el’ を探し、最後に ‘~/gnus’ を (この順番に) 探します。‘-q’ または ‘--no-init-file’ オプション (see section “Initial Options” in *The Emacs Editor*) が指定されて Emacs が起動された場合、Gnus は gnus-init-file を読み込みません。

1.8 自動保存

何か Gnus のデータを変更すること (記事を読む、印を付ける、グループを削除または購読する) をしたとき、変更は特別な「ドリブルバッファ」(dribble buffer) に書き込まれます。このバッファは Emacs が普通するように自動保存されます。‘.newsrc’ ファイルを保存する前に Emacs が落ちたときは、すべての変更をこのファイルから回復することができるでしょう。

起動時に Gnus がこのファイルの存在を発見すると、Gnus はそれを読み込むかどうかを利用者に尋ねます。本当の起動ファイルが保存されれば、自動保存ファイルは削除されます。

gnus-use-dribble-file が nil であると、Gnus はドリブルバッファを作ったり、維持したりしません。デフォルトは t です。

Gnus はドリブルファイルを gnus-dribble-directory に置きます。デフォルトではそのようになっていますが、この変数が nil であると、Gnus は ‘.newsrc’ ファイルの置かれているディレクトリー (これは普通は利用者のホームディレクトリーです) に入っているドリブルファイルを作ります。ドリブルファイルは ‘.newsrc’ と同じ許可属性を与えられます。

もし gnus-always-read-dribble-file が nil でなければ、Gnus は利用者に尋ねること無く、ドリブルファイルを起動時に読み込みます。

1.9 アクティブファイル

Gnus は起動したときや、実際に新しい記事が到着しているかを判定しようとするときに、アクティブファイルを読み込みます。これはとても大きなファイルで、そのサーバーの活動中のグループと記事のすべてのリストが入っています。

アクティブファイルを検査する前に、Gnus は正規表現 gnus-ignored-newsgroups に合うすべての行を削除します。これは主に偽の名前を持つグループを排除するために使われてきましたが、興味の無いグループの階層を無視するために使うこともできます。しかし、これはお勧めできません。本当のことを言うと、まったく賛成できません。代わりに、そのような用途に用いられる変数の概略を知るために、Section 1.5 [New Groups], page 5 を参照して下さい。

アクティブファイルは比較的大きくなる傾向があるので、遅い回線を使っているときは、アクティブファイルを読み込まないように gnus-read-active-file を nil に設定することができます。この変数はデフォルトでは some です。

そのような時は、Gnus は実際に購読されているグループに関する情報だけを得てやっていこうとします。

気を付けてほしいのは、あなたが山ほどのたくさんのグループを購読しているときにこの変数を nil に設定すると、Gnus は速くなるどころか遅くなってしまうということです。現状では、ニュースを 2400bps 以上のモデムを通して読んでいない限り、Gnus の速度はかなり遅くなるでしょう。

この変数は some という値も取ることができます。その時は、Gnus は購読しているグループに関する情報をだけを得ようとします。いくつかのサーバー (LIST ACTIVE group 命令を使うことのできる、最新鋭の INN サーバー) では、非常に早くなるでしょうが、他のサーバーでは速くはありません。どのようにせよ、遅い回線では some は nil よりも速く、それはもちろん t よりも速くなります。

いくつかのニュースサーバー (例えば古い Leafnode や古い INN) には LIST ACTIVE group 命令がありません。そういうサーバーには nil をこの変数の値に設定するのが、おそらくもっとも有効でしょう。

もしこの変数が nil であると、Gnus は完全にがんじがらめの方法でグループの情報を得ようとします。そして、これはあまり速くありません。もしそれが some で NNTP サーバーを使っている

ときは、Gnus はできるだけ速く命令を出し、一撃ですべての返答を読み込みます。この方が普通はより良い結果をもたらしますが、サーバーが LIST ACTIVE group 命令を理解しないなら、サーバーにとってはあまり良いとは言えません。

Gnus の起動にあまりに時間がかかると思ったなら、この変数にこれらの三つの違った値を試してみ、どれが一番良いかを探して下さい。

some か nil を使うのであれば、どちらにしろ速度を上げるためにすべての興味の無いグループを必ず削除するべきでしょう。

この変数は二次 (secondary) 選択方法のアクティブファイル取得にも影響することに気を付けて下さい。

1.10 起動変数

gnus-load-hook

Gnus (のプログラム) が読み込まれるときに実行されるフックです。何度 Gnus を起動しても、Emacs が起動してから終了するまでに普通はこのフックは一回しか実行されないことに注意して下さい。

gnus-before-startup-hook

Gnus の起動に成功した後に実行されるフックです。

gnus-startup-hook

Gnus が起動された後に、一番最後に実行されるフックです。

gnus-started-hook

Gnus の起動に成功した後に、一番最後に実行されるフックです。

gnus-setup-news-hook

‘.newsrsrc’ ファイルを読み込んだ後で、グループバッファーを作成する前に実行されるフックです。

gnus-check-bogus-newsgroups

もし nil でないと、Gnus は起動時にすべての偽グループを調べて削除します。「偽グループ」(bogus group) はあなたの ‘.newsrsrc’ ファイルには存在するけれど、ニュースサーバーには実際には存在しない、というグループのことです。偽グループを調べるのにはかなり時間がかかるので、時間と資源を節約するために、この機能は使わないほうがいいでしょう。そして、代わりにグループバッファーで時々偽グループを調べるのが良いでしょう (see Section 2.13 [Group Maintenance], page 32)。

gnus-inhibit-startup-message

もし nil でないと、起動時のメッセージは表示されません。そのようにすれば、仕事の代わりにニュースを読んでいるのを上司に気付かれにくくなるでしょう。この変数は ‘~/gnus.el’ がロードされる前に使われるので、‘.emacs’ に設定するべきである点に注意して下さい。

gnus-no-groups-message

グループが一つも存在しないときに Gnus が表示するメッセージです。

gnus-play-startup-jingle

もし nil でないと、起動時に Gnus の短い曲を演奏します。

gnus-startup-jingle

上の変数が nil でないときに演奏される短い曲です。デフォルトは ‘Tuxdemon.Jingle4.au’ です。

2 グループバッファ

グループバッファ (*group buffer*) は有効なグループを全部 (あるいは一部を) 一覧表示します。これは Gnus を起動したときに最初に表示されるバッファで、Gnus が生きている限り決して消されることはありません。

2.1 グループバッファの形式

グループモードのツールバーをカスタマイズすることができます。*M-x customize-apropos RET gnus-group-tool-bar* を試してみてください。この機能を利用できるのは Emacs だけです。

ツールバーのアイコンは、今ではカーソルの位置に応じて正しく有効に、または無効にされるので、グループバッファ内での移動は遅くなります。これは変数 `gnus-group-update-tool-bar` で禁止することができます。そのデフォルト値は Emacs のバージョンに依存しています。

2.1.1 グループ行の仕様

グループバッファのデフォルトの形式はきれいでつまらないけど、これは君の好きなように、サイコーにダサくすることもできます。

これがグループ行の例です。

```
25: news.announce.newusers
*   0: alt.fan.andrea-dworkin
```

とっても簡単でしょ？

‘news.announce.newusers’ には 25 の未読記事があるのがわかります。‘alt.fan.andrea-dworkin’ には未読記事はないけれども、印を付けた記事がいくつかあります (行頭のちっちゃなアスタリスクが見える?)。

この形式は `gnus-group-line-format` 変数をいじることで、どんな風にでも変えられます。この変数は `format` の仕様風に動作します。つまり (あのクソ) C 言語を使う人たちのため、`printf` の仕様とほぼ同じです。See Section 8.4 [Formatting Variables], page 250.

上記の行を生成するのは ‘`%M%S%5y:%B%(g%)\n`’ という値です。

コロンは、この行の中に必ず無くてはいけません。カーソルは何かの操作をした後は常にコロンのところに移動するからです。See Section 8.4.6 [Positioning Point], page 253. 他には何も必要ではありません—グループ名さえもです。表示されている文字はすべてただの画面の飾りであり、Gnus がそれを調べることはありません。Gnus は必要とするすべての実情報を、テキスト属性を使って憶えています。

(もし君が、すごくヘンな、素晴らしい、表計算風のレイアウトを作ったとしたら、みんな、君は会計の仕事が忙しくて、ニュースを読んで時間を無駄使いしたりなんかしてない、って信じてくれるよ。)

以下が使用できるフォーマット文字のリストです。

- ‘M’ そのグループに印の付いた記事しか無いときは、アスタリスク文字。
- ‘S’ そのグループが購読されているかどうか。
- ‘L’ 購読度のレベル。
- ‘N’ 未読記事の数。
- ‘I’ 保留記事の数。

- ‘T’ 印付き記事の数。
- ‘R’ 既読記事の数。
- ‘U’ まだ読まれたことが無い記事の数。
- ‘t’ 推定全記事数 (これは実際は $\text{max-number} - \text{min-number} + 1$)。
Gnus がこの推定を使うのは、NNTP プロトコルは能率の良い max-number と min-number へのアクセスを提供するものの、本当の未読記事の数を得るには必ずしも能率的ではないからです。ヒステリックなレーズン (訳注: 「歴史的な理由」のモジリ) により、メールバックエンドにおいても、限定された同じインターフェースを使って、本当の未読記事の数を能率的に得ることはできるかもしれませんが。この制限を Gnus から取り払うことはバックエンドのインターフェースを変更することを意味し、それは楽な仕事ではありません。
nnml バックエンド (see Section 6.3.13.3 [Mail Spool], page 169) には、この欠陥を巧みに補う「グループ圧縮」(group compaction) という機能があります。それは、記事の番号を 1 から順に振り直してすきまを取り除けば正しい全記事数を得ることができる、という着想によります。将来は他のバックエンドもこれをサポートするかもしれませんが。全記事数をまあまあ最新の状態にしておくためには、時々グループを (またはサーバーのディレクトリーを) 圧縮する必要があるでしょう。See Section 2.17 [Misc Group Stuff], page 38, See Section 6.1.2 [Server Commands], page 134.
- ‘y’ 未読でも、印付きでも、保留でもない記事の数。
- ‘i’ 印付き記事と保留記事の数。
- ‘g’ グループ名のフルネーム。
- ‘G’ グループ名。
- ‘C’ グループのためのコメント (see Section 2.10 [Group Parameters], page 23)、またはグループパラメーターにコメントの要素が無い場合はグループ名。
- ‘D’ ニュースグループの説明。これらが現れる前に、グループの説明を読む必要があります。それには `gnus-read-active-file` を設定するか、グループバッファで `M-d` コマンドを使って下さい。
- ‘o’ 司会者付きの場合 ‘m’。
- ‘O’ 司会者付きの場合 ‘(m)’。
- ‘s’ 選択方法。
- ‘B’ そのグループの概略バッファが開いているかどうか。
- ‘n’ どこからの選択か。(訳注: バックエンドのシンボル名)
- ‘z’ 外部選択方法が使われている場合、‘<%s:%n>’ と同じ文字列。
- ‘P’ トピック (see Section 2.16 [Group Topics], page 33) のレベルに応じた字下げ。
- ‘c’ 短い (省略した) グループ名。gnus-group-uncollapsed-levels 変数は、どのレベルまでグループ名を全部残すかを示します。デフォルトは 1 です—この意味は、‘gnu.emacs.gnus’ のようなグループ名を ‘g.e.gnus’ に短縮するということです。
- ‘m’ そのグループに最近新着メールが届いている場合は ‘%’ (gnus-new-mail-mark)。

- ‘p’ ‘#’ (gnus-process-mark) で、そのグループにプロセス印が付いていることを示します。
- ‘d’ 最後にいつこのグループを読んだかを示す文字列 (see Section 2.17.3 [Group Time-stamp], page 41)。
- ‘F’ キャッシュとエージェントの両方によって取得された記事がディスクに占める容量。値はカラム幅を最小にするために、自動的にバイト (B)、キロバイト (K)、メガバイト (M)、またはギガバイト (G) に縮尺されます。固定幅カラム用には %7F の形式で足ります。
- ‘u’ 利用者定義指定。フォーマット文字列中で、この次の文字はアルファベット文字でなければいけません。Gnus は gnus-user-format-function-‘X’ 関数を呼び出します。ここで ‘X’ は ‘%u’ に続いている文字です。この関数は引数に一つのダミーパラメーターを渡されます。この関数は、他の各指定文字の情報と同様に、バッファに挿入される文字列を返さなければなりません。

すべての「～の数」の指定は、もしその情報が利用できない場合にはアスタリスク (‘*’) で埋められます—例えば、起動されていない外部グループや、不正な基本グループの場合です。

2.1.2 グループモード行の仕様

モード行は gnus-group-mode-line-format (see Section 8.4.2 [Mode Line Formatting], page 251) を設定することで変更できます。こいつは指定文字をあんまりたくさん知っていません。

‘S’ 基本ニュースサーバー。

‘M’ 基本選択方法。

2.1.3 グループのハイライト

グループバッファのハイライトは gnus-group-highlight 変数によって制御されます。これは (form . face) のようなものを要素に持つ連想リストです。form が評価された結果が、nil 以外の何かになると、その行に対して face が使用されます。

以下がこの変数の値の例です。これは背景が暗い設定ではきれいに見えるかもしれません。

```
(cond (window-system
      (setq custom-background-mode 'light)
      (defface my-group-face-1
        '((t (:foreground "Red" :bold t))) "First group face")
      (defface my-group-face-2
        '((t (:foreground "DarkSeaGreen4" :bold t))) "Second group face")
      (defface my-group-face-3
        '((t (:foreground "Green4" :bold t))) "Third group face")
      (defface my-group-face-4
        '((t (:foreground "SteelBlue" :bold t))) "Fourth group face")
      (defface my-group-face-5
        '((t (:foreground "Blue" :bold t))) "Fifth group face"))

      (setq gnus-group-highlight
        '(((> unread 200) . my-group-face-1)
          ((and (< level 3) (zerop unread)) . my-group-face-2)
          ((< level 3) . my-group-face-3))
```

```
((zerop unread) . my-group-face-4)
(t . my-group-face-5)))
```

Section 8.6 [Faces and Fonts], page 258 も参照して下さい。

この form が評価されるときに動的に束縛されている変数には以下のものがあります。

group	グループ名。
unread	そのグループの未読記事の数。
method	選択方法。
mailp	そのグループがメールのグループかどうか。
level	そのグループのレベル。
score	そのグループのスコア。
ticked	そのグループ中の印の付いた記事の数。
total	そのグループ中の全記事数。もっと正確に言うと、 <i>max-number</i> マイナス <i>min-number</i> プラス 1。
topic	トピックマイナーモードを使用している時、この変数は挿入されている現在のトピックに束縛されます。

この form が評価 (eval) されるときは、ポイントは問題のグループの行頭にあります。従って、通常の Gnus の関数のほとんどを使ってそのグループの情報を取ってくることができます。

`gnus-group-update-hook` はグループ行が変更されたときに呼び出されます。これは `gnus-visual` が `nil` のときは呼び出されません。このフックはデフォルトでは `gnus-group-highlight-line` を呼び出します。

2.2 グループ操作

すべての移動コマンドは数値接頭引数を理解するので、期待する通りの動作をします。たぶんね。

<i>n</i>	次の未読記事のあるグループに移動します (<code>gnus-group-next-unread-group</code>)。
<i>P</i>	
<i>DEL</i>	一つ前の未読記事のあるグループに移動します (<code>gnus-group-prev-unread-group</code>)。
<i>N</i>	次のグループに移動します (<code>gnus-group-next-group</code>)。
<i>P</i>	一つ前のグループに移動します (<code>gnus-group-prev-group</code>)。
<i>M-n</i>	一つ前の同じレベル (もしくはそれより小さいレベル) の未読グループに移動します (<code>gnus-group-prev-unread-group-same-level</code>)。
<i>M-p</i>	次の同じレベル (もしくはそれより小さいレベル) の未読グループに移動します (<code>gnus-group-next-unread-group-same-level</code>)。

次の三つの命令はグループにジャンプするためのものです:

<i>j</i>	グループにジャンプします (それが見えるようになっていなかったら見えるようにします) (<code>gnus-group-jump-to-group</code>)。kill されているグループも、生きているグループと同様にジャンプできます。
,	最も小さいレベルの未読グループにジャンプします (<code>gnus-group-best-unread-group</code>)。

- ・ 最初の未読記事のあるグループにジャンプします (`gnus-group-first-unread-group`)。

`gnus-group-goto-unread` を `nil` にすると、すべての移動コマンドは、次の未読グループではなく次のグループに移動するようになります。そのコマンドが次の未読グループに移動すると言い張っていてもです。デフォルトは `t` です。

2.3 グループの選択

SPACE 現在のグループを選択し、概略バッファに切り替えて最初の未読記事を表示します (`gnus-group-read-group`)。もしそのグループに未読記事が無い、もしくはこの命令に数値以外の接頭引数を与えると、Gnus はサーバーからこのグループのすべての古い記事を取得しようとします。 n の数値接頭引数を与えると、Gnus の取得する記事数は n になります。 n が正の数であれば Gnus は新しい方から n 個の記事を取得し、 n が負の数であれば Gnus は古い方から $\text{abs}(n)$ 個の記事を取得します。

したがって、*SPC* では普通にグループに入り、*C-u SPC* では古い記事が現れます。*C-u 4 2 SPC* では 42 個の最新の記事を取得し、*C-u - 4 2 SPC* では 42 個の最も古い記事を取得します。

グループにいる (概略バッファにいる) ときは、*M-g* で新しい記事を取得できるし、*C-u M-g* では古い記事を表示することができます。

RET 現在のグループを選択し、概略バッファに切り替えます (`gnus-group-select-group`)。 `gnus-group-read-group` と同じ引数を取ります—唯一の違いは、グループに入ったときに最初の未読記事を表示しない、ということです。

M-RET これは上記のコマンドと同じ動作をしますが、「ゴタゴタ」は最低限にしようとします (`gnus-group-quick-select-group`)。スコア・kill の処理は行なわれず、ハイライトも記事消去もしません。これは、あなたが本当に急いでいて、どっかのやたらでっかいグループに入らなければいけないときに役に立つかもしれません。また、接頭引数に 0 を与えれば (すなわち *0 M-RET*)、Gnus は概略バッファを作ろうとさえしません。これは概略バッファを作る前にスレッド表示を切り替えたいとき役に立ちます (see Section 3.26.3 [Summary Generation Commands], page 107)。

M-SPACE これは *RET* コマンドと同じ動作をするさらにもう一つのコマンドですが、このコマンドは記事消去と保留記事を隠す処理を行ないません (`gnus-group-visible-select-group`)。

C-M-RET 最後にこのコマンドは、現在のグループを一度限り、その内容に一切の処理をすることのないように選択します (`gnus-group-select-group-ephemerally`)。スレッド表示さえも行なわれません。この方法で選択した後このグループに対して行なったことはすべて、その後に影響を与えることはありません。

`gnus-large-newsgroup` 変数は、何を大きなグループと考えるべきかを Gnus に与えます。`nil` だったら、どのグループも大きいと考えません。デフォルト値は 200 です。グループに (未読と可視の) 記事がこの数以上あれば、Gnus はそのグループに入る前に利用者に確認を求めます。利用者はサーバーからいくつの記事を取得するかを指定できます。もし利用者が負の数 ($-n$) を指定すれば、古い方から n 個の記事を取得します。正の数であれば、新しく到着した方から n 個の記事を取得します。

`gnus-large-ephemeral-newsgroup` は `gnus-large-newsgroup` と同じですが、一時ニュースグループのためにだけ使われます。

もし `gnus-auto-select-first` が非-nil だったら、`SPACE` コマンドでグループに入ったときに自動的に記事を選択します。どの記事が選択されるかは、変数 `gnus-auto-select-subject` で制御されます。この変数に設定できる有効な値は:

`unread` 最初の未読記事の表題の行にポイントを移動させます。

`first` 最初の記事の表題の行にポイントを移動させます。

`unseen` まだ読まれたことが無い最初の記事の表題の行にポイントを移動させます。

`unseen-or-unread`
 まだ読まれたことが無い最初の記事があれば、その記事の表題の行にポイントを移動させ、無かったら最初の未読記事の表題の行にポイントを移動させます。

`best` スコアが最も高い未読記事の表題の行にポイントを移動させます。

この変数は関数であることもできます。その場合、その関数は表題の行にポイントを移動させるために呼ばれます。

もしあるグループで自動記事選択をやめたいのであれば (例えばでっかい記事のあるバイナリーグループでは、とか)、グループが選択されたときに呼び出される `gnus-select-group-hook` の中で変数 `gnus-auto-select-first` を `nil` に設定することができます。

2.4 購読制御コマンド

`S t`
`u` 現在のグループを購読する/しないを切り替えます (`gnus-group-unsubscribe-current-group`)。

`S s`
`U` グループを購読するかどうかを確認し、購読します。すでに購読している場合には、購読を止めます (`gnus-group-unsubscribe-group`)。

`S k`
`C-k` 現在のグループを `kill` します (`gnus-group-kill-group`)。

`S y`
`C-y` 最後に `kill` したグループを `yank` します (`gnus-group-yank-group`)。

`C-x C-t` 二つのグループの順序を置き換えます (`gnus-group-transpose-groups`)。これは本当は購読コマンドではありませんが、`kill` と `yank` を何度か続ける代わりにこのコマンドが使えます。

`S w`
`C-w` リージョン内のすべてのグループを `kill` します (`gnus-group-kill-region`)。

`S z` すべてのゾンビグループを `kill` します (`gnus-group-kill-all-zombies`)。

`S C-k` あるレベルのグループをすべて `kill` します (`gnus-group-kill-level`)。kill した後、これらのグループを `yank` で戻すことはできないので、このコマンドはいくらか注意して使って下さい。このコマンドが本当に便利になるのは、`‘.newsrc’` に捨ててしまいたい未購読のグループがたくさんあるときだけです。レベル 7 で `S C-k` を行なうと、`‘.newsrc’` ファイル中にメッセージ番号がない未購読グループをすべて `kill` します。

Section 2.6 [Group Levels], page 19 も参照して下さい。

2.5 グループデータ

- c* そのグループ内のすべての無印の記事を既読にします (gnus-group-catchup-current)。グループバッファから既読にした場合は gnus-group-catchup-group-hook が呼び出されます。
- C* そのグループの全記事を、印付きの記事も含めて既読にします (gnus-group-catchup-current-all)。
- M-c* 現在のグループのすべてのデータをクリアします—印と既読記事のリストを消し去ります (gnus-group-clear-data)。
- M-x gnus-group-clear-data-on-native-groups*
 もし NNTP サーバーを別のものに切り替えたとなると、すべての印と既読情報はもう役には立ちません。このコマンドを使って基本グループのすべてのデータをクリアすることができます。注意して使ってね。

2.6 グループレベル

すべてのグループは「購読度」(*subscribedness*) のレベルを持ちます。例えば、あるグループがレベル 2 だとすれば、それはレベル 5 のグループよりも「より購読している」ということです。Gnus に対して、あるレベルかそれより小さいレベルのグループのみ一覧表示するように頼むこともできるし (see Section 2.11 [Listing Groups], page 29)、あるレベルかそれより小さいレベルのグループの新着記事のみを確認することもできます (see Section 2.17.1 [Scanning New Messages], page 40)。

忘れないで: グループのレベルが大きいほど、重要度は低くなるということ。

- S 1* 現在のグループのレベルを設定します。数値の接頭引数が与えられると、そこから *n* 個のグループのレベルが設定されます。レベルを入力するためのプロンプトが出ます。

Gnus はレベル 1 から gnus-level-subscribed (この値を含む) (デフォルトは 5) までのグループを購読、gnus-level-subscribed (この値を含まない) から gnus-level-unsubscribed (この値を含む) (デフォルトは 7) までのグループを非購読、gnus-level-zombie をゾンビ (歩く屍) (デフォルトは 8)、gnus-level-killed を kill されている (完全に死んでいる) (デフォルトは 9) と判断します。Gnus は購読と非購読のグループはまったく同様に扱いますが、ゾンビと kill グループは、どの記事を読んだか、存在するかなどの情報を一切持ちません。この死んでいるグループと生きているグループの区別は、別にそれがきれいだからとか賢いからというわけではなく、純粋に効率的な理由のためです。

メール用のグループは (もしあれば) 非常に小さいレベル (例えば 1 か 2) にしておくことをお勧めします。

次の Gnus のデフォルトの動作の説明は、ことによると、これらのレベルのすべてを理解する助けになるかもしれません。デフォルトでは、Gnus は購読している空でないグループを表示しますが、*L* を叩くことによって空のグループや非購読のグループも表示させることができます。つまり、非購読のグループは隠されている、と言っても良いでしょう。

ゾンビと kill グループは、デフォルトでは隠されている点で非購読のグループに似ています。しかし、Gnus がニュースサーバーに対してゾンビと kill グループに関する情報 (記事数、未読記事数) の問い合わせをしない点で、購読および非購読のグループとは違ってきます。ふつう、あなたは興味の無いグループを *C-k* で kill しますよね。もし、ほとんどのグループが kill されていると、Gnus は速くなります。

なぜ Gnus はゾンビと kill グループを区別するのでしょうか? ええと、サーバーに新しいグループができると、Gnus はデフォルトでそれをゾンビにします。これは、あなたがふつうは新しいグ

グループに煩わされないことを意味しますが、あなたは `A z` で新しいグループのリストを得ることができます。あなたは好みのものを購読し、要らないものは `kill` すれば良いのです。(`A k` で `kill` されたグループのリストを表示します。)

もしレベル変数で遊びたいのであれば、多少注意をしてまわる必要があります。いったんそれを設定したら、二度とそれに触らないで下さい。さらに言えば、自分で何をやっているかを正確に理解していない限り、一切触らないで下さい。

身近に関係する二つの変数は `gnus-level-default-subscribed` (デフォルトは 3) と `gnus-level-default-unsubscribed` (デフォルトは 6) です。これらは新しいグループが (非) 購読されたときのレベルです。もちろん、これら二つの変数の値は、意味のある正しい範囲でなくてはなりません。

`gnus-keep-same-level` が `nil` 以外であれば、移動コマンドのいくつかは同一 (あるいはそれより小さい) レベルのグループのみの移動になります。特に、あるグループの最後の記事から次のグループに移るとき、次の同一 (あるいはそれより小さい) レベルのグループに移動します。これは残りのグループを読むより先に、より重要なグループを読んでおきたいときには便利かもしれません。

もしこの値が `best` だったら、最も重要な (最もレベルの値が小さい) グループに移動します。

デフォルトでは `gnus-group-default-list-level` と同じかそれより小さいレベルのグループが、グループバッファに一覧表示されます。

`gnus-group-list-inactive-groups` が `nil` 以外であれば、未読のグループにアクティブでないグループも一緒に表示します。この変数はデフォルトでは `t` です。もしこれが `nil` であれば、アクティブでないグループは表示されません。

`gnus-group-use-permanent-levels` が `nil` 以外であれば、いったん `g` や `l` コマンドの接頭引数にレベルを与えると、その後のすべてのコマンドにおいてそのレベルが「作用する」レベルになります。

Gnus は通常、`gnus-activate-level` がそれより小さいレベルのグループのみを起動します (つまりサーバーに問い合わせをする)。購読していないグループを起動したくなければ、この変数を例えば 5 に設定するとよいかもしれません。デフォルトは 6 です。

2.7 グループのスコア

普通は重要なグループは高レベルにしておくでしょうけれども、この方法では少々制限がきついですよね。ひょっとしたら、グループをどれくらい頻繁に読むかによってグループバッファを並べ替えたくなあ、なんて思いませんか? 理にかなってるでしょ?

「グループスコア」(*group score*) はそのためのものです。Gnus に以下で説明されている機構で、それぞれのグループに対してスコアを指定することができます。そしてグループバッファをこのスコアを基に並べ替えることができます。あるいは、スコア順で並べ替えてその後レベルで並べ替えることもできます。(レベルとスコアをひとまとめにして、グループの「ランク」(*rank*) と呼びます。レベルが 4 でスコアが 1 のグループは、レベルが 5 でスコアが 300 のグループよりも高いランクとなります。(レベルの方が重要度が高く、スコアの方は重要度が低くなります。))

頻繁に読むグループに、めったに読まないグループよりも高いスコアを与えたいときは、`gnus-summary-exit-hook` フックに `gnus-summary-bubble-group` 関数を追加することができます。これでバブル並べ替えの実行結果が (並べ替えの後で) 得られるでしょう。概略モードを終了するたびにこの活動をさせたいのであれば、同じフックに `gnus-group-sort-groups-by-rank` または `gnus-group-sort-groups-by-score` を追加できますが、いくらか遅くなるでしょう。

2.8 グループへの印

もしいくつかのグループに対して何らかの命令を実行したい場合で、それらがグループバッファに連続してある場合には、通常通り命令に対して数値接頭引数を与えるだけです。そうすればほとんどのグループ命令は、これらのグループに対してあなたの命令に従います。

しかしそれらのグループが順番に並んでいない場合においても、いくつかのグループに対して命令を実行することができます。単に始めにプロセス印でグループに印を付けておき、そして命令を実行するだけです。

#	
<i>M m</i>	現在のグループに印を付ける (gnus-group-mark-group)。
<i>M-#</i>	
<i>M u</i>	現在のグループから印を削除する (gnus-group-unmark-group)。
<i>M U</i>	すべてのグループから印を削除する (gnus-group-unmark-all-groups)。
<i>M w</i>	ポイントとマークの間のすべてのグループに印を付ける (gnus-group-mark-region)。
<i>M b</i>	バッファ内のすべてのグループに印を付ける (gnus-group-mark-buffer)。
<i>M r</i>	ある正規表現に合致するすべてのグループに印を付ける (gnus-group-mark-regexp)。

Section 8.1 [Process/Prefix], page 249 も参照して下さい。

プロセス印が付けられているすべてのグループに対して何かの命令を実行したいときは、*M-&* (gnus-group-universal-argument) 命令を使うことができます。プロンプトから実行したい命令を入力します。

2.9 外部グループ

以下では、一般的な外部グループの作成、変更を行なうグループモードの命令をいくつか、および特別な目的のグループを簡単に作成する命令を紹介します。これらの命令はすべて、新規に作成したグループをポイント位置に挿入します—gnus-subscribe-newsgroup-method は参照されません。

これらグループを編集するコマンド群で行なった変更は ‘~/newsrsrc.eld’ (gnus-startup-file) に格納されます。代わりの手段として、変数 gnus-parameters も用意されています (see Section 2.10 [Group Parameters], page 23)。

<i>G m</i>	新しいグループを作成します (gnus-group-make-group)。Gnus はプロンプトを表示して、名前と方法と、場合によっては <i>address</i> の入力を求めてきます。より簡単に NNTP グループを購読する方法については、Section 2.14 [Browse Foreign Server], page 32 を参照して下さい。
<i>G M</i>	一時ニュースグループを作成します (gnus-group-read-ephemeral-group)。Gnus はプロンプトを表示して、名前、方法および <i>address</i> の入力を求めます。
<i>G r</i>	現在のグループの名前を、何か別のものに変更します (gnus-group-rename-group)。これはある種のグループ—主にメールグループに対してのみ有効です。このコマンドはバックエンドによっては非常に遅いことも有り得ます。
<i>G c</i>	グループパラメーターをカスタマイズする (gnus-group-customize)。
<i>G e</i>	現在のグループの選択方法を修正するためのバッファに移動します (gnus-group-edit-group-method)。

- G p* グループパラメーターを修正するためのバッファーに移動します (gnus-group-edit-group-parameters)。
- G E* グループ情報を修正するためのバッファーに移動します (gnus-group-edit-group)。
- G d* ディレクトリーグループを作成します (see Section 6.6.1 [Directory Groups], page 194)。ディレクトリー名をプロンプトで入力します (gnus-group-make-directory-group)。
- G h* Gnus ヘルプグループを作成します (gnus-group-make-help-group)。
- G a* Gnus アーカイブグループを作成します (gnus-group-make-archive-group)。デフォルトでは最も最近の記事を指しているグループが作成されますが (gnus-group-recent-archive-directory)、接頭引数を与えるとすべての記事を含むグループが gnus-group-archive-directory を基に作成されます。
- G k* kiboze グループを作成します。プロンプトで名前と、kiboze グループに「含めたい」グループに合う正規表現と、ヘッダーに合う文字列の組を入力します (gnus-group-make-kiboze-group)。See Section 6.7.2 [Kibozed Groups], page 204.
- G D* 任意のディレクトリーを mneething バックエンドニュースグループであるかのように読み込みます (gnus-group-enter-directory)。See Section 6.6.2 [Anything Groups], page 195.
- G f* 何らかのファイルをもとにグループを作成します (gnus-group-make-doc-group)。このコマンドに接頭引数を与えた場合、ファイル名とファイルタイプをプロンプトで入力します。現在サポートされているファイルタイプは mbox, babyl, digest, news, rnews, mmdf, forward, rfc934, rfc822-forward, mime-parts, standard-digest, slack-digest, clari-briefs, nsmail, outlook, oe-dbx および mailman です。接頭引数なしでこのコマンドを実行すると、Gnus はファイルタイプを推測します。See Section 6.6.3 [Document Groups], page 196.
- G u* gnus-useful-groups にあるグループの一つを作ります (gnus-group-make-useful-group)。
- G w* ウェブ検索結果をもとに一時的なグループを作成します (gnus-group-make-web-group)。このコマンドに接頭引数を与えると、一時的ではなく固定したグループを作成します。プロンプトで検索エンジンの種類 (search engine type) と検索文字列を入力します。有効な検索エンジンの種類には google, dejanews, gmane があります。See Section 6.4.2 [Web Searches], page 180.
- もし、google 検索エンジンを用いる場合には、‘shaving group:alt.sysadmin.recovery’ のような合致する文字列を用いることによって、検索対象を特定のグループに限定することが可能です。
- G R* RSS feed に基づくグループを作ります (gnus-group-make-rss-group)。URL の入力を促されます。See Section 6.4.6 [RSS], page 183.
- G DEL* この関数は現在のグループを削除します (gnus-group-delete-group)。接頭引数を与えられると、この関数はそのグループ内の全記事を本当に削除し、グループ自身をこの世から強制的に抹殺してしまいます。接頭引数は、あなたが何をやろうとしているか、本当に自信があるときにのみ使って下さい。まあ、このコマンドは (nntp グループのような) 読み出し専用グループには使えませんけれど。

- G V* 新しい、新鮮な、空の `nnvirtual` グループを作成します (`gnus-group-make-empty-virtual`)。See Section 6.7.1 [Virtual Groups], page 203.
- G v* 現在のグループを `nnvirtual` グループに追加します (`gnus-group-add-to-virtual`)。これはプロセス印/接頭引数の習慣に従います。

さまざまな選択方法に関するさらなる情報は Chapter 6 [Select Methods], page 133 を参照して下さい。

もし `gnus-activate-foreign-newsgroups` が正の数であれば、Gnus は起動時に、この数がそれよりも小さいレベルの外部グループをすべてチェックします。これは特に違った NNTP サーバーからたくさんのグループを購読している場合には、しばらく時間がかかるかもしれません。Section 2.6 [Group Levels], page 19 も参照して下さい。 `gnus-activate-level` も外部ニュースグループの活性化に影響を及ぼします。

2.10 グループパラメーター

グループパラメーターは、ある特定のグループに固有な情報を保持します。以下はグループパラメーターリストの例です：

```
((to-address . "ding@gnus.org")
 (auto-expire . t))
```

それぞれの要素は『点対』(dotted pair)—つまり点 (dot) の前に鍵、点の後ろに値があるもので構成されます。すべてのパラメーターはこの形式を取りますが、例外としてローカル変数の指定は点対ではなく通常のリスト (訳注: 後述の (variable form) の項を参照) になります。

いくつかのパラメーターは対応するカスタマイズ可能な変数を持っています。それらは正規表現と値の連想リストです。

以下は利用可能なグループパラメーターです：

`to-address`

フォローアップとニュースへの投稿をするときに使用されるアドレス。

```
(to-address . "some@where.com")
```

これは主に、閉じたメーリングリストを表わすメールグループにおいて便利なものです—すなわちメーリングリストに投稿する人はすべてそれを購読しているはず、というメーリングリストのことです。このパラメーターを使用すると、メールはそのメーリングリストにしか投稿されないことが保証されるので、参加者はあなたのフォローアップ記事を二通受け取ることはありません。

`to-address` を指定すると、そのグループが外部グループであるかどうかに関わらず有効になります。例えば `'fa.4ad-1'` というグループがサーバー上にあったとしましょう。これは本当のニュースグループですが、サーバーはメールニュースゲートウェイを通して記事を受け付けます。つまりこのグループに対して直接投稿することは不可能で、代わりにそのメーリングリストにメールを送信しなければなりません。

`gnus-parameter-to-address-alist` も参照して下さい。

`to-list`

そのグループで `a` を押したときに使用されるアドレス。

```
(to-list . "some@where.com")
```

これはフォローアップをしたときは完全に無視されます—例外はそれがニュースグループを表わしているときは、`f` を押したときにメールグループのルールが適用されるということです。

もし `a` コマンドをメールグループで実行したときに、`to-list` グループパラメーターも `to-address` もグループパラメーターも無ければ、`to-list` グループパラメーターは、`gnus-add-to-list` が `t` に設定されていればメッセージ送信時に自動的に付加されます。

もしこのグループパラメーターが設定されていると、概略バッファに入ったときに `gnus-mailing-list-mode` が有効になります。

`gnus-parameter-to-list-alist` も参照して下さい。

subscribed

もしこのパラメーターが `t` に設定されていると、Gnus はあなたがこのグループを `to-address` と `to-list` パラメーターのアドレスで購読しているメーリングリストであると解釈します。この情報を Gnus に与えることは、あなたがそれらのメーリングリストに投稿するときに正しい Mail-Followup-To ヘッダーを生成するための (ほんの) 第一歩です。二歩目は `gnus.el` に以下を入れることです。

```
(setq message-subscribed-address-functions
      '(gnus-find-subscribed-addresses))
```

利用できる MFT 対応機能を完全に扱うには、ここ (see section “メーリングリスト” in *The Message Manual*) を見て下さい。

visible グループパラメーターのリスト中に (`visible . t`) という要素があれば、そのグループはグループバッファにおいて、未読記事があるかどうかに関わらず、常に表示されます。

このパラメーターを `gnus-parameters` を介して設定することはできませんが、代わりに `gnus-permanently-visible-groups` を使えば良いでしょう。

broken-reply-to

(`broken-reply-to . t`) という要素があれば、そのグループでは Reply-To は無視され、`reply-to` が `gnus-boring-article-headers` の部分であれば、ヘッダーが隠されるという意味です。これはある `listserv` によるメーリングリストを購読していて、それが Reply-To 欄を `listserv` 自身に返すように付けられている場合に有効でしょう。これはおかしな振る舞いです。だからこれが要るんです!

to-group (`to-group . "some.group.name"`) という要素は、そのグループへの投稿はすべて `some.group.name` に送られる、という意味です。

newsgroup

グループパラメーターリストに (`newsgroup . t`) があれば、Gnus はすべての応答をニュース記事に対する応答であるかのように扱います。これは実際にはニュースグループのミラーであるメールグループに対して有効です。

gcc-self グループパラメーターリストに (`gcc-self . t`) があれば、新しく作成するメッセージは現在のグループに Gcc されます。もし (`gcc-self . none`) があれば、Gcc: 欄は生成されず、(`gcc-self . "string"`) があればこの文字列はそのまま gcc 欄に挿入されます。このパラメーターは以下で説明するどんなデフォルトの Gcc の規則よりも優先されます (see Section 5.5 [Archived Messages], page 126)。

警告: `nntp` (またはその種の) グループのパラメーターリストに (`gcc-self . t`) を加えることに効力はありません。nntp サーバーは記事を受け入れません。

auto-expire

グループパラメーターに (auto-expire . t) のような要素があれば、すべての既読記事は期限切れ消去されるように印が付けられます。他の方法は、See Section 6.3.9 [Expiring Mail], page 163.

gnus-auto-expirable-newsgroups も参照して下さい。

total-expire

グループパラメーターに (total-expire . t) のような要素があれば、既読記事は、期限切れ消去の印が付いていなくてもすべて期限切れ消去処理を施されます。注意して使用して下さい。未読記事、印付き記事、保留記事は期限切れ消去されません。

gnus-total-expirable-newsgroups も参照して下さい。

expiry-wait

グループパラメーターに (expiry-wait . 10) のような要素があれば、この値は記事を期限切れ消去するときに nnmail-expiry-wait と nnmail-expiry-wait-function の設定 (see Section 6.3.9 [Expiring Mail], page 163) よりも優先されます。この値は期限切れ消去の日数 (整数である必要はない) かもしくは never か immediate のシンボルを指定できます。

expiry-target

期限切れ消去されるメッセージの果てる場所。このパラメーターは nnmail-expiry-target よりも優先されます。

score-file

(score-file . "file") のような要素は、'file' を現在のグループに適用されるスコアファイルにします。すべての適用されるスコア・エントリーはこのファイルに入ります。

adapt-file

(adapt-file . "file") のような要素は、'file' を現在のグループの適応ファイルにします。すべての適応スコア・エントリーはこのファイルに入ります。

admin-address

メーリングリストから脱会するときは、脱会通知メールをそのメーリングリスト自身に送信してはいけません。代わりに管理用アドレスにメッセージを送信します。このパラメーターにはどこか都合の良い管理用アドレスを書いておくことができます。

display

(display . MODE) のような要素は、グループに入るときにどの記事を表示するかを指定します。有効な値は、

all 未読、既読記事の両方をすべて表示します。

an integer

そのグループの最後の *integer* 個の記事を表示します。これは C-u *integer* でそのグループに入るのと同じです。

default デフォルトの記事を表示します。これは通常は未読記事と可視記事です。

配列 述語を満足するように記事を表示します。

いくつか例を挙げます:

[unread] 未読の記事だけを表示します。

[not expire]

期限切れ消去可能な記事以外のすべてを表示します。

[and (not reply) (not expire)]

期限切れ消去可能とすでに返信した記事以外のすべてを表示します。

利用できる演算子は not, and および or です。述語は tick, unsend, undownload, unread, dormant, expire, reply, killed, bookmark, score, save, cache, forward, unseen および recent を含みます。

display パラメーターは、概略バッファを指定した一部の組だけに制限するように働きます。制限を外すのは / w コマンドでできます (see Section 3.8 [Limiting], page 61)。

comment (comment . "This is a comment") のような要素は、そのグループに対する任意のコメントです。グループ行に表示することができます (see Section 2.1.1 [Group Line Specification], page 13)。

charset (charset . iso-8859-1) のような要素は、iso-8859-1 をデフォルトの文字セットにします。すなわち、文字セットを指定しないすべての記事に、その文字セットが使われます。

gnus-group-charset-alist も見て下さい。

ignored-charsets

(ignored-charsets x-unknown iso-8859-1) のような要素は、iso-8859-1 と x-unknown を無視します。すなわち、記事のデコードにデフォルトの文字セットが使われます。

gnus-group-ignored-charsets-alist も見て下さい。

posting-style

このグループの追加の投稿様式をここに保存することができます (see Section 5.6 [Posting Styles], page 128)。書式は gnus-posting-style 連想リストと同じですが、ここにはグループ名に合致する正規表現はありません (当然です)。このグループの様式の要素は gnus-posting-styles で見つかったものよりも優先されます。

例えば、このグループのみ、かっこいい名前と署名にしたいなら、gnus-posting-styles をいじらずに、このようなものをグループパラメーターに入れることができます:

```
(posting-style
  (name "Funky Name")
  ("X-My-Header" "Funky Value")
  (signature "Funky Signature"))
```

グループバッファを整理するためにトピック (see Section 2.16 [Group Topics], page 33) を使っている場合は、トピックパラメーターでも投稿様式を設定することができます。トピックパラメーターにある投稿様式は、そのトピックのすべてのグループに適用されます。もっと正確に言うと、あるグループのための投稿様式の設定は、そのグループおよびそれが属するすべてのトピックのパラメーターにあるすべての投稿様式の設定を、階層的に合併することによって生成されます。

post-method

もしこれが設定されていると、メッセージを送信するための選択方法として gnus-post-method の代わりに使われます。

banner (banner . regex) のような項目は、記事のすべての場所で正規表現 *regex* に合致するものを削除します。regex の代わりにシンボル *signature* (最後の署名を削除) や連想リスト *gnus-article-banner-alist* の各要素を使うこともできます。

sieve このパラメーターは、入ってきたメールがこのグループに置くに値するかどうかを調べる Sieve (ふるい) テストを持ちます。このグループパラメーターを元に 'fileinto "group.name";' というテスト条件を本体に持つ、Sieve の 'IF' 制御構造体が作られます。

例えば、もし 'INBOX.list.sieve' グループが (sieve address "sender" "sieve-admin@extundo.com") というグループパラメーターを持っていたならば、グループパラメーターを Sieve スクリプトに変換する (see Section 2.17.5 [Sieve Commands], page 42) ときに、以下の Sieve コードが作られます:

```
if address "sender" "sieve-admin@extundo.com" {
  fileinto "INBOX.list.sieve";
}
```

複数の電子メールアドレスのためのテストを生成するには、(sieve address "sender" ("name@one.org" else@two.org")) のようなグループパラメーターを使って下さい。Sieve スクリプト (see Section 2.17.5 [Sieve Commands], page 42) を生成すると、以下のような Sieve コードが作られます:

```
if address "sender" ["name@one.org", "else@two.org"] {
  fileinto "INBOX.list.sieve";
}
```

Sieve パラメーターに関連する重要なコマンドと変数については、Section 2.17.5 [Sieve Commands], page 42 を参照して下さい。

Sieve 言語は RFC 3028 で述べられています (see section "Top" in *Emacs Sieve*)。

(agent parameters)

エージェントを使うようにしてあると、個々のグループでエージェントの振る舞いを制御するどのパラメーターも設定することができます。エージェントパラメーターについては Section 6.9.2.1 [Category Syntax], page 212 を参照して下さい。たいいていの利用者は、設定に要する苦勞を最小限にするために、エージェントカテゴリーかグループトピックのどちらかでエージェントパラメーターを設定することを選ぶでしょう。

(variable form)

グループに入るときに、そのグループローカルの変数を設定するグループパラメーターを使用することができます。'news.answers' においてスレッド表示を行ないたくないときは、そのグループにグループパラメーターに (gnus-show-threads nil) と書けます。gnus-show-threads は、その概略バッファーの中のローカル変数になり、form の nil はそこで eval (評価) されます。

この機能は、変数を概略バッファーでローカルに設定することに注意して下さい。でも、いくつかの変数は記事バッファーか (返信、フォロー、あるいは新規に作られたメッセージの) メッセージバッファーで評価されます。代わりに、問題の変数を gnus-newsgroup-variables に加えることが助けになるかもしれません。したがって、グループパラメーターを介して message-from-style を設定したいならば、'~/ .gnus' ファイルのどこか他のところに、次の述語が必要になるかもしれません:

```
(add-to-list 'gnus-newsgroup-variables 'message-from-style)
```

この機能の用途の一つは、記事の表題欄からメーリングリストの標識タグをはぎ取ることです。もしニュースグループ

```
nntp+news.gnus.org:gmane.text.docbook.apps
```

が、すべての記事の表題に 'DOC-BOOK-APPS:' というタグを持っているならば、そのグループのグループパラメーターに (gnus-list-identifiers "DOCBOOK-APPS:") を入れることによって、そのグループの概略バッファーに表示される記事の表題からタグをはぎ取ることができます。

これはもし必要であれば、グループ毎のフック関数としても使用できます。もしあるグループに入ったときにピープ音を鳴らしたければ、そのグループのパラメーターに (dummy-variable (ding)) みたいなものを書いておくこともできます。dummy-variable という変数に (無意味な) (ding) の評価結果が設定されます。

あるいは、variable はそのグループに対してローカルになるので、この様式は一時的にフックを変更するために使うことができます。例えば、以下のものがグループパラメーターに追加されると、

```
(gnus-summary-prepared-hook
 '(lambda nil (local-set-key "d" (local-key-binding "n"))))
```

そのグループに入ったときに d キーは記事に期限切れ消去の印を付けないようになります。

グループパラメーターの修正には Gp か Gc 命令を使って下さい (Gp は Lisp ベースの、Gc は Custom ふうのインターフェースを提供します)。トピックパラメーターについて読んでみることも面白いでしょう (see Section 2.16.5 [Topic Parameters], page 37)。

グループパラメーターは gnus-parameters 変数を介在して設定することもできます。でもいくつかのパラメーター、例えば visible は効力を発揮しません (その場合、代替として gnus-permanently-visible-groups を使うことができます)。例です:

```
(setq gnus-parameters
 '(("mail\\.*"
   (gnus-show-threads nil)
   (gnus-use-scoring nil)
   (gnus-summary-line-format
    "%U%R%z%I%([%d:%ub%-23,23f%]) %s\n")
   (gcc-self . t)
   (display . all))

  ("^nnimap:\\(foo.bar\\)$"
   (to-group . "\\1"))

  ("mail\\.me"
   (gnus-use-scoring t))

  ("list\\.*"
   (total-expire . t)
   (broken-reply-to . t))))
```

文字列の値は、to-group の例が示すように、正規表現による置き換えを受けることがあります。

グループ名と gnus-parameters で指定されたこれらの正規表現の一つを比較するときに大文字と小文字を区別するかどうかは、デフォルトではその比較を行なう時点での case-fold-search

の値に依存します。一般的に `case-fold-search` の値は `t` で、それは例えば (`"INBOX\\.FOO"` (`total-expire . t`)) という要素が、`'INBOX.FOO'` グループと `'INBOX.foo'` グループの両方に適用されることを意味します。これらの正規表現が常に大文字と小文字を区別するようにしたい場合は、`gnus-parameters-case-fold-search` 変数の値を `nil` に設定して下さい。あるいは、それらが常に大文字と小文字を区別しないようにしたいなら、それを `t` に設定して下さい。

`gnus-parameters` を介することによって、グループによって異なる並べ替えを定義することができます。これは、NNTP グループでは最新のニュースが先頭になるように日付で、RSS グループでは表題で、それぞれ並べ替えを行なう例です。この例の最初のグループは、`news.gmane.org` から取得する Debian のデイリーニュースです。RSS グループは RSS フィードで配信されている Debian のウィークリーニュース `http://packages.debian.org/unstable/newpkg_main.en.rdf` に対応します。See Section 6.4.6 [RSS], page 183.

```
(setq
  gnus-parameters
  '(("nntp.*gmane\\.debian\\.user\\.news"
    (gnus-show-threads nil)
    (gnus-article-sort-functions '((not gnus-article-sort-by-date)))
    (gnus-use-adaptive-scoring nil)
    (gnus-use-scoring nil))
    ("nnrss.*debian"
    (gnus-show-threads nil)
    (gnus-article-sort-functions 'gnus-article-sort-by-subject)
    (gnus-use-adaptive-scoring nil)
    (gnus-use-scoring t)
    (gnus-score-find-score-files-function 'gnus-score-find-single)
    (gnus-summary-line-format "%U%R%Z%d %I%([ %s %])\n")))))
```

2.11 グループの一覧表示

これらのコマンドは、利用できるグループをいろいろに切り分けて表示します。

- `l`
- `A s` 未読記事を持つすべてのグループを表示します (`gnus-group-list-groups`)。数値接頭引数を使うと、このコマンドは引数の数かそれよりも小さいレベルのグループのみを表示します。デフォルトでは、これはレベル 5 (つまり `gnus-group-default-list-level`) かそれよりも小さいレベル (すなわち購読しているグループのみ) を表示します。
- `L`
- `A u` 未読記事のあるなしに関わらず、すべてのグループを表示します (`gnus-group-list-all-groups`)。数値接頭引数を使用すると、このコマンドは引数の数かそれよりも小さいレベルのグループのみを表示します。デフォルトでは 7 かそれよりも小さいレベルのグループ (すなわち購読、非購読のグループのみ) が表示されます。
- `A l` 未読記事があるグループのうち、指定したレベルのものだけを表示します (`gnus-group-list-level`)。接頭引数を与えると、未読記事の無いグループも含めて表示します。
- `A k` `kill` されたグループをすべて表示します (`gnus-group-list-killed`)。接頭引数を与えると、購読または非購読のどちらにもなっていないすべての利用可能なグループを表示します。これはサーバーからアクティブファイルを読むことになるでしょう。

- A z* すべてのゾンビグループを表示します (gnus-group-list-zombies)。
- A m* 正規表現に合致する名前を持つグループで、未読記事のある購読グループをすべて表示します (gnus-group-list-matching)。
- A M* 正規表現に合致するグループを表示します (gnus-group-list-all-matching)。
- A A* 今接続しているサーバーのアクティブファイルにあるグループを、本当に全部表示します (gnus-group-list-active)。これはしばらく時間がかかることも有り得ます。たぶん *A M* を実行して、合致させたい部分を ‘.’ としてすべての合致するリストを表示させた方が良いでしょう。また、このコマンドは (まだ) 存在しないグループも表示するかもしれませんが—これは kill されたグループであるかのように表示されます。出力は多少割り引いて受け取ってね。
- A a* 正規表現に合致する名前を持つグループをすべて表示します (gnus-group-apropos)。
- A d* 正規表現に合致する名前か説明文を持つグループをすべて表示します (gnus-group-description-apropos)。
- A c* キャッシュ記事を持つグループをすべて表示します (gnus-group-list-cached)。
- A ?* 保留記事を持つグループをすべて表示します (gnus-group-list-dormant)。
- A /* 現在の選択された範囲に限定したグループを表示します (gnus-group-list-limit)。
- A f* 現在の選択されたグループを書き出します (gnus-group-list-flush)。
- A p* 現在の選択されたグループを加えたグループを表示します (gnus-group-list-plus)。

gnus-permanently-visible-groups 正規表現に合致するグループは、未読記事があるかないかに関わらず常に表示されます。あるいはグループパラメーターにおいて visible 要素を追加することでも同様の効果を得ることができます。

印付きの記事のみを持つグループは通常グループバッファーに表示されます。もし gnus-list-groups-with-ticked-articles が nil であれば、そのグループは完全に空のグループであるかのように扱われます。デフォルトは t です。

2.12 グループの並べ替え

C-c C-s (gnus-group-sort-groups) 命令は、グループバッファーを gnus-group-sort-function 変数で与えられる関数に従って並べ替えます。利用可能な並べ替え関数 (sorting function) には以下のものがあります:

gnus-group-sort-by-alphabet

グループ名でアルファベット順に並べ替えます。これがデフォルトです。

gnus-group-sort-by-real-name

グループを本当の (前に何も付いていない) グループ名でアルファベット順に並べ替えます。

gnus-group-sort-by-level

グループレベルで並べ替えます。

gnus-group-sort-by-score

グループのスコアで並べ替えます。See Section 2.7 [Group Score], page 20.

`gnus-group-sort-by-rank`

グループのスコアで並べ替え、次にグループレベルで並べ替えます。レベルとスコアは、ひとまとめにして「ランク」と呼ばれます。See Section 2.7 [Group Score], page 20.

`gnus-group-sort-by-unread`

未読記事の数で並べ替えます。

`gnus-group-sort-by-method`

選択方法のアルファベット順で並べ替えます。

`gnus-group-sort-by-server`

サーバー名のアルファベット順で並べ替えます。

`gnus-group-sort-function` は並べ替え関数のリストであっても構いません。この場合、もっとも重要な並べ替えの鍵を持つ関数は最後でなくてはなりません。

ある種の並べ替え用には、直接並べ替える命令もいくつかあります。

G S a グループバッファーをグループ名のアルファベット順で並べ替えます (`gnus-group-sort-groups-by-alphabet`)。

G S u グループバッファーを未読記事の数で並べ替えます (`gnus-group-sort-groups-by-unread`)。

G S l グループバッファーをグループレベルで並べ替えます (`gnus-group-sort-groups-by-level`)。

G S v グループバッファーをグループのスコアで並べ替えます (`gnus-group-sort-groups-by-score`)。See Section 2.7 [Group Score], page 20.

G S r グループバッファーをグループのランクで並べ替えます (`gnus-group-sort-groups-by-rank`)。See Section 2.7 [Group Score], page 20.

G S m グループバッファーをバックエンドの名前でアルファベット順に並べ替えます (`gnus-group-sort-groups-by-method`)。

G S n グループバッファーを本当の (前に何も付いていない) グループ名でアルファベット順に並べ替えます (`gnus-group-sort-groups-by-real-name`)。

以下のすべての命令はプロセス/接頭引数の習慣に従います (see Section 8.1 [Process/Prefix], page 249)。

シンボル接頭引数 (see Section 8.3 [Symbolic Prefixes], page 250) が与えられたときは、これらすべての命令は逆順で並び換えます。

また、グループの一部を並べ替えることもできます。

G P a グループをグループ名のアルファベット順で並べ替えます (`gnus-group-sort-selected-groups-by-alphabet`)。

G P u グループを未読記事の数で並べ替えます (`gnus-group-sort-selected-groups-by-unread`)。

G P l グループをグループレベルで並べ替えます (`gnus-group-sort-selected-groups-by-level`)。

G P v グループをグループのスコアで並べ替えます (`gnus-group-sort-selected-groups-by-score`)。See Section 2.7 [Group Score], page 20.

- G P r* グループをグループのランクで並べ替えます (gnus-group-sort-selected-groups-by-rank)。See Section 2.7 [Group Score], page 20.
- G P m* グループをバックエンドの名前でアルファベット順に並べ替えます (gnus-group-sort-selected-groups-by-method)。
- G P n* グループを本当の (前に何も付いていない) グループ名でアルファベット順に並べ替えます (gnus-group-sort-selected-groups-by-real-name)。
- G P s* グループを gnus-group-sort-function に従って並べ替えます。

最後に、*C-k* と *C-y* を使って、手動でグループをあちこちに移動できることもお忘れなく。

2.13 グループの管理

- b* 不正なグループを見つけて、削除します (gnus-group-check-bogus-groups)。
- F* 新しいグループを見つけて、それら进行处理します (gnus-group-find-new-groups)。一回の *C-u* の後で押されると、サーバーに新しいグループを尋ねるために ask-server の方法を使います。二回の *C-u* の後で押されると、サーバーに新しいグループを尋ねるために最も完全であると思われる方法を用い、新しいグループをゾンビとして購読します。
- C-c C-x* 現在のグループの期限切れ消去可能な記事に対して (もしあれば) すべて期限切れ消去の処理を行ないます (gnus-group-expire-articles)。これは、そのグループにしばらく存在していた期限切れ消去可能なすべての記事を消去するということです。 (see Section 6.3.9 [Expiring Mail], page 163)。
- C-c C-M-x* すべてのグループのすべての期限切れ消去可能な記事に対して、期限切れ消去の処理を行ないます。 (gnus-group-expire-all-groups)。

2.14 外部サーバーの閲覧

- B* 選択方法とサーバー名を聞かれます。Gnus はこのサーバーに接続し、そこにあるグループを閲覧しようとします (gnus-group-browse-foreign-server)。

利用可能なグループのリストを持った新しいバッファが現れます。このバッファは gnus-browse-mode を使用します。このバッファは通常のグループバッファにちょっと (というか、とっても) 似ています。

以下が閲覧モード (browse mode) で使用できるキー操作のリストです:

- n* 次のグループに移動します (gnus-group-next-group)。
- p* 一つ前のグループに移動します (gnus-group-prev-group)。
- SPACE* 現在のグループに入り、最初の記事を表示します (gnus-browse-read-group)。
- RET* 現在のグループに入ります (gnus-browse-select-group)。
- u* 現在のグループを非購読にします。と言うよりは (訳注: このコマンドはトグルなので)、この場合は購読することになるのでしょうか (gnus-browse-unsubscribe-current-group)。
- l*
- q* 閲覧モード (browse mode) を終了します (gnus-browse-exit)。

- d* 現在のグループを購読にします (`gnus-browse-describe-group`)。
- ?* 閲覧モード (`browse mode`) を簡単に説明します (まあ、大して説明することもないんですけどね) (`gnus-browse-describe-briefly`)。

2.15 Gnus の終了

そう、Gnus は最後 (サイコー) です (訳注: く、苦しい。原文は“ Yes, Gnus is ex(c)iting. ”)。

- z* Gnus を中断します (`gnus-group-suspend`)。これは Gnus を実際には終了させず、グループバッファ以外のすべてのバッファを消すだけです。僕はこれのうれしさがよくわかんないんだけど、誰か分かる人います?
- q* Gnus を終了します (`gnus-group-exit`)。
- Q* ‘.newsrsrc’ ファイルを保存せずに Gnus を終了します (`gnus-group-quit`)。ドリブルファイルは保存されますけれど (see Section 1.8 [Auto Save], page 9)。

Gnus を中断するときは `gnus-suspend-gnus-hook` が呼び出されます。Gnus を終了するときには `gnus-exit-gnus-hook` が呼び出され、さらに Gnus を終了するときの最後として `gnus-after-exiting-gnus-hook` が呼び出されます。

Note:

ミス Lisa Cannifax は英語の授業中、後ろに座っている少年が彼女のプラスチックの椅子の背越しに、鉛筆で繰り返し線を描くのにつられて、足がしびれて重くなり、意識が朦朧としてきました。

2.16 Group Topics

もしあなたがたーくさんのグループを読んでいるのであれば、グループをトピック毎に階層分けできると便利でしょう。Emacs のグループをこっちへ、セックスのグループをあっちへ、で、残りを (え? グループが二つくらいしかないの?) 邪魔にならないようにその他のセクションに入れましょう。あるいは Emacs セックスのグループを Emacs グループ、セックスグループのどちらかの副トピックとすることさえもできます—あるいは両方に! すんごいでしょう!

これが例です:

```
Gnus
  Emacs -- こいつはすげーぜ!
    3: comp.emacs
    2: alt.religion.emacs
    えっちな Emacs
    452: alt.sex.emacs
    0: comp.talk.emacs.recovery
  その他
    8: comp.binaries.fractals
    13: comp.sources.unix
```

この素晴らしい機能を使うには、`gnus-topic` マイナーモードを (何と!) 単にスイッチオンするだけ—グループバッファで、`t` を押して下さい (これはトグルコマンドです)。

さあやってみよう。とにかく試してみて。君が戻ってくるまで、僕はここで待ってるからさ。ララ、タララン...、いい曲だね、これ...ラ、ラ、ラ...え? 戻ってきた? よし、じゃ次は `l` を押してみて。ほら。これですべてのグループが ‘misc’ の下に表示されました。興奮してクラクラしてこない? アツくって、いまいましいくらいでしょ?

これをずっと有効にしたければ、グループモードのフックにこのマイナーモードを追加して下さい。以下の行を ‘~/gnus.el’ ファイルに入れて、ね。

```
(add-hook 'gnus-group-mode-hook 'gnus-topic-mode)
```

2.16.1 トピック命令

トピックマイナーモードが有効であるときは、*T* サブマップが新しく利用できるようになります。さらに標準キーの中でも、定義がちょっと変わるものが少しあります。

だいたいにおいて、次のようなトピックの操作が可能です。まず第一に、あなたはトピックを作ること望むでしょう。第二に、あなたはグループをトピックに入れて、それらをあなたの好みの順序になるまで、あちこちに移動することを望むでしょう。第三に行なう操作は、それらの一切合切を表示したり隠すことでしょう。他のグループの概要を見やすくするために、あなたは副トピックやグループによっては、トピックが隠れるようにする必要があるかもしれませんね。

ここには、あなたの好むやり方でトピックを設定するために必要になりそうな、基本的なキーのリストがあります。

<i>T n</i>	新しいトピック名の入力を促し、それを作成します (gnus-topic-create-topic)。
<i>T TAB</i>	
<i>TAB</i>	現在のトピックの「字下げ」を行ない、その前のトピックの副トピックにします (gnus-topic-indent)。接頭引数を与えると、反対にそのトピックの字下げを回復 (un-indent) します。
<i>M-TAB</i>	現在のトピックの「字下げ回復」(un-indent) を行ない、それが現在の親の親の副トピックになるようにします (gnus-topic-unindent)。

以下の二つのキーは、グループとトピックをあちこち移動するために使われます。それらは、よく知られているカット＆ペーストのように動作します。*C-k* はカット、*C-y* はペーストです。もちろん、Emacs ではカット＆ペーストではなくて kill & yank という用語を使いますが。

<i>C-k</i>	グループあるいはトピックを kill します (gnus-topic-kill-group)。トピック内にあったグループもすべて、トピックと一緒に削除されます。
<i>C-y</i>	直前の kill されたグループあるいはトピックを yank します (gnus-topic-yank-group)。すべてのトピックは、すべてのグループの前に yank されることに気を付けて下さい。

ですから、あるトピックをトピックのリストの先頭に移動するには、単にそこで *C-k* を叩きます。これはカット＆ペーストのカットに相当します。そうしたらカーソルをバッファの先頭 (“ Gnus ”トピックの真下) に移動して、*C-y* を叩いて下さい。これはカット＆ペーストのペーストに相当します。なあんだ、簡単じゃん。

C-k と *C-y* はトピックと同様にグループにも使えます。すなわち、あなたはグループと同じようにトピックの移動もできるのです。

あなたの望みのままにトピックを使えるようにした後で、あなたはトピックを隠したり再び見えるようにしようと思うでしょう。そのために以下のキーを用意しています。

<i>RET</i>	
<i>SPACE</i>	グループを選択するか、あるいはトピックを折りたたみます (gnus-topic-select-group)。グループの上でこのコマンドを実行すると、通常通りそのグループに入ります。トピック行の上で行なうと、そのトピックは (すでに表示されているときは) 折りたたまれるか、(すでに折りたたまれているときは) 展開されます。つまりトピックに対し

てはこれはトグルコマンドです。さらに、数値の接頭引数を与えると、そのレベル（とそれよりも小さいレベル）のグループが表示されます。

さてお次は、他のコマンドのリストです。順序には特に意味はありません。

<i>T m</i>	現在のグループを、どこか他のトピックに移動させます (gnus-topic-move-group)。このコマンドはプロセス印/接頭引数の習慣に従います (see Section 8.1 [Process/Prefix], page 249)。
<i>T j</i>	トピックにジャンプします (gnus-topic-jump-to-topic)。
<i>T c</i>	現在のグループを、どこか他のトピックにコピーします (gnus-topic-copy-group)。このコマンドはプロセス印/接頭引数の習慣に従います (see Section 8.1 [Process/Prefix], page 249)。
<i>T h</i>	現在のトピックを隠します。接頭引数を与えらると、そのトピックを永久に隠します。
<i>T s</i>	現在のトピックを表示します。接頭引数を与えられると、そのトピックを永久に表示します。
<i>T D</i>	グループを現在のトピックから削除します (gnus-topic-remove-group)。この命令は主にいくつかのトピックに同じグループがあって、それをトピックの一つから取り除きたいときに役立ちます。あなたはグループをすべてのトピックから取り除くかもしれませんが、その場合は、Gnus はあなたが次回に Gnus を起動したときにそれをルートトピックに付け加えます。実際のところ、すべての新しいグループ（もちろん、それはどのトピックにも属していません）はルートトピックに現われます。 この命令はプロセス印/接頭引数の習慣に従います (see Section 8.1 [Process/Prefix], page 249)。
<i>T M</i>	正規表現に合致するすべてのグループを、あるトピックに移動させます (gnus-topic-move-matching)。
<i>T C</i>	正規表現に合致するすべてのグループを、あるトピックにコピーします (gnus-topic-copy-matching)。
<i>T H</i>	空のトピックの表示・非表示を切り替えます (gnus-topic-toggle-display-empty-topics)。
<i>T #</i>	現在のトピックにあるグループすべてにプロセス印を付けます (gnus-topic-mark-topic)。接頭引数を与えられない場合、このコマンドは副トピックに対して再帰的に働きます。
<i>T M-#</i>	現在のトピックにあるすべてのグループからプロセス印を消します (gnus-topic-unmark-topic)。接頭引数を与えられない場合、このコマンドは副トピックに対して再帰的に働きます。
<i>C-c C-x</i>	(もしあれば) 現在のグループかトピックかのすべての期限切れ消去可能記事を期限切れ消去します (gnus-topic-expire-articles)。 (see Section 6.3.9 [Expiring Mail], page 163)。
<i>T r</i>	トピックの名前を変更します (gnus-topic-rename)。
<i>T DEL</i>	空のトピックを削除します (gnus-topic-delete)。
<i>A T</i>	Gnus が知っているトピック化されたグループをすべて表示します (gnus-topic-list-active)。

- T M-n* 次のトピックに移動します (`gnus-topic-goto-next-topic`)。
- T M-p* 前のトピックに移動します (`gnus-topic-goto-previous-topic`)。
- G p* トピックパラメーターを修正します (`gnus-topic-edit-parameters`)。See Section 2.16.5 [Topic Parameters], page 37.

2.16.2 トピック変数

前の章では、どのトピックを表示するかを Gnus に言う方法を説明しました。この章では、それぞれのトピックの何を表示するかを Gnus に言う方法を説明します。

トピック行それ自体は、`gnus-topic-line-format` 変数の値に従って作成されます (see Section 8.4 [Formatting Variables], page 250)。有効な要素は、

- ‘i’ 字下げ。
- ‘n’ トピック名。
- ‘v’ 見えるかどうか。
- ‘l’ レベル。
- ‘g’ トピック中のグループの数。
- ‘a’ トピック中の未読記事の数。
- ‘A’ トピックとすべての副トピックの未読記事の数。

各副トピック (と副トピック内のグループ) は、トピックレベル数の `gnus-topic-indent-level` 倍の空白分の字下げが行なわれます。デフォルトは 2 です。

`gnus-topic-mode-hook` はトピックマイナーモードバッファーで呼び出されます。

`gnus-topic-display-empty-topics` はトピックの中に未読記事が無い場合でもそのトピックを表示するようにします。デフォルトは `t` です。

2.16.3 トピックの並べ替え

以下に示す命令で、各トピック毎に別々にグループを並べ替えることができます:

- T S a* 現在のトピックをグループ名のアルファベット順に並べ替えます (`gnus-topic-sort-groups-by-alphabet`)。
- T S u* 現在のトピックを未読記事の数で並べ替えます (`gnus-topic-sort-groups-by-unread`)。
- T S l* 現在のトピックをグループのレベルで並べ替えます (`gnus-topic-sort-groups-by-level`)。
- T S v* 現在のトピックをグループのスコアで並べ替えます (`gnus-topic-sort-groups-by-score`)。See Section 2.7 [Group Score], page 20.
- T S r* 現在のトピックをグループのランクで並べ替えます (`gnus-topic-sort-groups-by-rank`)。See Section 2.7 [Group Score], page 20.
- T S m* 現在のトピックをバックエンドの名前でアルファベット順に並べ替えます (`gnus-topic-sort-groups-by-method`)。
- T S e* 現在のトピックをサーバーの名前でアルファベット順に並べ替えます (`gnus-topic-sort-groups-by-server`)。

T S s 現在のトピックを、変数 `gnus-group-sort-function` で与えられる関数に従って並べ替えます (`gnus-topic-sort-groups`)。

接頭引数が与えられると、これらすべてのコマンドは逆順の並べ替えを行ないます。グループの並べ替えについてのさらなる情報は Section 2.12 [Sorting Groups], page 30 を参照して下さい。

2.16.4 トピックの位相構造

それでは、グループバッファの例を見ていきましょう。

```
Gnus
  Emacs -- こいつはすげーぜ!
    3: comp.emacs
    2: alt.religion.emacs
  えっちな Emacs
    452: alt.sex.emacs
    0: comp.talk.emacs.recovery
  その他
    8: comp.binaries.fractals
    13: comp.sources.unix
```

つまり、ここでは一つのトップレベルのトピック ('Gnus') があり、その下に二つのトピックがあり、そのうちの一方の副トピック中に一つ副トピックがあります (トップレベルトピックは常に一つしかありません)。この構造は、以下のように表現できます:

```
((("Gnus" visible)
  ((("Emacs -- こいつはすげーぜ!" visible)
    ((("えっちな Emacs" visible))))
  ((("その他" visible)))))
```

これは実に、上記の表示を行なうための、変数 `gnus-topic-topology` の値そのもののなのです。この変数は '`.newsrc.eld`' ファイルに保存され、手でいじくり回してはいけません—本当にやりたいときは別ですが。この変数は '`.newsrc.eld`' ファイルから読み込まれるので、その他のスタートアップファイルの設定にはまったく影響を与えません。

この構造は、どのトピックがどのトピックの副トピックであるかと、どのトピックが表示されているかを示しています。現在は二つの設定値—`visible` と `invisible` を使うことができます。

2.16.5 トピックパラメーター

トピック内のすべてのグループはグループパラメーターを、その親 (と先祖) のトピックパラメーターから継承します。グループパラメーターとして正しいものはすべて、トピックパラメーターとしても正しいものです (see Section 2.10 [Group Parameters], page 23)。エージェントを使うようにしてあると、すべてのエージェントパラメーター (Section 6.9.2.1 [Category Syntax], page 212 の Agent Parameters を参照 (訳注: 必要なら Index を使って)) は有効なトピックパラメーターでもあります。

さらに、以下のパラメーターはトピックパラメーターとしてのみ有効です:

`subscribe`

トピックで新しいグループを購読している場合 (see Section 1.5.2 [Subscription Methods], page 6)、`subscribe` トピックパラメーターはどのグループがどのトピックに行くかを指定します。値はそのトピックに行くグループに合致する正規表現である必要があります。

subscribe-level

トピックで新しいグループを購読している場合 (subscribe パラメーターを参照)、そのグループの購読度のレベルは `gnus-level-default-subscribed` の代わりに `subscribe-level` トピックパラメーターの値になります。

グループパラメーターは (もちろん) トピックパラメーターよりも優先され、副トピックのトピックパラメーターは親トピックのトピックパラメーターよりも優先されます。分かるよね。ごく普通の継承ルールです (ルール (*Rules*) はここでは名詞であって、動詞の「線を引く」ではありません。このルールには反対したくなるかもしれないけど、それはご自由に)。

```
Gnus
  Emacs
    3: comp.emacs
    2: alt.religion.emacs
  452: alt.sex.emacs
  息抜き
    452: alt.sex.emacs
    0: comp.talk.emacs.recovery
  その他
    8: comp.binaries.fractals
    13: comp.sources.unix
    452: alt.sex.emacs
```

‘Emacs’ トピックはトピックパラメーター (`score-file . "emacs.SCORE"`) を持っています。‘息抜き’ トピックはトピックパラメーター (`score-file . "relief.SCORE"`) を持ち、‘その他’ トピックはトピックパラメーター (`score-file . "emacs.SCORE"`) を持っています。さらに、‘alt.religion.emacs’ はグループパラメーター (`score-file . "religion.SCORE"`) を持っています。

さて、ここで ‘息抜き’ トピックの ‘alt.sex.emacs’ グループに入ったとき、‘relief.SCORE’ が基本スコアファイルとなります。もし ‘Emacs’ トピックの同じグループに入ると、‘emacs.SCORE’ が基本スコアファイルになるでしょう。‘alt.religion.emacs’ グループに入れば、‘religion.SCORE’ が基本スコアファイルになるでしょう。

これってとっても簡単で自明のこのように見えるでしょ？ まあ、その通りです。ですが問題がある場合もあります。特に `total-expiry` パラメーターに関してです。例えばあるメールグループを二つのトピックの中に、一方は `total-expiry` ありで、もう一方はそれなしで持っているとしましょう。ここで `M-x gnus-expire-all-expirable-groups` を実行すると、何が起こるでしょうか？ Gnus は、あなたがどちらのトピックから記事を期限切れ消去したいのかを知る方法がないため、最悪の事態が発生するかもしれません。実際、私はこのとき何が起こるのかは「未定義 *undefined*」である、とここに宣言します。この手のことをやりたい場合には十分注意しなければなりません。

2.17 その他のグループ関連

v `v` キーはユーザー用に予約されています。そのまま何かの機能に割り当てても構いませんが、接頭キーとして使う方が良いでしょう。例です:

```
(define-key gnus-group-mode-map (kbd "v j d")
  (lambda ()
    (interactive)
    (gnus-group-jump-to-group "nndraft:drafts")))
```


Emacs でユーザー用に予約されているキーとキーバインドについては、See section “Keymaps” in *The Emacs Editor*.

- ˆ サーバーバッファモードに入ります (`gnus-group-enter-server-mode`)。See Section 6.1 [Server Buffer], page 133.
- a メッセージ (デフォルトはニュース) の作成を開始します (`gnus-group-post-news`)。接頭引数が与えられると、現在位置のグループに投稿します。もし接頭引数が 1 だったら、どのグループに投稿するかを尋ねます。この関数の名前から連想されることとは裏腹に、接頭引数でメールグループが指定された場合は、ニュースの代わりにメールの様式が用意されます。See Chapter 5 [Composing Messages], page 123.
- m メールをどこかに送ります (`gnus-group-mail`)。接頭引数が与えられると、現在位置のグループの投稿様式 (posting style) を使います。もし接頭引数が 1 だったら、どのグループの投稿様式を使うかを尋ねます。See Chapter 5 [Composing Messages], page 123.
- i ニュースの作成を開始します (`gnus-group-news`)。接頭引数が与えられると、現在位置のグループに投稿します。もし接頭引数が 1 だったら、どのグループに投稿するかを尋ねます。See Chapter 5 [Composing Messages], page 123.
この関数は、たとえメールグループで使われたとしても、実際にはニュースの様式を用意します。これは、メッセージを実際にはネットワーク経由で送らずに、メールグループに「投稿」するのに便利です; それらは当のグループに単に直接保存されます。対応するバックエンドが投稿のためのメソッド (`request-post method`) を持っていないかもしれません。

G z

現在位置のグループを圧縮します (`gnus-group-compact-group`)。今のところ `nnml` (see Section 6.3.13.3 [Mail Spool], page 169) だけに実装されています。これは記事番号のすきまを取り除くので、正しい全記事数を得ることができるようになります。

以下はグループバッファのための変数です:

`gnus-group-mode-hook`

グループバッファが作成された時に呼び出されます。

`gnus-group-prepare-hook`

グループバッファが生成されたあとに呼び出されます。これはバッファを何か変な、自然ではない方法で修正したいときに使われるかもしれません。

`gnus-group-prepared-hook`

グループバッファが生成された後の一番最後に呼び出されます。例えばポイントをどこかに移動させたいときなどに使えます。

`gnus-permanently-visible-groups`

この正規表現に合致するグループは、それが空であるかどうかに関わらず、常にグループバッファに表示されます。

`gnus-group-name-charset-method-alist`

グループ名用の選択方法と文字セットの連想リストです。これは英字ではないグループ名を表示するために使います。

例:

```
(setq gnus-group-name-charset-method-alist
      '(((nntp "news.com.cn") . cn-gb-2312)))
```

`gnus-group-name-charset-group-alist`

グループ名用のグループ名の正規表現と文字セットの連想リストです。これは英字ではないグループ名を表示するために使います。UTF-8 がサポートされている場合は `((".*" utf-8))` がデフォルト値で、それ以外の場合のデフォルトは `nil` です。

例:

```
(setq gnus-group-name-charset-group-alist
      '(("\\.com\\.cn:" . cn-gb-2312)))
```

2.17.1 新着メッセージを探す

- g* サーバーの新着記事をチェックします。数値の接頭指数を使用すると、この命令は引数 *arg* かそれより小さいレベルのグループのみをチェックします (`gnus-group-get-new-news`)。数値以外の接頭指数を与えると、この命令はそのバックエンドからアクティブファイルを強制的に全部読み直します。
- M-g* 現在のグループに新着記事があるかどうかをチェックします (`gnus-group-get-new-news-this-group`)。 `gnus-goto-next-group-when-activating` はこの命令が次のグループ位置へ移動するかどうかを決めます。デフォルトは `t` です。
- C-c M-g* 無条件にすべてのグループを起動します (`gnus-activate-all-groups`)。
- R* Gnus を再起動します (`gnus-group-restart`)。これは `‘.newsrc’` ファイルをセーブし、すべてのサーバーの接続を閉じ、すべての Gnus ランタイム変数をクリアした後、Gnus をもう一度最初から開始します。

`gnus-get-new-news-hook` は新着ニュースをチェックする直前に実行されます。

`gnus-after-getting-new-news-hook` 新着ニュースをチェックした後に実行されます。

2.17.2 グループ情報

- H f* 現在のグループの FAQ を取得しようとします (`gnus-group-fetch-faq`)。Gnus は FAQ を `gnus-group-faq-directory` から取得しようとします。これは通常リモートマシン上のディレクトリーです。この変数はディレクトリーのリストであっても構いません。この場合、このコマンドに接頭指数を与えることでいくつかのサイトの中から選ぶことができます。ファイルの取得には `ange-ftp` (または `efs`) が使用されます。もし最初のサイトからの取得が失敗した場合、Gnus は `gnus-group-faq-directory` の値をすべて、一つ一つオープンしてみようとします。
- H c* 現在のグループの憲章を web ブラウザーで開こうとします (`gnus-group-fetch-charter`)。接頭指数が与えられるとグループ名を尋ねます。
Gnus は `gnus-group-charter-alist` を使って憲章の所在を探します。所在がわからなかったら、Gnus はそのグループのコントロールメッセージを取得します。それには憲章を含んでいることがありますから。
- H C* そのグループのコントロールメッセージを `ftp.isc.org` の記録庫から取得します (`gnus-group-fetch-control`)。接頭指数が与えられるとグループ名を尋ねます。
`gnus-group-fetch-control-use-browse-url` が非-`nil` だったら、Gnus は `browse-url` を使ってコントロールメッセージを開きます。そうでない場合は `ange-ftp` を使って取得され、一時グループに表示されます。

コントロールメッセージは圧縮されていることに注意して下さい。このコマンドを使うには `auto-compression-mode` を `on` にしておく必要があります (see section “Compressed Files” in *The Emacs Editor*).

`H d`

`C-c C-d` 現在のグループの説明を表示します (`gnus-group-describe-group`)。接頭引数を与えると、説明文をサーバーから強制的に再読み込みします。

`M-d` すべてのグループの説明を表示します (`gnus-group-describe-all-groups`)。接頭引数を与えると、説明文ファイルをサーバーから強制的に再読み込みします。

`H v`

`V` 現在の Gnus のバージョン番号を表示します (`gnus-version`)。

`?`

とても短いヘルプメッセージを与えます (`gnus-group-describe-briefly`)。

`C-c C-i`

Gnus の `info` ノードに移動します (`gnus-info-find-node`)。

2.17.3 グループの日付

Gnus に、あなたが最後にいつグループを読んだかを記録させると便利かもしれません。この活動を始めさせるには、`gnus-group-set-timestamp` を `gnus-select-group-hook` に追加して下さい。

```
(add-hook 'gnus-select-group-hook 'gnus-group-set-timestamp)
```

これを行なった後、あなたがグループに入るたびにそれが記録されます。

この情報はさまざまな方法で表示できます—もっとも簡単なのは、グループ行フォーマットで ‘`%d`’ 指定を使う方法です:

```
(setq gnus-group-line-format
      "%M%S%p\P%5y: %(%-40,40g%) %d\n")
```

この結果として、各行は以下のように表示されます:

```
*      0: mail.ding                      19961002T012943
      0: custom                          19961002T012713
```

見て分かるとおり、日付はコンパクトな ISO 8601 形式で表示されます。これではちょっとあんまりなので、以下のような感じにすると日付だけを表示できます。

```
(setq gnus-group-line-format
      "%M%S%p\P%5y: %(%-40,40g%) %6,6~(cut 2)d\n")
```

もっと凝った日付の形式をお望みなら、利用者定義によるフォーマットの仕様を使うことができます。以下のようなものはうまくいくでしょう:

```
(setq gnus-group-line-format
      "%M%S%p\P%5y: %(%-40,40g%) %ud\n")
(defun gnus-user-format-function-d (headers)
  (let ((time (gnus-group-timestamp gnus-tmp-group)))
    (if time
        (format-time-string "%b %d %H:%M" time)
        "")))
```

2.17.4 ファイル命令

- r** 初期化ファイルの再読み込みを行ないます (gnus-init-file、デフォルトは '~/.gnus.el') (gnus-group-read-init-file)。
- s** '.newsrsrc.elc' ファイル (と、もしそうしなければ '.newsrsrc' ファイル) をセーブします (gnus-group-save-newsrsrc)。

2.17.5 Sieve コマンド

Sieve はサーバー側で使われるメールフィルター言語です。Gnus では、各グループに適用される sieve の規則を指定する、sieve グループパラメーター (see Section 2.10 [Group Parameters], page 23) を使うことができます。gnus はそれらすべてのグループパラメーターを、サーバーで使うことも可能な正しい Sieve スクリプトに翻訳する、二つのコマンドを提供します。

作成された Sieve スクリプトは gnus-sieve-file (デフォルトは '~/.sieve') に置かれます。Gnus が作るコードは二つの区切り記号 gnus-sieve-region-start と gnus-sieve-region-end の間に置かれるので、これらの区切り記号の外に追加の Sieve コードを書いても、次回 Sieve スクリプトを再作成するときに消されてしまうことはありません。

変数 gnus-sieve-crosspost は Sieve スクリプトがどのように作られるかを制御します。もし非-nil (デフォルト) だったら記事は規則に合致するすべてのグループに置かれます。そうでない場合、記事は最初の規則に合致するグループだけに置かれます。例えばグループパラメーター '(sieve address "sender" "owner-ding@hpc.uh.edu")' は、gnus-sieve-crosspost が nil だったら以下の Sieve コードの断片を作ります。(gnus-sieve-crosspost が非-nil だった場合は、行が含む stop の呼び出しが削除されること以外は同じです。)

```
if address "sender" "owner-ding@hpc.uh.edu" {
    fileinto "INBOX.ding";
    stop;
}
```

See section “Top” in *Emacs Sieve*.

- D g** sieve グループパラメーターから Sieve スクリプトを再作成して、gnus-sieve-file に書き込みます。以前の内容は保存されません。
- D u** sieve グループパラメーターを元に gnus-sieve-file の Gnus が管理している部分を再作成してファイルにセーブし、sieveshell プログラムを使ってサーバーにアップロードします。

3 概略バッファ

概略バッファ (summary buffer) ではそれぞれの記事が一行で表示されます。その中を動き回り、記事を読み、投稿し、返答をすることができます。

概略バッファに移る一番普通の方法は、グループバッファでグループを選択することです (see Section 2.3 [Selecting a Group], page 17)。

好きなだけたくさんの概略バッファを開いておくことができます。

概略モードのツールバーをカスタマイズすることができます。M-x *customize-apropos RET gnus-summary-tool-bar* を試してみてください。この機能を利用できるのは Emacs だけです。

v キーはユーザー用に予約されています。そのまま何かの機能に割り当てても構いませんが、接頭キーとして使う方が良いでしょう。例です:

```
;; 副スレッドのスコアを下げる。
(define-key gnus-summary-mode-map (kbd "v -") "LrS")
```

3.1 概略バッファの様式

Gnus は変数 `gnus-extract-address-components` の値を From ヘッダーの名前とアドレスの部分抽出するための関数として使います。すでに定義されている関数が二つ存在します: デフォルトは `gnus-extract-address-components` で、とても簡単に割り切った解決法ですが非常に速く動作します。mail-extract-address-components は良く動作しますが遅いです。デフォルトの関数は 5% の割合で間違った答を返します。もしこれに我慢ならないのであれば、代わりに他の関数を使って下さい:

```
(setq gnus-extract-address-components
      'mail-extract-address-components)
```

`gnus-summary-same-subject` は今読んでいる記事が、その前の記事と同じ表題 (subject) であることを示す文字列です。この文字列は、それを要求する書法仕様で使われます。デフォルトでは "" です。

3.1.1 概略バッファの行

変数 `gnus-summary-line-format` の値を変えることによって、概略バッファの行の様式 (format) を変更することができます。いくつかの拡張 (see Section 8.4 [Formatting Variables], page 250) とともに、普通の format 文字列と同じように動作します。

行には常にコロンのポイント位置のマーカーが存在していなければなりません。操作した後に、カーソルはいつもコロンのポイント位置のマーカーの場所に移動します。(もちろん、この動作を変えることができないとしたら Gnus にはあるまじきことです。関数 `gnus-goto-colon` を、あなたが好きなカーソルの動きになるように、新たに書けば良いのです。) See Section 8.4.6 [Positioning Point], page 253.

デフォルトの文字列は '%U%R%z%I%([%4L: %-23,23f%]) %s\n' です。

以下の様式指示文字と拡張様式指示を使うことができます:

- 'N' 記事数。
- 'S' 表題の文字列。gnus-list-identifiers の設定によってメーリングリストの標識が削除されます。See Section 3.17.3 [Article Hiding], page 83.

- ‘s’ スレッド (thread) の元記事であるときか直前の記事が違う表題のときはその表題で、それ以外は `gnus-summary-same-subject`。(`gnus-summary-same-subject` はデフォルトで `"`。)
- ‘F’ 完全な From 欄。
- ‘n’ 名前 (From 欄より)。
- ‘f’ 名前、To 欄の内容、または Newsgroups 欄の内容のどれかです (see Section 3.1.2 [To From Newsgroups], page 46)。
- ‘a’ 名前 (From 欄より)。これと n との違いは、これは変数 `gnus-extract-address-components` で指定されている関数を使って名前を取得することです。この方が遅いですが、おそらくより完全に近いでしょう。
- ‘A’ 名前 (From 欄より)。これは a と同じように動作します。
- ‘L’ 記事の行数。
- ‘c’ 記事の文字数。この名前指定子は (nnfolder のような) いくつかの選択方法をサポートしません。
- ‘k’ 整形された記事の文字数; 例えば ‘1.2k’ や ‘0.4M’。
- ‘I’ スレッドのレベルによる字下げ (see Section 3.9.1 [Customizing Threading], page 63)。
- ‘B’ 複雑な trn 様式のスレッド木 (tree)。どのような応答が行なわれたかの記録を表示します。スレッドはこのように描かれるでしょう:

```

>
+->
| +->
| | \->
| |   \->
| |     \->
| \->
+->
\->

```

以下のオプションで見栄えをカスタマイズすることができます。デフォルトの ASCII 文字を線描画用の図案で置き換えることによって、スレッド表示を実に巧妙に見せることができることに気付いて下さい。

`gnus-sum-thread-tree-root`

スレッドの根 (root) に使われます。nil だったら、代わりに表題を使います。デフォルトは ‘>’ です。

`gnus-sum-thread-tree-false-root`

スレッドのにせの根に使われます (see Section 3.9.1.1 [Loose Threads], page 63)。nil だったら、代わりに表題を使います。デフォルトは ‘>’ です。

`gnus-sum-thread-tree-single-indent`

単一のメッセージのスレッドに使われます。nil だったら、代わりに表題を使います。デフォルトは ‘ ’ です。

<code>gnus-sum-thread-tree-vertical</code>	縦線の描画に使われます。デフォルトは「 」です。
<code>gnus-sum-thread-tree-indent</code>	行下げ (indenting) に使われます。デフォルトは「」です。
<code>gnus-sum-thread-tree-leaf-with-other</code>	兄弟がいる葉っぱに使われます。デフォルトは「+>」です。
<code>gnus-sum-thread-tree-single-leaf</code>	兄弟がない葉っぱに使われます。デフォルトは「\->」です。
‘T’	記事が元記事であれば何も表示せず、そうでない場合はたくさんの空白です (それより後のものをすべて画面の外に追い出してしまいます)。
‘[’	開き括弧。普通は「[」ですが、養子記事には「<」にすることができます (see Section 3.9.1 [Customizing Threading], page 63)。
‘]’	閉じ括弧。普通は「]」ですが、養子記事には「<」にすることができます。
‘>’	それぞれのスレッドのレベルに対して一つの空白。
‘<’	(20 - スレッドレベル) 個の空白。
‘U’	未読。See Section 3.7.2 [Read Articles], page 57.
‘R’	この紛らわしい名前指定子は「第二の印」(the secondary mark) です。この印は記事がすでに返答済みのものか、キャッシュされたものか、あるいは保存されたものかを表します。See Section 3.7.3 [Other Marks], page 57.
‘i’	数値としてのスコア (see Chapter 7 [Scoring], page 227)。
‘z’	これは、zcore でデフォルトのレベルよりも上であれば「+」で、デフォルトのレベルよりも下であれば「-」です。gnus-summary-default-score との差が gnus-summary-zcore-fuzz よりも小さいと、この仕様は使われません。
‘V’	スレッド全体のスコア。
‘x’	Xref.
‘D’	Date.
‘d’	DD-MM 様式による Date。
‘o’	YYYYMMDDTHHMMSS 様式による Date。
‘M’	Message-ID.
‘r’	References.
‘t’	現在の副スレッドの記事の数。この仕様を使うと概略バッファの生成が幾分遅くなります。
‘e’	記事に子記事があると、「=」(gnus-not-empty-thread-mark) が表示されます。
‘P’	行数。
‘O’	ダウンロードの印。
‘*’	カーソルを (最初のコロンの後ろの代わりに) 置く場所。

`'&user-date;'`

経過時間の様式。いろいろな様式が `gnus-user-date-format-alist` で定義されています。

`'u'`

利用者定義指定子。フォーマット文字列の中の次の文字は英字でなければなりません。これにより Gnus は関数 `gnus-user-format-function-x` を呼び出しますが、ここで `x` は `'u'` の次の文字です。関数には現在の記事のヘッダーが引数として渡されます。関数は文字列を返さなければなりません。それは他の概略指定と同様に概略に挿入されます。

`'%('` と `'%)'` の間にあるテキストは、そこにマウスがあるときに `gnus-mouse-face` でハイライトされます。そういう領域は一つだけです。

`'%U'` (状態), `'%R'` (返答済み), `'%Z'` (zcore) の扱いには気を付ける必要があります。効率のために、Gnus はこれらの文字がどの桁に現れるかを計算し、『ハード・コード』します。これは、可変長の仕様の後では、これらは意味を持たないということです。まあ、さすがに逮捕はされないでしょうが、概略バッファは変になります。それでも十分悲しいでしょうけど。

賢い選択はこれらの指定をできるだけ左に持ってくることです。(でも、そういうことはすべてに当てはまるのではないのでしょうか。閑話休題。)

この制限は将来の版では無くなるかもしれません。

3.1.2 To From Newsgroups

いくつかのグループ (特にアーカイブグループ) では From ヘッダーはあまり興味を引きません。そのすべての記事はあなたによって書かれたものですから。代わりに、To や Newsgroups ヘッダーの情報を表示するためには、三つのことを決める必要があります: どの情報を集めるか、どこに表示するか、いつ表示するか。

1. 追加のヘッダーの情報は `gnus-extra-headers` により制御されます。これはヘッダーのシンボルのリストです。例えば:

```
(setq gnus-extra-headers
      '(To Newsgroups X-Newsreader))
```

これによって Gnus はこれらの三つのヘッダーを取得しようとし、後の容易な取得のためにヘッダー構造に保存します。

2. これらの追加のヘッダーの値は `gnus-extra-function` 関数を通じて取得することができます。これは X-Newsreader ヘッダーを使う書式行の仕様です:

```
"%~(form (gnus-extra-header 'X-Newsreader))@"
```

3. `gnus-ignored-from-addresses` 変数はいつ `'%f'` 概略行仕様が To, Newsreader や From ヘッダーを返せば良いかを決めます。この正規表現が From ヘッダーの内容と合致すると、To や Newsreader ヘッダーの値が代わりに使用されます。

それらの From フィールドが入れ替わっている記事と、普通の記事を区別するために、概略行の To または Newsgroups ヘッダーに、ある文字列が前置されます。その文字列はデフォルトで、To には `'->'` が、Newsgroups には `'=>'` が使われますが、`gnus-summary-to-prefix` と `gnus-summary-newsgroup-prefix` によって、それらの文字列をカスタマイズすることができます。

関連する変数は `nnmail-extra-headers` で、`overview` (NOV) ファイルを作る際にいつ追加のヘッダーを含めるかを制御します。古い `overview` ファイルがある場合は、この変数を変更した後にサーバーバッファに `^` で入って適切なメールサーバー (例えば `nnml`) で `g` を押し、再生成する必要があります。

さらに `gnus-summary-line-format` 変数の `%n` 仕様を `%f` 仕様に変更することによってデータを表示するように、Gnus に指示する必要があります。

要約すると、普通は以下のようなものを `~/gnus.el` に置くことになります:

```
(setq gnus-extra-headers
      '(To Newsgroups))
(setq nnmail-extra-headers gnus-extra-headers)
(setq gnus-summary-line-format
      "%U%R%z%I%(<[%4L: %-23,23f%]) %s\n")
(setq gnus-ignored-from-addresses
      "Your Name Here")
```

(上記の値は Gnus のデフォルト値です。あなたの役に立つように変えて下さい。)

ニュース管理人、またはニュース管理人を説得してサポートの追加をしてもらおうと思っている利用者のみなさんへのご注意:

NOV ファイルの生成を制御できるメールグループでは、上記のことはたいていの場合役立ちます。しかし、管理人を説得して (特に INN の普通の実装において) 以下のものを `overview.fmt` ファイルの最後に追加してもうらうことができれば、メールグループでの追加ヘッダーのようにそれを使うことができます。

```
Newsgroups:full
```

3.1.3 概略バッファのモード行

概略のモード行の様式も変更することができます (see Section 8.4.2 [Mode Line Formatting], page 251)。`gnus-summary-mode-line-format` を何でも好きなものに設定して下さい。デフォルトは `'Gnus: %%b [%A] %Z'` です。

以下はあなたが遊ぶことのできる要素たちです:

- 'G' グループ名。
- 'p' 接頭語を取り除いた名前。
- 'A' 現在の記事番号。
- 'z' 現在の記事スコア。
- 'V' Gnus バージョン。
- 'U' そのグループでの未読記事の数。
- 'e' 概略バッファに表示されていない未読記事の数。
- 'Z' 未読と未選択の記事の数とともに表される文字列で、未読かつ未選択の記事がある場合の `'<%U(+%e) more>'`、および未読記事のみの場合の `'<%U more>'` のどちらかです。
- 'g' 短縮グループ名。例えば、`'rec.arts.anime'` は `'r.a.anime'` に短縮されます。
- 'S' 現在の記事の表題。
- 'u' 利用者定義の仕様 (see Section 8.4.4 [User-Defined Specs], page 252)。
- 's' 現在のスコアファイルの名前 (see Chapter 7 [Scoring], page 227)。
- 'd' 保留記事の数 (see Section 3.7.1 [Unread Articles], page 56)。
- 't' 可視印付き記事の数 (see Section 3.7.1 [Unread Articles], page 56)。

‘r’ その概略バッファで記事を読んだ結果、既読の印が付いた記事の数。

‘E’ スコアファイルによって抹消された記事の数。

3.1.4 概略のハイライト

`gnus-visual-mark-article-hook`

このフックは記事を選択した後に実行されます。これは何らかの方法で記事をハイライトするように意図されています。`gnus-visual` が `nil` だったら実行されません。

`gnus-summary-update-hook`

このフックは概略行が変化したときに呼ばれます。`gnus-visual` が `nil` だったら実行されません。

`gnus-summary-selected-face`

これは概略バッファでの現在の記事をハイライトするために使われるフェース (もしくは、ある人たちが「フォント」と呼ぶようなもの) です。

`gnus-summary-highlight`

概略行はこの変数にしたがってハイライトされます。この変数は要素が (*form* . *face*) の形式のリストです。例えば、印付きの記事を斜体、高いスコアの記事を太字にしたければ、この変数を次のように設定することができます。

```
((eq mark gnus-ticked-mark) . italic)
(> score default) . bold))
```

ご想像のとおり、*form* が `nil` でない値を返すと、*face* がその行に適用されます。

3.2 概略間の移動

すべての直接移動命令は数値接頭引数を受け付け、かなり期待どおりに動作するでしょう。

これらの命令はどれも記事を選択しません。

G M-n

M-n 概略行の次の未読記事に移ります (`gnus-summary-next-unread-subject`)。

G M-p

M-p 概略行の前の未読記事に移ります (`gnus-summary-prev-unread-subject`)。

G g

記事番号を尋ね、その記事を表示せずに、その概略行に行きます (`gnus-summary-goto-subject`)。

Gnus が次のグループ移動することを確認するためにキー入力を求めた場合、*C-n* キーと *C-p* キーを使うことによって、実際にグループバッファに戻らなくても、次に読むグループを探すことができます。

概略の移動に関連した変数:

`gnus-auto-select-next`

移動命令の一つ (*n* のような) を発したときに現在の記事の後に未読記事が無いと、Gnus は次のグループに移動することをうながします。この変数が `t` で次のグループが空っぽだったら、Gnus は概略モードを抜けてグループバッファに戻ります。この変数が `t` でも `nil` でもなければ、Gnus はさらに次の未読記事があるグループを選択します。特別な場合として、この変数が `quietly` だったら、Gnus は確認をせずに次のグループを選択します。この変数が `almost-quietly` だった場合は、グループの一番最後の記事

を読んでいたときに限って同じことが起こります。最後に、もしこの変数が `slightly-quietly` だったら、`Z n` 命令は確認をせずに次のグループに移ります。Section 2.6 [Group Levels], page 19 も参照して下さい。

`gnus-auto-select-same`

`nil` でないと、すべての移動命令は現在の記事と同じ表題の記事に移動しようとします。(「同じ」はここでは「大体同じ」という意味かもしれません。詳細は `gnus-summary-gather-subject-limit` を見て下さい (see Section 3.9.1 [Customizing Threading], page 63)。) 同じ表題の記事が無いときは、最初の未読記事に移動します。

この変数は、スレッド表示を行なっているときはあまり役に立たないでしょう。

`gnus-summary-check-current`

これが `nil` ではない場合、すべての『未読』移動命令は、現在の記事が未読だったら次(もしくは前)の記事に移動しません。代わりに、それらは現在の記事を選びます。

`gnus-auto-center-summary`

`nil` でないと、Gnus は概略バッファでのポイントを常に真中に保ちます。これをする、とてもこざれいになります、遅いネットワークに接続していたり、この Emacs らしくない流儀が好きになれないのであれば、この変数を `nil` にすることによって、普通の Emacs のスクロールにすることができます。これは概略バッファの水平方向でポイントが真ん中になるようにする操作 (horizontal re-centering) も禁止してしまうので、非常に長いスレッドを読むときは不便かもしれません。

この変数は数値でも構いません。その場合は、ウィンドウの先頭からその数の行だけ下がった位置に常にポイントがあるように制御されます。

3.3 記事の選択

3.3.1 選択命令

以下の移動コマンドはどれも数値接頭引数を受け付けません。それらはすべて、記事を選択して表示します。

新しい記事を取り込んだり、グループを再表示したいときは Section 3.27 [Exiting the Summary Buffer], page 108 を参照して下さい。

SPACE 現在の記事、またはそれが既読だった場合は次の未読記事を選択します (`gnus-summary-next-page`)。

すでに記事ウィンドウを開いているときに再び **SPACE** を押すと、その記事はスクロールされます。これによって、ニュースグループ全体を **SPACE** だけで便利に通読することができます。See Section 3.4 [Paging the Article], page 50.

G n

n 次の未読記事に移動します (`gnus-summary-next-unread-article`)。

G p

p 前の未読記事に移動します (`gnus-summary-prev-unread-article`)。

G N

N 次の記事に移動します (`gnus-summary-next-article`)。

G P

P 前の記事に移動します (`gnus-summary-prev-article`)。

- G C-n* 同じ表題の次の記事に移動します (`gnus-summary-next-same-subject`)。
- G C-p* 同じ表題の前の記事に移動します (`gnus-summary-prev-same-subject`)。
- G f*
.
最初の未読記事に移動します (`gnus-summary-first-unread-article`)。
- G b*
,
最高スコアの未読記事に移動します (`gnus-summary-best-unread-article`)。接頭引数が与えられると、デフォルトのスコアより大きいスコアを持つ最初の未読記事に移動します。
- G l*
l 直前に読んだ記事に移動します (`gnus-summary-goto-last-article`)。
- G o* 概略の履歴 (history) から最後の記事の一つ取り出して選択します (`gnus-summary-pop-article`)。この命令が上の命令と違うのは、*l* が最後の二つの記事の間を移動するだけなのに対して、これは好きなだけ前の記事を履歴から選び出すことができる点です。これに多少関係することについて、Section 3.14 [Article Backlog], page 73 を参照して下さい (これらの命令をたくさん使うのであれば)。
- G j*
j 記事番号か Message-ID を尋ね、それからその記事に行きます (`gnus-summary-goto-article`)。

3.3.2 選ぶための変数

記事の移動と選択に関連するいくつかの変数:

`gnus-auto-extended-newsgroup`

この変数が `nil` でないと、すべての移動命令は、記事が概略バッファに表示されていない場合でも、前 (もしくは次) の記事に移動しようとします。その際 Gnus はサーバーから記事を取得して、記事バッファに表示します。

`gnus-select-article-hook`

このフックは記事が選択されたときに常に呼ばれます。デフォルトは `nil` です。講読するそれぞれの記事をエージェントに保存させたい場合は、このフックに `gnus-agent-fetch-selected-article` を追加すれば良いでしょう。

`gnus-mark-article-hook`

このフックは記事が選択されたときに常に呼ばれます。これは記事に既読の印を付けるために使われることを意図しています。デフォルト値は `gnus-summary-mark-read-and-unread-as-read` で、ほとんどすべての読んだ記事の印を `gnus-read-mark` に変更します。この関数に影響されない記事は、可視、保留、期限切れ消去可能記事だけです。未読記事に既読の印を付けたいだけであれば、代わりに `gnus-summary-mark-unread-as-read` を使うことができます。`gnus-low-score-mark` や `gnus-del-mark` (など) の印はそのまま残します。

3.4 記事のスクロール

- SPACE* *SPACE* を押すと、現在の記事を一ページ先にスクロールします。記事の最後に行き着いた場合は次の記事を選択します (`gnus-summary-next-page`)。

gnus-article-skip-boring が非-nil で、かつ記事の残りに引用と署名しか無い場合、それはスキップされ、代わりに次の記事が表示されます。gnus-article-boring-faces で、つまらないと思うものをカスタマイズすることができます。どんなにうんざりするものでも、C-M-v を使うことによって、手動で記事のページを見ることはできます。

DEL 現在の記事を一ページ前にスクロールします (gnus-summary-prev-page)。

RET 現在の記事を一行先にスクロールします (gnus-summary-scroll-up)。

M-RET 現在の記事を一行後ろへスクロールします (gnus-summary-scroll-down)。

A g

g 現在の記事を (再) 取得します。もし接頭引数が与えられると、現在の記事を取得しますが、記事をトリートメントする関数は実行しません。これは、サーバーから来たままの『生の』記事をもたらします。

接頭引数を与えると、手動で文字セットの操作を行なうことができます。C-u 0 *g* cn-gb-2312 *RET* により、メッセージはあたかも cn-gb-2312 文字セットでエンコードされたかのようにデコードされます。以下のような設定を用意しておくと、C-u 1 *g* で同じ効果を得ることができます。

```
(setq gnus-summary-show-article-charset-alist
      '((1 . cn-gb-2312)
        (2 . big5)))
```

A <

< 記事の最初までスクロールします (gnus-summary-beginning-of-article)。

A >

> 記事の最後までスクロールします (gnus-summary-end-of-article)。

A s

s 記事バッファでインクリメンタルサーチ (isearch) を行ないます (gnus-summary-isearch-article)。

h

記事バッファを選択します (gnus-summary-select-article-buffer)。

3.5 返答、フォローアップ、投稿

3.5.1 概略でのメールの命令

メールメッセージを作成するための命令:

S r

r 現在の記事を書いた人に返答のメールを送ります (gnus-summary-reply)。

S R

R 現在の記事を書いた人に、元記事を含んだ返答のメールを出します (gnus-summary-reply-with-original)。この命令はプロセス/接頭引数の習慣を使います。

S w

現在の記事を書いた人に対して、広い返答 (wide reply) をします (gnus-summary-wide-reply)。「広い返答」とはヘッダーの To, From, (もしくは Reply-to) と Cc) のすべての人に返答をすることです。Mail-Followup-To があれば、代わりにそれが使われます。

- S W* 現在の記事に元記事を含んだ広い返答のメールを送ります (gnus-summary-wide-reply-with-original)。この命令はプロセス/接頭引数の習慣を使います。
- S v* 現在の記事を書いた人に対して、非常に広い返答 (very wide reply) をします (gnus-summary-very-wide-reply)。「非常に広い返答」とは、プロセス/接頭引数で指定されたすべての記事のヘッダーの To, From, (もしくは Reply-to) と Cc) のすべての人に返答をすることです。この命令はプロセス/接頭引数の習慣を使います。
- S V* 現在の記事に元記事を含んだ非常に広い返答のメールを送ります (gnus-summary-very-wide-reply-with-original)。この命令はプロセス/接頭引数の習慣を使います。
- S B r* 現在の記事を書いた人に対して返答をしますが Reply-To フィールドは無視します (gnus-summary-reply-broken-reply-to)。メーリングリストがそのリストを指す Reply-To を過って設定するためにこれが必要なのであれば、おそらくあなたは代わりに broken-reply-to グループパラメーターを設定する必要があります。そうすれば、ものごとは正しく働くようになるでしょう。See Section 2.10 [Group Parameters], page 23.
- S B R* 現在の記事を書いた人に対して元記事を含んだ返答をしますが Reply-To フィールドは無視します (gnus-summary-reply-broken-reply-to-with-original)。
- S o m*
C-c C-f 誰か他の人に現在の記事を転送します (gnus-summary-mail-forward)。接頭引数が与えられない場合、メッセージは message-forward-as-mime および message-forward-show-mml の値に従ったやり方で転送されます。接頭引数が 1 だったら、デコードされたメッセージが直接埋め込まれた転送用のバッファが作られます。2 だったら rfc822 形式の MIME パートが挿入されます。この場合、元のメッセージはデコードされません。3 ではデコードされた rfc822 形式の MIME パートが挿入されます (実際に送信する際に再びエンコードされます)。接頭引数 4 では、1 の場合と同じ動作になります。接頭引数がこれら以外の場合には、message-forward-as-mime の値を一時的に反転して、接頭引数が与えられなかった場合と同じ動作を行ないます。デフォルトでは、デコードされたメッセージが rfc822 形式の MIME パートとして生成されます。
- S m*
m メールを作成します (gnus-summary-mail-other-window)。デフォルトでは現在のグループの投稿様式 (posting style) を使います。接頭引数が与えられると、それは使いません。もし接頭引数が 1 だったら、どのグループの投稿様式を使うかを尋ねます。
- S i*
i ニュースを作成します (gnus-summary-news-other-window)。デフォルトでは現在のグループに投稿します。接頭引数が与えられると、現在のグループ名は使われません。もし接頭引数が 1 だったら、どのグループに投稿するかを尋ねます。
- この関数は、たとえメールグループで使われたとしても、実際にはニュースの様式を用意します。これは、メッセージを実際にはネットワーク経由で送らずに、メールグループに「投稿」するのに便利です; それらは当のグループに単に直接セーブされます。対応するバックエンドが投稿のためのメソッド (request-post method) を持っていなければなりません。
- S D b* メールを送ったのに、何らかの理由 (アドレスの間違い、転送の不調) で戻ってきたときに、この命令を使って戻ってきたメールをもう一回送ることができます (gnus-summary-resend-bounced-mail)。メールバッファにそのメールが現れて、そこでもう一度メー

ルを送る前にヘッダーを編集することができます。この命令に接頭引数を与えると、戻ってきたメールが何か他のメールへの返答であった場合に、Gnus はそのメールを取得して、そのヘッダーの精密調査ができるように画面に表示しようとします。ま、これはとてもよく失敗しますけど。

S D r 上の命令と混同しないで下さい。gnus-summary-resend-message は現在のメッセージを送る宛先のアドレスの入力を促して、その場所にメールを送ります。メッセージのヘッダーは変更されません—しかし Resent-To, Resent-From などの、たくさんのヘッダーが付け加えます。これは、(おそらく) あなた自身を To 欄に書いた本人にもメールを送ってしまうということです。これは混乱を招くでしょう。ですから当然、あなたが本当に邪悪な人でなければ、これは使わないでしょう。

この命令は主に、あなたがいくつかのメールアカウントを持っていて、自分自身の違ったアカウントにメールを転送したいときに用いられます。(もしあなたが root であり、postmaster でもあり、root 宛てに postmaster へのメールを受け取った場合は、それを postmaster にも再送する必要があるかもしれませんが。秩序がなければなりません! (Ordnung muss sein!))

この命令はプロセス/接頭引数の習慣に従います (see Section 8.1 [Process/Prefix], page 249)。

S D e 一つ前のコマンドに似ていますが、再送する前にあたかも新しいメッセージのように編集することができます。

S O m 現在の一連の記事 (see Section 3.16 [Decoding Articles], page 77) の要約を作り、メールでその結果を送ります (gnus-uu-digest-mail-forward)。この命令はプロセス/接頭引数の習慣に従います (see Section 8.1 [Process/Prefix], page 249)。

S M-c 現在の記事の書き手に、過剰なクロスポストへの苦情のメールを送ります (gnus-summary-mail-crosspost-complaint)。

この命令は、現在 Usenet に横行しているクロスポストの世界的流行に対して反撃を行なう手段として提供されています。これは変数 gnus-crosspost-complaint を序文にして返答を作成します。この命令はプロセス/接頭引数の習慣 (see Section 8.1 [Process/Prefix], page 249) に従い、それぞれのメールを送る前に送信するかどうかの確認をします。

また See section “ヘッダー命令” in *The Message Manual*, にさらなる情報があります。

3.5.2 概略の投稿命令

ニュースの記事を投稿するための命令:

S p
a 投稿するための記事を作成します (gnus-summary-post-news)。デフォルトでは現在のグループに投稿します。接頭引数が与えられると、現在のグループ名は使われません。もし接頭引数が 1 だったら、代わりに別のどのグループに投稿するかを尋ねます。

S f
f 現在の記事のフォローアップを投稿します (gnus-summary-followup)。

S F
F 元記事を取り込んで、現在の記事にフォローアップをします (gnus-summary-followup-with-original)。この命令はプロセス/接頭引数の習慣を用います。

- S n* メールのメッセージを受け取っていたとしても、現在の記事のフォローアップをニュースに投稿します (gnus-summary-followup-to-mail)。この命令はプロセス/接頭引数の習慣を用います。
- S N* メールのメッセージを受け取っていたとしても、元記事を引用して、現在の記事のフォローアップをニュースに投稿します (gnus-summary-followup-to-mail-with-original)。この命令はプロセス/接頭引数の習慣を用います。
- S o p* 現在の記事をニュースグループに転送します (gnus-summary-post-forward)。接頭引数が与えられない場合、メッセージは message-forward-as-mime および message-forward-show-mml の値に従ったやり方で転送されます。接頭引数が 1 だったら、デコードされたメッセージが直接埋め込まれた転送用のバッファが作られます。2 だったら rfc822 形式の MIME パートが挿入されます。この場合、元のメッセージはデコードされません。3 ではデコードされた rfc822 形式の MIME パートが挿入されます (実際に送信する際に再びエンコードされます)。接頭引数 4 では、1 の場合と同じ動作になります。接頭引数がこれら以外の場合には、message-forward-as-mime の値を一時的に反転して、接頭引数が与えられなかった場合と同じ動作を行ないます。デフォルトでは、デコードされたメッセージが rfc822 形式の MIME パートとして生成されます。
- S O p* 現在の一連の記事を要約して、その結果をニュースグループに送ります (gnus-uu-digest-post-forward)。この命令はプロセス/接頭引数の習慣を用います。
- S u* ファイルを uuencode して分割し、それらを連続して投稿します (gnus-uu-post-news)。 (see Section 3.16.5.3 [Uuencoding and Posting], page 80)。

また See section “ヘッダー命令” in *The Message Manual*, にさらなる情報があります。

3.5.3 概略メッセージ命令

- S y* 現在の記事を、すでに存在するメッセージ作成バッファに貼り付けます (gnus-summaryyank-message)。この命令は貼り付けたいメッセージバッファの入力を促し、プロセス/接頭引数の習慣を理解します (see Section 8.1 [Process/Prefix], page 249)。

3.5.4 記事を取り消す

何かを書いた後で、本当に、本当に、ほんとうにそれを投稿していなければなあと思ったことはありませんか。

えーと、メールは取り消すことはできないのですが、ニュースの投稿は取り消すことができます。

取り消したい記事を見つけて下さい (取り消すことができるのは自分の記事だけです。変なことは試さないで下さい)。そして *C* か *S c* を押して下さい (gnus-summary-cancel-article)。あなたの記事が取り消されます—世界中の機械があなたの記事を取り消します。この命令はプロセス/接頭引数の習慣を用います (see Section 8.1 [Process/Prefix], page 249)。

しかし注意して欲しいのは、すべてのサイトが取り消しを扱っているわけではないことです。ですから、たいていのサイトが問題の記事を取り消しても、あちこちであなたの記事は生き残るかもしれません。

Gnus は取り消すときに『現在』の選択方法を使います。標準の投稿方法を使いたいのであれば、文字接頭引数 ‘a’ を使って下さい (see Section 8.3 [Symbolic Prefixes], page 250)。

Gnus は Cancel-Lock ヘッダー (see section “ニュースを取り消す” in *The Message Manual*) を使って、あなただけがあなたのメッセージをキャンセルできるようにします。

もし何か間違いをしたのに気付いて、訂正をしたいのであれば、「代替」(*superseding*) 記事を投稿して元記事を置き換えることができます。

元記事のところへ移動して、*S s* を押して下さい (*gnus-summary-supersede-article*)。それを普通に送信する前に、記事を好きなように編集することができます。

代替に関しても、取り消しと同じことが当てはまります。こちらの方がもっとよく当てはまるかもしれませんが: いくつかのサイトは代替を受け付けません。これらのサイトでは、ほとんど同じ記事を二回投稿したようになってしまいます。

もしさっき記事を投稿したばかりですぐに変更したくなった場合、記事が最初にあなたのサイトに現れる前に取り消し/代替をするための巧妙な手段があります。まず、投稿バッファ (*sent ... * のようになっています) に戻して下さい。そこにはあなたがちょうど投稿した記事があり、すべてのヘッダーがそのままあります。Message-ID ヘッダーを Cancel もしくは Supersedes に変更して下さい。そして、いつもやっているように単に *C-c C-c* を押して記事を送信して下さい。前の記事は取り消されるか置き換えられるでしょう。

ちょっと覚えておいて下さい: 'supersede' (代替) という語の中に 'c' は無いということ。

3.6 遅延記事

ときとして、あなたはメッセージの送信を先延ばしにしたいと思うことはありませんか。例えば、あなたが大切なだれかの誕生日を思い出すために、ちょうどその日に届くメッセージを用意したいと思ったとしましょう。gnus-delay パッケージはこれにうってつけです。設定は簡単です:

```
(gnus-delay-initialize)
```

普段はメッセージを送信するのに Message モードで *C-c C-c* コマンドを使いますよね。先延ばしにするには、代わりに *C-c C-j* (*gnus-delay-article*) を使って下さい。そうすると、どのくらい遅らせるかを尋ねてきます。可能な返事は次の通りです:

- 期間。整数と一つの文字で指定します。例えば 42d は 42 日遅らせることを意味します。使うことができる文字は m (分)、h (時)、d (日)、w (週)、M (月) および Y (年) です。
- 日付。YYYY-MM-DD のような形式で指定します。メッセージの送信はその日の特定の時刻 (デフォルトは 8 時) まで遅らせられます。gnus-delay-default-hour も参照して下さい。
- 時刻。am/pm を含まない 24 時間制の、hh:mm の形式で与えます。送信されるのは今日のその時刻ですが、すでにその時刻を過ぎてしまっていた場合は翌日のその時刻になります。ですから、朝の 10 時に 11:15 を指定した場合は 1 時間 15 分後に送信されることになります。しかし 9:20 を指定した場合は翌日の時刻を意味します。

gnus-delay-article の動作は、以下の数個の変数に影響されます:

```
gnus-delay-default-hour
```

特定の日付を指定した場合に、メッセージがその日の何時に送信されるかを与えます。可能な値は 0 から 23 までの整数です。

```
gnus-delay-default-delay
```

デフォルトの遅延を与える文字列です。前述のどんな形式でも可能です。

```
gnus-delay-group
```

遅延記事は、ドラフトサーバーのこのグループに期限が来るまで保管されます。たぶんあなたはこれを変更する必要は無いでしょう。デフォルトの値は "delayed" です。

gnus-delay-header

それぞれの記事が送信される日時はヘッダーに記録されます。この変数はヘッダー名の文字列です。たぶんあなたはこれを変更する必要は無いでしょう。デフォルトの値は "X-Gnus-Delayed" です。

送信の先延ばしはこんなふうに行なわれます: `gnus-delay-article` コマンドで、あなたはどのくらい遅らせるかを指定します。Gnus はメッセージを送信する日時を計算して X-Gnus-Delayed ヘッダーに記録し、そのメッセージを `nn draft:delayed` グループに納めます。

そして、あなたが新着ニュースを取得しようとするときはいつも、Gnus は送信する期限に達した記事をそのグループで探して、それらを送信します。これには関数 `gnus-delay-send-queue` が使われます。デフォルトではこの関数は `gnus-get-new-news-hook` に追加されますが、もちろんあなたはこれを変更することができます。おそらくあなたは、ドラフトの送信にデーモンを使いたいと思うのではないのでしょうか？ それには、デーモンに関数 `gnus-delay-send-queue` を実行せよ、と言うだけで良いのです。

gnus-delay-initialize

デフォルトではこの関数は `gnus-delay-send-queue` の `gnus-get-new-news-hook` への追加を行ないます。ですが、これは第二オプション引数 `no-check` を受け付けます。もしそれが非-nil だったら `gnus-get-new-news-hook` は変更されません。第一オプション引数は無視されます。

例えば (`gnus-delay-initialize nil t`) は何もしないことを意味します。あなたは遅延記事の送信にデーモンを使いたいのでしょね。でも、それを設定することを忘れないで下さいね。:-)

3.7 記事に印を付ける

記事に付けられる印はいくつかあります。

記事の「購読度」(うひょーっ、何てすばやしい造語だ!) を決定する印があります。英字でない文字が一般に「未読」を意味するのに対して、英字の印は一般に「既読」を意味します。

加えて、購読度に影響しない印もあります。

3.7.1 未読記事

以下の印は何らかの方法で記事に未読の(ような)印を付けます。

‘!’ 可視記事 (ticked) として印を付けます (`gnus-ticked-mark`)。

「可視記事」とは常に可視状態である記事のことです。おもしろいと思う記事があった場合や、読むのを先に延ばしたいときや、後で返答をしたいときに、普通は可視印を付けます。しかし、記事は期限切れ消去されることもあります(ニュースサーバー上の記事を消去するのはニュースサーバーのソフトウェアで、Gnus 自体は可視記事を期限切れ消去しません)ので、永遠に記事を保存しておきたい場合は、その記事を永続にする必要があります (see Section 3.13 [Persistent Articles], page 72)。

‘?’ 保留として印を付けます (`gnus-dormant-mark`)。

「保留記事」はフォローアップがあったときにだけ概略バッファに現れます。フォローアップが無いときも表示させたいときは、`/D` 命令を使って下さい (see Section 3.8 [Limiting], page 61)。それ以外は(見えるかどうかは別にして)、可視記事 (ticked) と似たようなものです。

‘SPACE’ 未読として印を付けます (gnus-unread-mark)。
 「未読記事」は今までまったく読まれていない記事のことです。

3.7.2 既読記事

以下のすべての印は記事に既読の印を付けます。

- ‘r’ 利用者が手動で `d` 命令もしくはそれに類する手段を使って、既読の印を付けた記事です (gnus-del-mark)。
- ‘R’ 実際に読まれた記事 (gnus-read-mark)。
- ‘O’ 前回のセッションで既読の印を付けて、今は「古く」なってしまった記事。
- ‘K’ 削除された印 (gnus-killed-mark)。
- ‘X’ 削除ファイルによって削除の印が付いた記事 (gnus-kill-file-mark)。
- ‘Y’ 低すぎるスコアのために既読の印が付いた記事 (gnus-low-score-mark)。
- ‘C’ キャッチアップによって既読の印が付いた記事 (gnus-catchup-mark)。
- ‘G’ 取り消された記事 (gnus-canceled-mark)。
- ‘F’ SOUP されている記事 (gnus-souped-mark)。See Section 6.6.4 [SOUP], page 199.
- ‘Q’ まばらに参照された記事 (gnus-sparse-mark)。See Section 3.9.1 [Customizing Threading], page 63.
- ‘M’ 重複抑制により既読の印の付いた記事 (gnus-duplicate-mark)。See Section 3.29 [Duplicate Suppression], page 110.

これらのすべての印は、本当にただ記事が既読として印が付いていることを意味するだけです。適応スコアリングをしたときには違ったように解釈されますけれど。

もう一つ、特別な印があります:

- ‘E’ 期限切れ消去可能として印の付いた記事 (gnus-expirable-mark)。
 記事を「期限切れ消去可能」として印を付ける (もしくは、自動的にそのように印を付ける) ことは、普通のグループではあまり意味がありません—利用者はニュース記事の期限による削除を制御していません。しかし、例えばメールグループでは、「期限切れ消去可能」として印の付いた記事は、いつでも Gnus によって削除されることがあります。

3.7.3 他の印

記事が読まれたかどうかには関係しない印がいくつかあります。

- 現在の記事にしおりを挟むことができます。あなたは猫のおしこの習慣に関する長い論文を読んでいて、それを読み終わる前に晩ご飯を食べに家に帰らなければならなかったとしましょう。そんなとき、記事にしおりを挟むことができます。次にその記事に出くわすと、Gnus はそのしおりのところへ移動するでしょう。See Section 3.7.4 [Setting Marks], page 58.
- 返信したかフォローアップした (つまり、答えた) すべての記事には、二桁目に ‘A’ の印が付きます (gnus-replied-mark)。
- 転送したすべての記事には、二桁目に ‘F’ の印が付きます (gnus-forwarded-mark)。
- 記事キャッシュに貯められている記事は、二桁目に ‘*’ の印が付きます (gnus-replied-mark)。See Section 3.12 [Article Caching], page 71.

- (何らかの方法によって; 必ずしも宗教的というわけではなく) 『救済された』 (原文は saved==保存された) 記事は、二桁目に ‘S’ の印が付きます (gnus-saved-mark)。
まだサーバーが利用者に見せていない記事は、二桁目に ‘N’ の印が付きます (gnus-recent-mark)。たいていのサーバーはこの印をサポートせず、その場合は単に表示されません。gnus-unseen-mark と見比べて下さい。
- まだ Gnus で読まれたことがない記事は、二桁目に ‘.’ の印が付きます (gnus-unseen-mark)。gnus-recent-mark と見比べて下さい。
- Gnus エージェント (see Section 6.9.1 [Agent Basics], page 210) を使っているとき、記事は unplugged (ネットワークから切り離されている状態) で見られるためにダウンロードされるかもしれませんが。‘%0’ の仕様を使っていると、それらの記事にはその仕様に ‘+’ の印が付きます。(変数 gnus-downloaded-mark でどの文字を使うかを制御します。)
- Gnus エージェント (see Section 6.9.1 [Agent Basics], page 210) を使っているとき、いくつかの記事はダウンロードされていないかもしれません。Unplugged (ネットワークから切り離されている状態) ではそのような記事を見ることができません。‘%0’ の仕様を使っていると、それらの記事にはその仕様に ‘-’ の印が付きます。(変数 gnus-undownloaded-mark でどの文字を使うかを制御します。)
- Gnus エージェント (see Section 6.9.1 [Agent Basics], page 210) はいくつかの記事を自動的にダウンロードしますが、自動的にダウンロードされない記事にもダウンロードのための明示的な印を付けることは可能です。そのような明示的に印が付けられた記事には、最初の桁に ‘%’ の印が付きます。(変数 gnus-downloadable-mark でどの文字を使うかを制御します。)
- ‘%e’ の仕様が使われると、スレッドがあるかどうかの印が gnus-not-empty-thread-mark または gnus-empty-thread-mark によって、三桁目に付きます。
- 最後に「プロセス印」があります (gnus-process-mark)。いろいろな種類の命令が、プロセス印があるとそれに対して実行されます。例えば `X u` (gnus-uu-decode-uu) は、プロセス印の付いたすべての記事を `uudecode` して表示します。プロセス印の付いた記事は二桁目に ‘#’ があります。

たいていのこれら『購読度と関係無い』印は、デフォルトでは二桁目に現れることに気付いたでしょう。では、キャッシュされていて、保存されていて、返答した記事にプロセス印を付けた場合は、どうなるのでしょうか?

たいしたことはありません。優先順位は次のようになっています: プロセス キャッシュ 返答済み 保存。ですから、ある記事がキャッシュに入っていて返答されていた場合、キャッシュ印が見えるだけで、返答済み印は見えません。

3.7.4 印を付ける

すべての印を付けるための命令は、数値接頭引数を受け付けます。

M c

M-u 現在の記事から、すべての購読度に関する印を消去します (gnus-summary-clear-mark-forward)。要するに、記事に未読の印を付けます。

M t

! 現在の記事に可視記事の印を付けます (gnus-summary-tick-article-forward)。See Section 3.12 [Article Caching], page 71.

M ?

? 現在の記事に保留記事の印を付けます (gnus-summary-mark-as-read-forward)。See Section 3.12 [Article Caching], page 71.

<i>M d</i>	
<i>d</i>	現在の記事に既読の印を付けます (gnus-summary-mark-as-read-forward)。
<i>D</i>	現在の記事に既読の印を付け、前の行にポイントを移動します (gnus-summary-mark-as-read-backward)。
<i>M k</i>	
<i>k</i>	現在の記事と同じ表題を持つすべての記事を既読として印を付け、次の未読記事を選択します (gnus-summary-kill-same-subject-and-select)。
<i>M K</i>	
<i>C-k</i>	現在の記事と同じ表題を持つすべての記事を既読として印を付けます (gnus-summary-kill-same-subject)。
<i>M C</i>	すべての未読記事に既読の印を付けます (gnus-summary-catchup)。
<i>M C-c</i>	グループのすべての記事に——可視記事や保留記事でさえも、既読の印を付けます (gnus-summary-catchup-all)。
<i>M H</i>	現在のグループの、現在位置とそれ以前の記事を既読として印を付けます (gnus-summary-catchup-to-here)。
<i>M h</i>	現在のグループの、現在位置とそれ以降の記事を既読として印を付けます (gnus-summary-catchup-from-here)。
<i>C-w</i>	ポイントとマークの間の記事に既読の印を付けます (gnus-summary-mark-region-as-read)。
<i>M V k</i>	デフォルトのスコア (もしくは数値接頭引数) よりも低いスコアの記事を削除します。
<i>M e</i>	
<i>E</i>	現在の記事を期限切れ消去可能として印を付けます (gnus-summary-mark-as-expirable)。
<i>M b</i>	現在の記事にしおりを設定します (gnus-summary-set-bookmark)。
<i>M B</i>	現在の記事のしおりを削除します (gnus-summary-remobe-bookmark)。
<i>M V c</i>	デフォルトのスコア (もしくは数値接頭引数) よりも大きいスコアを持つ記事のすべての印を消去します (gnus-summary-clear-above)。
<i>M V u</i>	デフォルトのスコア (もしくは数値接頭引数) よりも大きいスコアを持つすべての記事に可視印を付けます (gnus-summary-tick-above)。
<i>M V m</i>	印の入力を促し、デフォルトのスコア (もしくは数値接頭引数) よりも大きなスコアを持つすべての記事にその印を付けます (gnus-summary-mark-above)。

変数 `gnus-summary-goto-unread` は印が付けられた後にどのような動作がなされるかを決定します。もし `nil` でないと、ポイントは次/前の未読記事に移動します。もし `nil` であると、ポイントは一行上か下に行くだけです。特別な場合として、この変数が `never` であると、すべての印を付ける命令と (`SPACE` のような) 他の命令は次の記事が未読であろうが無かろうが次の記事に移動します。デフォルトは `t` です。

3.7.5 Generic Marking Commands

記事に可視の印を付ける命令 (!) に、次の記事に移動してもらいたい人がいます。次の未読記事に移動してもらいたい人もいます。さらに、現在の記事に留まっていてもらいたい人もいるでしょう。そして、前の (未読の) 記事に行って欲しい人がいるとはまだ聞いたことはありませんが、そうしたいと思う人も間違いなくいると思います。

この五つの動作を五つの違った印付け命令と掛け算すると、どの命令が何をすべきかの非常に複雑な変数の組を持つことになります。

この窮地を脱するために、Gnus はこれらすべての違ったことをする命令を提供します。これらは概略バッファの *MM* マップにあります。すべてを見るためには *MM C-h* を入力して下さい—このマニュアルで一覧を出すには多過ぎます。

これらの命令を直接使うことはできますが、ほとんどの利用者は概略モードのキーマップを交換する方を好むでしょう。例えば、! 命令に次の未読記事の代わりに次の記事に移動して欲しいとすると、このようなことができます:

```
(add-hook 'gnus-summary-mode-hook 'my-alter-summary-map)
(defun my-alter-summary-map ()
  (local-set-key "!" 'gnus-summary-put-mark-as-ticked-next))
```

もしくは、

```
(defun my-alter-summary-map ()
  (local-set-key "!" "MM!n"))
```

3.7.6 プロセス印を付ける

プロセス印は概略バッファに # として表示され、他のコマンドで処理させる記事に印を付けるために使われます。例えば、四つの記事に印を付けてから * コマンドを使うと、Gnus はそれら四つの記事をキャッシュに入れます。詳しくは Section 8.1 [Process/Prefix], page 249 をどうぞ。

MP p

現在の記事にプロセス印を付けます (gnus-summary-mark-as-processable)。

MP u

M-# もし現在の記事にプロセス印があれば取り除きます (gnus-summary-unmark-as-processable)。

MP U

すべての記事からプロセス印を取り除きます (gnus-summary-unmark-all-processable)。

MP i

プロセス印の付いている記事とそうでない記事を逆にします (gnus-uu-mark-by-regexp)。

MP R

正規表現に合致する Subject ヘッダーを持つ記事に印を付けます (gnus-uu-mark-by-regexp)。

MP G

正規表現に合致する Subject ヘッダーを持つ記事から印を削除します (gnus-uu-unmark-by-regexp)。

MP r

領域にある記事に印を付けます (gnus-uu-mark-region)。

MP g

領域にある記事から印を削除します (gnus-uu-unmark-region)。

MP t

現在のスレッド (または副スレッド) のすべての記事に印を付けます (gnus-uu-mark-thread)。

<i>M P T</i>	現在のスレッド (または副スレッド) のすべての記事から印を取り除きます (gnus-uu-unamrk-thread)。
<i>M P v</i>	接頭引数よりも大きなスコアを持つすべての記事に印を付けます (gnus-uu-mark-over)。
<i>M P s</i>	現在の一連の記事に印を付けます (gnus-uu-mark-series)。
<i>M P S</i>	すでいくつかの印の付いた記事を持つ一連の記事群すべてに印を付けます (gnus-uu-mark-sparse)。
<i>M P a</i>	一連の記事が出てくる順番にそれに属するすべての記事に印を付けます (gnus-uu-mark-all)。
<i>M P b</i>	バッファのすべての記事を現れている順番に印を付けます (gnus-uu-mark-buffer)。
<i>M P k</i>	現在のプロセス印をスタックに積んで、すべての記事を無印にします (gnus-summary-kill-process-mark)。
<i>M P y</i>	スタックから前回のプロセス印を取り出して、それを復元します (gnus-summary-yank-process-mark)。
<i>M P w</i>	現在のプロセス印をスタックに積みます (gnus-summary-save-process-mark)。

そして、記事の本文の内容に基づいてプロセス印を付けるやり方については、Section 3.26.2 [Searching for Articles], page 107 の `&` 命令を参照して下さい。

3.8 制限をする

現在グループにある記事の一部だけを表示するように概略バッファを制限できれば便利があります。多くの制限命令が持つ効果は、概略バッファから少し (もしくは多く) の記事を削除することです。

すべての制限命令はサーバーからすでに取得された記事の一部分に作用します。これらの命令はどれもサーバーに追加の記事を要求しません。

<code>//</code>	
<code>/ s</code>	概略バッファをいくつかの表題と合致するものだけに制限します (gnus-summary-limit-to-subject)。接頭引数が与えられると、合致する記事を除外します。
<code>/ a</code>	概略バッファを何人かの著者に合致するものだけに制限します (gnus-summary-limit-to-author)。接頭引数が与えられると、合致する記事を除外します。
<code>/ R</code>	概略バッファをいくつかの受信者に合致する記事だけに制限します (gnus-summary-limit-to-recipient)。接頭引数が与えられると、合致する記事を除外します。
<code>/ S</code>	概略バッファを表示されているスレッドに属さない記事だけに制限します (gnus-summary-limit-to-singletons)。接頭引数が与えられると、表示されているスレッドに属する記事だけに制限します。
<code>/ x</code>	「追加」のヘッダーの一つに合致する記事に概略バッファを制限します (see Section 3.1.2 [To From Newsgroups], page 46) (gnus-summary-limit-to-extra)。接頭引数が与えられると、合致する記事を除外します。
<code>/ u</code>	
<code>x</code>	概略バッファを既読の印が付いていない記事に制限します (gnus-summary-limit-to-unread)。接頭引数が与えられると、バッファを完全に未読記事のみに制限します。これは、可視と保留の記事は含まれないということです。

- `/ m` 印を尋ねて、その印が付いている記事に制限します (`gnus-summary-limit-to-marks`)。
- `/ t` 数値を尋ねて、概略バッファをその日数より古い (もしくは同じ) 記事に制限します (`gnus-summary-limit-to-age`)。接頭引数が与えられると、その数値の日よりも新しい記事に制限します。
- `/ n` 概略バッファを、接頭引数 ‘n’ で指定された次の ‘n’ 個の記事に制限します。接頭引数が与えられないと、代わりにプロセス印が付いている記事に制限します (`gnus-summary-limit-to-articles`)。
- `/ w` 前の制限をスタックから取り出して、復元します (`gnus-summary-pop-limit`)。接頭引数が与えられると、すべての制限をスタックから取り出します。
- `/ .` 概略バッファをまだ読まれたことが無い記事に制限します (`gnus-summary-limit-to-unseen`)。
- `/ v` 概略バッファをあるスコアと同じか、それより大きなスコアを持つ記事に制限します (`gnus-summary-limit-to-score`)。
- `/ p` 概略バッファを `display` グループパラメーターの述語を満足させるように制限します (`gnus-summary-limit-to-display-predicate`)。この述語に関する詳細は Section 2.10 [Group Parameters], page 23 を参照して下さい。
- `/ r` 概略バッファを返信した記事だけに制限します (`gnus-summary-limit-to-replied`)。接頭引数が与えられると、返信した記事以外の記事に制限します。
- `/ E`
`MS` すべての消去された記事を制限に含めます (`gnus-summary-limit-include-expunged`)。
- `/ D` すべての保留記事を制限に含めます (`gnus-summary-limit-include-dormant`)。
- `/ *` すべてのキャッシュに入っている記事を制限に含めます (`gnus-summary-limit-include-cached`)。
- `/ d` すべての保留記事を制限から除外します (`gnus-summary-limit-exclude-dormant`)。
- `/ M` すべての印付き記事を除外します (`gnus-summary-limit-exclude-marks`)。
- `/ T` 現在のスレッドのすべての記事を制限に含めます (`gnus-summary-limit-include-thread`)。
- `/ c` 子記事の無いすべての保留記事を制限から除外します (`gnus-summary-limit-exclude-childless-dormant`)。
- `/ C` すべての除外された未読の記事に既読の印を付けます (`gnus-summary-limit-mark-excluded-as-read`)。接頭引数が与えられると、可視と保留のみの印の記事も既読として印を付けます。
- `/ N` すべての新しい記事を概略バッファに挿入します。`back-end-get-new-mail` が非-nil だったら、新しいメールの到来を調べることです。
- `/ o` すべての古い記事を概略バッファに挿入します。数値の接頭引数が与えられると、その個数の記事を取り込みます。

- / b 概略バッファを、ある正規表現に本文が合致する記事だけに制限します (gnus-summary-limit-to-bodies)。接頭引数が与えられると、制限を逆にします (訳注: 合致しない記事だけに制限します)。合致するものを探すためにそれぞれの記事を取り込まなければならないので、このコマンドはとても遅いです。
- / h この前のコマンドに似ていますが、代わりにこれは、ある正規表現にヘッダーが合致する記事だけに制限します (gnus-summary-limit-to-headers)。

3.9 スレッド

Gnus はデフォルトで記事をスレッド表示します。「スレッドにする」とは、ある記事への応答を応答した記事の直後に置く一階層的流儀で、ということです。

スレッドは記事の References 欄を調べることによって行なわれます。理想的な世界では、これだけで木を完成させるのに十分なのですが、不運なことに References 欄はしばしば壊れているか、時には単に無いことがあります。怪しげなニュースの伝搬は問題を悪化させるので、満足な結果を得るためには他の検出法を採用しなければなりません。過剰な対策法は存在していて、その恐るべき詳細は Section 3.9.1 [Customizing Threading], page 63 に詳しく書いてあります。

まず、概念の概観です:

根本 (*root*)

スレッドで一番頂点にある記事です; スレッドの最初の記事です。

スレッド (*thread*)

木のような記事の構成です。

副スレッド (*sub-thread*)

木のような構造の (より) 小さな部分です。

無束縛スレッド (*loose threads*)

記事の期限切れ消去や、根本がすでに前回のセッションで読まれたことにより概略バッファに表示されない、等の理由により、スレッドはしばしば根本を失います。そのようなときには、普通は多くの副スレッドがあって、本当は一つのスレッドに属しているのですが、根本にはつながっていない、ということになります。こういうスレッドが無束縛スレッドと呼ばれています。

スレッド集め (*thread gathering*)

副スレッドを大きなスレッドに集めようとする試みです。

まばらスレッド (*sparse threads*)

そこではたぶんいくつかの記事が失われてしまったのだらうと『推測された』スレッドのことで、概略バッファでは空行で表示されます。

3.9.1 スレッドをカスタマイズする

3.9.1.1 無束縛スレッド

gnus-summary-make-false-root

もし nil でないと、Gnus はすべてのつながっていない部分木を一つの大きな木にして、頂上にみせかけの根本を作ります。(ちょっと待って下さい。頂上に根元 (*root*) ですって? ええ、そうなのです。) つながっていない部分木は本当の根本が期限切れ消去されたか、前回のセッションで根本を読んだり削除したときにできます。

本当のスレッドが無いときは、Gnus は何かでっち上げをする必要があります。この変数は Gnus が使うべきごまかしの方法を示しています。値としてとることができる四つの候補があります。

養子 (adopt)

Gnus は孤児になった記事群の最初のを親にします。この親はすべての他の記事を養子にします。それらの養子記事は、標準の角括弧 ('[]') の代わりに、先の尖った括弧 ('<>') で印が付けられます。これがデフォルトの手段です。

みせかけ (dummy)

Gnus は親のふりをするみせかけの概略行を作ります。このみせかけの行はどの本当の記事にも対応しないので、それを選択することは、みせかけの記事の後の最初の本当の記事を選択をするだけになります。みせかけの根本の様式を指定するために、`gnus-summary-dummy-line-format` が使われます。これはたった一つだけのフォーマットの仕様を受け付けます: それは 'S' で、記事の表題です (see Section 8.4 [Formatting Variables], page 250)。たとえ集めるものが無くても、すべてのスレッドにみせかけの根本を持たせたい場合は、`gnus-summary-make-false-root-always` を `t` に設定して下さい。

空 (empty)

Gnus は実際にはどの記事も親にはせず、最初の孤児を除いてすべての孤児の表題欄を単に空のままにします。(実際は `gnus-summary-same-subject` を表題として使います (see Section 3.1 [Summary Buffer Format], page 43)。)

`none` まったくどの記事も親にしません。スレッドを集めて、単に順繰りに表示するだけです。

`nil` 無束縛スレッドを集めません。

`gnus-summary-gather-subject-limit`

無束縛スレッドは記事の表題を比較することによって集められます。もしこの変数が `nil` であると、Gnus は無束縛スレッドを一つの大きな超スレッドに集める前に、無束縛スレッドの表題が完全に一致することを要求します。これは、長い表題の行を切り落としてしまう間抜けなニュースリーダーが存在する現状では、あまりに厳しい要求かもしれません。そう思うのなら、この変数を例えば 20 に設定して、表題の最初の 20 文字だけが一致することを要求するようにして下さい。この変数を本当に低い数値に設定すると、目についたもののすべてを Gnus が一つのスレッドに集めるのを見ることになるでしょう。それはあまり有用ではありません。

この変数を特別な値 `fuzzy` に設定すると、Gnus は表題の文字列を大雑把に比較するアルゴリズムを使います (see Section 8.18 [Fuzzy Matching], page 269)。

`gnus-simplify-subject-fuzzy-regexp`

正規表現または正規表現のリストのどちらかです。表題の大雑把な比較を行なうときに、それらに合致する文字列を表題から取り除きます。

`gnus-simplify-ignored-prefixes`

もし `gnus-summary-gather-subject-limit` を 10 くらいに低く設定したならば、この変数を何か意味のあるものに設定することを考えるでしょう:

```
(setq gnus-simplify-ignored-prefixes
      (concat
        "\\['\\[?\\\\"
        (mapconcat
          'identity
          '("looking"
            "wanted" "followup" "summary\\( of\\)?"
            "help" "query" "problem" "question"
            "answer" "reference" "announce"
            "How can I" "How to" "Comparison of"
            ;; ...
          )
        "\\|")
        "\\)\\s *\\\\"
        (mapconcat 'identity
          '("for" "for reference" "with" "about")
          "\\|")
        "\\)?\\[?:?[ \\t]*"))
```

この正規表現に合致するすべての語は、二つの表題を比較する前に取り除かれます。

gnus-simplify-subject-functions

nil でないと、この変数は gnus-summary-gather-subject-limit よりも優先されます。この変数は Subject の文字列に反復して作用させて簡単にするための、関数のリストである必要があります。

このリストに入れて役に立つような関数は次のようなものです:

gnus-simplify-subject-re

前の方にある 'Re:' を取り除きます。

gnus-simplify-subject-fuzzy

大雑把な比較ができるように簡単にします。

gnus-simplify-whitespace

余分な空白 (whitespace) を取り除きます。

gnus-simplify-all-whitespace

すべての空白 (whitespace) を取り除きます。

もちろん、あなた自身の関数を書くこともできます。

gnus-summary-gather-exclude-subject

無束縛スレッド集めは表題だけで行なわれるので、特に ' や '(none)' のような良くある表題のときは、多くの間違いを起こす可能性があります。この状況を少し良くするために、正規表現 gnus-summary-gather-exclude-subject を使うことによって、集める過程においてどんな表題を除外するかを指示することができます。デフォルトは '^ *\$\\|^^(none)\$' です。

gnus-summary-thread-gathering-function

Gnus は Subject 欄を調べることによってスレッドを集めます。これは、結果的にまったく関係の無い記事が同じ『スレッド』に含まれるかもしれないことを意味し、混乱の元です。代替手段は、合致するものを見つけるために References 欄にある Message-ID

をすべて調べることです。これは集められたスレッドが関係の無い記事をまったく含まないことを保証しますが、いかれたニュースリーダーで投稿した記事は適切に集められないということでもあります。ペストかコレラかの選択権はあなたにあります。

`gnus-gather-threads-by-subject`

この関数はデフォルトの収集関数で、排他的に Subject を調べます。

`gnus-gather-threads-by-references`

この関数は排他的に References 欄を調べます。

References によって集めることを試してみたいのであれば、次のようにすることができます:

```
(setq gnus-summary-thread-gathering-function
      'gnus-gather-threads-by-references)
```

3.9.1.2 スレッドを埋める

`gnus-fetch-old-headers`

もし `nil` でないと、Gnus は古いスレッドをもっと古いヘッダー、すなわち既読の印が付いている記事のヘッダー、を取得することで構築しようとしています。できるだけ少ない概略行を表示したいけれど、できるだけたくさんの無束縛スレッドをつなげておきたいときは、この変数を `some` か数値に設定して下さい。もし数値に設定したときは、それより多い追加のヘッダーは取得されません。どちらの場合でも、古いヘッダーの取得は、使っているバックエンドが `overview` ファイルを使っている場合だけ動作します。それらのバックエンドは、普通は `nntp`, `nnsPOOL`, `nnml` および `nnmaildir` です。スレッドの根がサーバーによって期限切れ消去されてしまったら、Gnus はどうしようもないことも覚えておいて下さい。

この変数は `invisible` に設定することもできます。これは視覚的な効果は何もありませんが、`A T` 命令をよく使うのであれば役に立つでしょう (see Section 3.22 [Finding the Parent], page 99)。

`gnus-fetch-old-ephemeral-headers`

`gnus-fetch-old-headers` と同じですが、一時ニュースグループのためにだけ使われます。

`gnus-build-sparse-threads`

古いヘッダーを取得すると遅くなることがあります。この変数を `some` に設定することによって、同じような低賃金の効果を得ることができます。そうすると、Gnus はすべての記事の完全な References 欄を見て、同じスレッドに属する記事をつなごうとします。これは、記事がそのスレッドから失われていると Gnus が推測したスレッド表示に「ずれ」を残すでしょう。(これらのずれは普通の概略行のように見えます。もしずれを選択すると、Gnus はその当の記事を取得しようとしています。) この変数が `t` であると、Gnus はスレッドを補完するのに役立つかどうかを考慮せずに、すべての「ずれ」を表示します。最後に、この変数が `more` であると、Gnus はどこにもつながっていない枝葉のまばらな節を切り落としません。この変数はデフォルトでは `nil` です。

`gnus-read-all-available-headers`

これはあまり役に立たない、いささかはっきりしない変数です。ニュースではないグループにおいて、概略バッファを作るためにバックエンドが極めて多くのものを取り込まなければならない、しかも親記事を辿ることができない場合に使うことを想定しています。それは主に `nnultimate` グループのような、ウェブに基づいたグループでの場合です。

そんなグループを使わない場合はデフォルトの `nil` のままにしておくのが無難です。使いたい場合はグループ名に合致する正規表現か、すべてのグループが対象になる `t` にして下さい。

3.9.1.3 もっとスレッドを

`gnus-show-threads`

この変数が `nil` であると、スレッドは作られず、ここにある残りのすべての変数はまったく効果が無くなります。スレッド作りを止めるとグループの選択が少し速くなりますが、記事を読むのがもっと遅く、不便になることは確実です。

`gnus-thread-hide-subtree`

これが `nil` でないと、すべてのスレッドは概略バッファが生成されたときに隠れます。

これは述語指示子であることもできます (see Section 8.14 [Predicate Specifiers], page 264)。利用できる述語は `gnus-article-unread-p` と `gnus-article-unseen-p` です。

これは例です:

```
(setq gnus-thread-hide-subtree
      '(or gnus-article-unread-p
           gnus-article-unseen-p))
```

(これはかなりばかげた例です。なぜならすべてのまだ読まれたことが無い記事は未読でもあるからなのですが、趣旨は汲み取して下さい。)

`gnus-thread-expunge-below`

この数値より少ない総スコア (`gnus-thread-score-function` で定義された関数を使って算出されます) を持つすべてのスレッドは消去されます。この変数はデフォルトでは `nil` で、これはどのスレッドも消去されないということです。

`gnus-thread-hide-killed`

スレッドを削除すると、この変数が `nil` でない場合、部分木は隠されます。

`gnus-thread-ignore-subject`

ときどき誰かがスレッドの途中で表題を変更することがあります。この変数が `nil` でないと (これがデフォルトですが)、表題の変更は無視されます。もし `nil` だと、表題の変更をすると別のスレッドになります。

`gnus-thread-indent-level`

これは、それぞれの副スレッドがどれくらい字下げ (`indent`) されるべきかを決める数値です。デフォルトは 4 です。

`gnus-sort-gathered-threads-function`

とりわけメーリングリストでは、ときとして手元にメールが到着する順番は必ずしもメーリングリストに到着した順番と同じでは無いことがあります。その結果、副スレッドをデフォルトの `gnus-thread-sort-by-number` で並べ換えると、応答の方がその元記事より先に現れてしまうことがあります。グループパラメーターや適切なフック (例えば `gnus-summary-generate-hook`) でこの変数を代替の値 (例えば `gnus-thread-sort-by-date`) に設定することによって、そのような場合に、より論理的な副スレッドの順番を生成することができます。

3.9.1.4 低レベルにおけるスレッド作成

gnus-parse-headers-hook

すべてのヘッダーを解析する前に実行されるフックです。

gnus-alter-header-function

この変数の値が `nil` ではなくて関数であると、ヘッダー構造 (訳注: 記事の主要なヘッダーの内容を効率良く保持するための Lisp オブジェクト) を変更するために呼ばれます。関数は記事ヘッダーのベクトル (訳注: すなわちヘッダー構造) とともに呼ばれ、それが何らかの方法で変更されます。例えば Message-ID を体系的な方法で (接頭語などを付け加えることによって) 変更してしまうメールからニュースへのゲートウェイがある場合、この変数を設定することによって、その Message-ID を元の意味のあるものに戻すことができます。これは一つの例です:

```
(setq gnus-alter-header-function 'my-alter-message-id)

(defun my-alter-message-id (header)
  (let ((id (mail-header-id header)))
    (when (string-match
           "\\(<[^<>@]*\\)\\.?.*cygnus\\.\\.?.*\\(<[^<>@]*>\\)" id)
      (mail-header-set-id
       (concat (match-string 1 id) "@" (match-string 2 id))
       header))))
```

訳注: 取得した記事の Message-ID 欄から、'@' の前に付加された 'cygnus.' で始まる文字列を取り除きます。

3.9.2 スレッドの命令

T k

C-M-k 現在のスレッド (または副スレッド) のすべての記事に既読の印を付けます (`gnus-summary-kill-thread`)。もし接頭引数が正であると、代わりにすべての印を取り除きます。接頭引数が負であると、代わりに記事を可視にします。

T l

C-M-l 現在のスレッド (または副スレッド) のスコアを下げます (`gnus-summary-lower-thread`)。

T i

現在のスレッド (または副スレッド) のスコアを上げます (`gnus-summary-raise-thread`)。

T #

プロセス印を現在のスレッド (または副スレッド) に付けます (`gnus-uu-mark-thread`)。

T M-#

現在のスレッド (または副スレッド) からプロセス印を取り除きます (`gnus-uu-unmark-thread`)。

T T

スレッド表示を切り替えます (`gnus-summary-toggle-threads`)。

T s

もしあれば、現在の記事の下に隠れているスレッドを表示します (`gnus-summary-show-thread`)。

T h

現在のスレッド (または副スレッド) を隠します (`gnus-summary-hide-thread`)。

T S

すべての隠されているスレッドを表示します (`gnus-summary-show-all-threads`)。

<i>T H</i>	すべてのスレッドを隠します (<code>gnus-summary-hide-all-threads</code>)。
<i>T t</i>	現在の記事のスレッドをもう一度作り直します (<code>gnus-summary-rethread-current</code>)。これは概略バッファがスレッド表示されていないときでも動作します。
<i>T ^</i>	現在の記事を印付きの (もしくは前の) 記事の子記事にします (<code>gnus-summary-reparent-thread</code>)。
<i>T M-^</i>	現在の記事を印付きの記事の親記事にします (<code>gnus-summary-reparent-children</code>)。

以下の命令はスレッド移動命令です。これらはすべて数値接頭引数を受け付けます。

<i>T n</i>	
<i>C-M-f</i>	
<i>M-down</i>	次のスレッドに移動します (<code>gnus-summary-next-thread</code>)。
<i>T p</i>	
<i>C-M-b</i>	
<i>M-up</i>	前のスレッドに移動します (<code>gnus-summary-prev-thread</code>)。
<i>T d</i>	スレッドを下ります (<code>gnus-summary-down-thread</code>)。
<i>T u</i>	スレッドを登ります (<code>gnus-summary-up-thread</code>)。
<i>T o</i>	スレッドの頂上に移動します (<code>gnus-summary-top-thread</code>)。

スレッドを作成するときに表題を無視すると、当然ながらいくつかの違った表題があるスレッドが出来上がります。そして *T k* (`gnus-summary-kill-thread`) のような命令を発するときに、全体のスレッドを削除するのではなく、現在の記事と同じ表題を持つ部分だけを削除したいときがあるかもしれません。もしこの発想が良いと思うのであれば、`gnus-thread-operation-ignore-subject` をいじってみて下さい。これが `nil` でないと (それがデフォルトですが)、スレッドの命令を実行しているときに表題は無視されます。これが `nil` だったら、同じスレッドにある異なる表題を持つ記事は、そのとき行なう操作の対象に含まれません。この変数が `fuzzy` であると、大雑把な比較によって等しいと判定される表題を持つ記事だけが対象に含まれます (see Section 8.18 [Fuzzy Matching], page 269)。

3.10 並べ替え

概略でスレッドの表示を使っているのであれば、`gnus-thread-sort-functions` を設定することによってスレッドを並べ替えることができます。この変数の値は単独の関数、関数のリスト、または関数と (関数でないもの) の要素を含むリストであることができます。

デフォルトでは並べ替えは記事番号に基づいて行なわれます。すでに用意されている並べ替え述語関数は `gnus-thread-sort-by-number`, `gnus-thread-sort-by-author`, `gnus-thread-sort-by-recipient`, `gnus-thread-sort-by-subject`, `gnus-thread-sort-by-date`, `gnus-thread-sort-by-date-reverse`, `gnus-thread-sort-by-score`, `gnus-thread-sort-by-most-recent-number`, `gnus-thread-sort-by-most-recent-date`, `gnus-thread-sort-by-random` および `gnus-thread-sort-by-total-score` です。

それぞれの関数は二つのスレッドをとり、最初のスレッドがもう一方より先に並べ替えられるべきであれば `nil` でない値を返します。実際の並べ替えは、普通それぞれのスレッドの根本だけを調べることによって行なわれることに気を付けて下さい。

二つ以上の関数を使う場合、並べ替えの第一の鍵はリストの最後の関数でなければなりません。並べ替え関数のリストのなるべく先頭に、おそらく常に `gnus-thread-sort-by-number` を含めてお

くべきでしょう。これは、他の並べ替えの基準が等しいスレッドが、記事番号の登り順に表示されることを保証します。

スコアの逆順、表題、そして最後に番号、の順に並べ替えたいのであれば、次のようにできます:

```
(setq gnus-thread-sort-functions
      '(gnus-thread-sort-by-number
        gnus-thread-sort-by-subject
        (not gnus-thread-sort-by-total-score)))
```

最大のスコアを持つスレッドが、最初に概略バッファに表示されます。スレッドが同じスコアの場合は、英字順に並び替えられます。同じスコアと表題を持つスレッドは番号で並べ替えられ、(普通は) 記事が到着した順番になります。

スコア、到着の逆順に並べ替えたいのであれば、次のようにできます:

```
(setq gnus-thread-sort-functions
      '((not gnus-thread-sort-by-number)
        gnus-thread-sort-by-score))
```

変数 `gnus-thread-score-function` (デフォルトは `+`) に設定されている関数は、スレッドの総スコアを計算するために用いられます。役立つ関数は `max`, `min`, もしくは二乗、もしくはあなたの好奇心をくすぐるような何かでしょう。

何か変な理由でスレッド表示を使っていないのなら、変数 `gnus-article-sort-functions` をいじくする必要があります。これは `gnus-thread-sort-functions` と非常に似ていますが、記事の比較には少々違った関数を使います。使用可能な並べ替え述語関数は `gnus-article-sort-by-number`, `gnus-article-sort-by-author`, `gnus-article-sort-by-subject`, `gnus-article-sort-by-date`, `gnus-article-sort-by-random` および `gnus-article-sort-by-score` です。

スレッドを使っていない概略の表示を表題で並べ替えたいのであれば、次のようなことをすることができます:

```
(setq gnus-article-sort-functions
      '(gnus-article-sort-by-number
        gnus-article-sort-by-subject))
```

`gnus-parameters` を介することによって、グループによって異なる並べ替えを定義することができます。See Section 2.10 [Group Parameters], page 23.

3.11 非同期記事取得

遠くにある NNTP サーバーからニュースを取得していると、ネットワークの遅延が記事を読むことを嫌な仕事にしてしまうかもしれません。`n` を押してから次の記事が現れるまで、しばらく待たなければなりませんものね。どうして前の記事を読んでいる間に Gnus が先行して記事を取得してくれないのでしょうか? なぜできないんでしょう、本当に。

まず警告しておきましょう。非同期で記事を取得、特に Gnus がそれを行なう場合には、いくつかの落とし穴があります。

例えば、あなたは短い記事 1 を読んでいて、記事 2 はとても長くて、あなたはそれを読むことには興味が無いとしましょう。Gnus はこのことはわからないので、先行して記事 2 を取得します。あなたは記事 3 を読むことにしますが、Gnus は記事 2 を取得している最中なので、接続は封鎖されています。

この状況を避けるために、Gnus はサーバーに二つ (二まで数えて下さい) の接続を張ります。これはあまり良いことではないと考える人もいるでしょうが、私には実際の代替手段が見つからないのです。余分な接続を立ち上げるためにはいくばくかの時間がかかるので、Gnus の起動は遅くなります。

Gnus はあなたが読むであろう記事よりもたくさんの記事を取得します。これは記事の先行取得を使わないときよりも、あなたのマシンと NNTP サーバー間の接続にもっと負荷をかけることになるでしょう。サーバー自身にももっと負荷がかかるようになります—余分な記事の要求と、余分な接続によって。

はい、本当はこのようなことをすべきで無いことがこれで分かったでしょう... 本当にそうしたいと思わない限りは。

やり方です: `gnus-asynchronous` を `t` に設定して下さい。それ以外の諸々のことは自動的に行なわれます。

`gnus-use-article-prefetch` を設定することによって、どれくらいの記事を先に取得すべきかを操作することができます。これはデフォルトでは 30 で、グループの記事を読んでいるときに、バックエンドが次の 30 通の記事を先行取得するということです。この変数が `t` であると、バックエンドは取得できるすべての記事を際限なく先行取得しようとします。これが `nil` であると、先行取得は行なわれません。

おそらく先行取得をしたくない記事がいくつかあるでしょう—例えば既読記事です。変数 `gnus-async-prefetch-article-p` は記事が先に取得されるかどうかを制御します。この変数に設定される関数は、問題の記事を先行取得するのであれば `nil` でない値を返さなければなりません。デフォルトの関数は `gnus-async-read-p` で、これは既読記事には `nil` を返します。この関数は記事のデータ構造を唯一の引数として呼ばれます。

例えば、100 行よりも短い未読記事だけを先に取得したいのであれば、次のようにできます:

```
(defun my-async-short-unread-p (data)
  "Return non-nil for short, unread articles."
  (and (gnus-data-unread-p data)
        (< (mail-header-lines (gnus-data-header data))
            100)))
```

```
(setq gnus-async-prefetch-article-p 'my-async-short-unread-p)
```

これらの関数は何度も何度も呼ばれるので、Gnus を遅くしすぎないように、短く甘美であるのが好ましいです。このようなものをバイトコンパイル (`byte-compile`) するのは、おそらく良い着想でしょう。

記事は非同期バッファから遅かれ早かれ削除されなければなりません。 `gnus-prefetched-article-deletion-strategy` はいつ記事を削除するかを指定します。これは以下の要素を含むリストです:

`read` 記事が読まれたときに削除します。
`exit` グループを抜けたときに記事を削除します。

デフォルトの値は (`read exit`) です。

3.12 記事のキャッシュ

非常に 遅い NNTP 接続を使っているのならば、記事をキャッシュすることを考えても良いでしょう。それをする、それぞれの記事はあなたのホームディレクトリーの下にローカルに溜められます。もう感付いたかもしれませんが、これは `i` ノードを非常に速く食いつぶすだけでなく、巨大な ディスクスペースを食う可能性があります。それはあなたにウォッカの中で泳ぐようなめまいを起こさせるでしょう。

でも注意深く使われれば、それは記事を保存する、より楽な方法になり得ます。

キャッシュを実行させるには `gnus-use-cache` を `t` に設定して下さい。デフォルトでは、すべての可視または保留として印の付いている記事はローカルのキャッシュ (`gnus-cache-directory`) に複写されます。このキャッシュが平らな構造か階層的であるかは、通常通り、変数 `gnus-use-long-file-name` で制御されます。

可視記事が保留記事を再選択した場合は、サーバーの代わりにキャッシュから取得されます。キャッシュにある記事は期限切れ消去されないの、記事をそれらが属するところに居続けさせている間、それらを保存する方法としてこれは役立つかもしれません。保存したいすべての記事に保留の印を付けるだけで、後は心配無用です。

記事に既読の印が付いたときに、それはキャッシュから削除されるのでしょうか。

記事をキャッシュに入れたりキャッシュから削除することは、変数 `gnus-cache-enter-articles` および `gnus-cache-remove-articles` によって制御されます。これらは両方ともシンボルのリストです。前者はデフォルトでは (`ticked dormant`) で、可視記事と保留記事はキャッシュに入れます。後者はデフォルトでは (`read`) で、既読の印が付いた記事はキャッシュから削除されます。おそらくこれら二つのリストに含まれるシンボルは `ticked`, `dormant`, `unread` および `read` でしょう。

それでは、大規模な記事の取得と格納は、どこで関係してくるのでしょうか。 `gnus-jog-cache` 命令は、すべての購読グループに対して、すべての未読記事を要求し、スコアを付け、キャッシュに保存します。この命令をいつもいつもいつも使うのは、1) NNTP サーバーとの接続が本当に本当に遅くて、2) 本当に本当に巨大なディスクを持っているときだけにすべきです。これは真面目に言っています。ダウンロードされる記事の数を控える一つの方法は、欲しくない記事のスコアを低くして、それらに既読の印を付けることです。そうすれば、それらはこの命令ではダウンロードされません。

すべてのグループではキャッシュをしたくないというのは良くあることです。例えば `nnml` のメールがホームディレクトリーにあるのなら、それをホームディレクトリーの別の場所にキャッシュするのは意味がありません。二倍の容量を使う方が良いと思うのであれば。

キャッシュを制限するには、`gnus-cacheable-groups` を例えば `‘~nntp’` のようなキャッシュするグループの正規表現に設定するか、または正規表現 `gnus-uncacheable-groups` を例えば `‘~nnml’` に設定して下さい。両方の変数ともにデフォルトは `nil` です。もしグループが両方の変数に合致すると、そのグループはキャッシュされません。

キャッシュは、それがどの記事を含んでいるかの情報を、そのアクティブファイル (`gnus-cache-active-file`) に格納します。このファイル (もしくはキャッシュの他の部分) が何らかの理由でぐちゃぐちゃになってしまった場合、Gnus はものごとを正しくするための二つのコマンドを提供します。 `M-x gnus-cache-generate-nov-databases` はすべての NOV ファイルを (再) 作成し、 `M-x gnus-cache-generate-active` はアクティブファイルを (再) 作成します。

`gnus-cache-move-cache` コマンドは、すべての `gnus-cache-directory` をどこか別の場所に移動します。あなたはどこに移動させるかを尋ねられます。それってカッコいいでしょ？

3.13 永続記事

記事のキャッシュと近い関係にあるものに「永続記事」があります。実際それはキャッシュを見るための別の方法で、私に言わせればはるかに役に立ちます。

例えば、ニュースグループを読んでいて、永久に秘蔵しておく価値のある宝石に出会ったとしましょう。普通はそれをファイルに保存します (多くの保存命令の一つを使って)。問題は、単にあの、嫌なだけです。理想的には、記事はグループで見つけた場所に永遠に残っていることが好ましいでしょう。ニュースサーバーにおける期限切れ消去には影響されないで。

これが「永続記事」です—記事は削除されません。それは普通のキャッシュ命令を使って実装されていますが、永続記事の管理をするために二つの明示的な命令を使います:

- * 現在の記事を永続にします (gnus-cache-enter-article)。
- M-* 現在の記事を永続記事から取り除きます (gnus-cache-remove-articles)。これは普通は記事を削除します。

これらの命令は両方ともプロセス/接頭引数の習慣を理解します。

永続記事にだけ興味があるのなら、可視記事 (やその他のもの) がキャッシュに入るのを避けるために、gnus-use-cache を passive に設定するのが良いでしょう:

```
(setq gnus-use-cache 'passive)
```

3.14 記事のバックログ

回線が遅いために、キャッシュを使うという発想があまり魅力的ではないとき (実際そうなのですが)、「バックログ」に切り替えることによって状況を何とかすることができます。これはすでに読んだ記事を再取得しなくても良いように、すでに読んだ記事を Gnus が一時保存しておくところです。これはもちろん、あなたに最近読んだ記事を再び選択する癖があるときだけ役立ちます。絶対にそれをしない人にとっては、バックログを on にすることは Gnus を少し遅くし、メモリーの使用量をいくらか増やすだけのことです。

gnus-keep-backlog を数値 *n* に設定すると、Gnus は最大で *n* 個の古い記事を後の再取得のためにバッファに溜めておきます。この変数が nil ではなく、数値でもない場合、Gnus はすべての既読記事を蓄えます。それは Emacs が爆発するまで制限なく膨れ上がって、マシンがあなたもろとも落ちてしまうということです。私はみなさんがいつも注意を怠らないようにするために、ここに書き加えました。

デフォルト値は 20 です。

3.15 記事の保存

Gnus はたくさんの方法で記事を保存することができます。以下のものは非常に率直な方法 (すなわち記事が保存する前にほとんど何もなされない) で記事を保存するための説明です。異なる手続き (uudecode, unshar) のためには gnus-uu を使うのが良いでしょう (see Section 3.16 [Decoding Articles], page 77)。

ここに列挙されているコマンドは対象がファイルです。グループに保存したい場合は *B c* (gnus-summary-copy-article) コマンドを参照して下さい (see Section 3.25 [Mail Group Commands], page 103)。

gnus-save-all-headers が nil でないと、Gnus は記事を保存する前に不要なヘッダーを消去しません。

もし上記の変数が nil であると、正規表現 gnus-saved-headers に合致するすべてのヘッダーが残される一方、残りのものは保存する前に削除されます。

O o

o デフォルトの記事を保存する手段を用いて現在の記事を保存します (gnus-summary-save-article)。

O m 現在の記事を Unix メール (mbox) ファイルに保存します (gnus-summary-save-article-mail)。

O r 現在の記事を Rmail の様式で保存します (gnus-summary-save-article-rmail)。

- `O f` 現在の記事を普通のファイル (plain file) 様式で保存します (`gnus-summary-save-article-file`)。
- `O F` 現在の記事を普通のファイル様式で保存し、以前のファイルの内容を上書きします (`gnus-summary-write-article-file`)。
- `O b` 現在の記事の本文を普通のファイル様式で保存します (`gnus-summary-save-article-body-file`)。
- `O h` 現在の記事を `mh` のフォルダーの様式で保存します (`gnus-summary-save-article-folder`)。
- `O v` 現在の記事を `VM` フォルダーに保存します (`gnus-summary-save-article-vm`)。
- `O p`
| 現在の記事をパイプに保存します。うーんと、あのぉ、私が言おうとしていることは——現在の記事をプロセスにパイプするということです (`gnus-summary-pipe-output`)。シンボル接頭引数 (see Section 8.3 [Symbolic Prefixes], page 250) が与えられると、パイプへの出力に完全なヘッダーを含めます。
- `O P` 現在の記事を `muttprint` に保存します。つまり、外部プログラム `Muttprint` (<http://muttprint.sourceforge.net/>) を使って記事を印刷するということです。プログラム名と使用するオプションは、変数 `gnus-summary-muttprint-program` で指定されます。(`gnus-summary-muttprint`)。

すべてのこれらの命令はプロセス/接頭引数の習慣を使います (see Section 8.1 [Process/Prefix], page 249)。もしこれらの関数を使ってたくさんの記事を保存した場合、それぞれのすべての記事に対してファイル名の入力を要求されることに飽き飽きするでしょう。入力を求める動作は変数 `gnus-prompt-before-saving` によって制御されます。これはデフォルトでは `always` で、あなたが嫌な思いを味わっている、過剰な入力要求をします。代わりにこの変数を `t` に設定すると、保存するそれぞれの一連の記事に対して一回だけ入力を要求します。本当に `Gnus` にすべての判断を任せてしまいたいのであれば、この変数を `nil` にすることさえできます。そうすれば、記事を保存するためのファイルを促されることはありません。`Gnus` は単純にすべての記事をデフォルトのファイルに保存します。

`Gnus` を思い通りに動作させるために、変数 `gnus-default-article-saver` をカスタマイズすることができます。下の八つの既製の関数を使うことができ、また自分自身の関数を作ることもできます。

`gnus-summary-save-in-rmail`

これがデフォルトで、*Babyl* という様式です。変数 `gnus-rmail-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。デフォルトは `gnus-plain-save-name` です。

`gnus-summary-save-in-mail`

Unix メール (`mbox`) ファイルに保存します。変数 `gnus-mail-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。デフォルトは `gnus-plain-save-name` です。

`gnus-summary-save-in-file`

記事を通常のファイルの後に追加します。変数 `gnus-file-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。デフォルトは `gnus-numeric-save-name` です。

gnus-summary-write-to-file

記事をストレートに通常のファイルに保存します。そのファイルが存在していたら上書きされます。変数 `gnus-file-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。デフォルトは `gnus-numeric-save-name` です。

gnus-summary-save-body-in-file

記事の本文を通常のファイルの後に追加します。変数 `gnus-file-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。デフォルトは `gnus-numeric-save-name` です。

gnus-summary-write-body-to-file

記事の本文をストレートに通常のファイルに保存します。そのファイルが存在していたら上書きされます。変数 `gnus-file-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。デフォルトは `gnus-numeric-save-name` です。

gnus-summary-save-in-folder

MH ライブラリーの `rcvstore` を使って、記事を MH フォルダーに保存します。変数 `gnus-folder-save-name` に設定されている関数を、記事を保存するファイルの名前を取得するために使います。デフォルトは `gnus-folder-save-name` ですが、`gnus-Folder-save-name` も使うことができ、こちらは先頭が大文字、残りが小文字になった名前を作ります。

gnus-summary-save-in-vm

記事を VM フォルダーに保存します。この設定を使うためには VM メールリーダーが必要です。

それぞれの変数のシンボルは以下の属性 (property) を持つことができます:

:decode `nil` ではない値が設定されているとデコードした記事を保存します。`gnus-summary-save-in-file`、`gnus-summary-save-body-in-file`、`gnus-summary-write-to-file` および `gnus-summary-write-body-to-file` でだけ、これを設定する意義があります。

:function

記事をファイルに上書するのではなく、追加するための代わりの関数を指定します。これを設定すると、複数の記事を一度に保存するときに `gnus-prompt-before-saving` が `t` に束縛され、すべての記事が単一のファイルに保存されます。`gnus-summary-write-to-file` および `gnus-summary-write-body-to-file` でだけ、これを設定する意義があります。

:headers 保存されるヘッダーを指定する変数のシンボルをこれで設定します。省略された場合は `gnus-save-all-headers` と `gnus-saved-headers` が、どのヘッダーを保存するかを制御します。

これらのすべての関数は最後の一つを除いて、環境変数 `SAVEDIR` によって初期化される `gnus-article-save-directory` に記事を保存します。これはデフォルトでは `~/News/` です。

上で述べたように、記事を保存するためのファイルの適切な名前を見つけるために、それらは違った関数を用います。以下は名前を生成するために使うことができる関数のリストです:

gnus-Numeric-save-name

`~/News/Alt.andera-dworkin/45` のようなファイル名。

```
gnus-numeric-save-name
    '~/News/alt.andera-dworkin/45' のようなファイル名。

gnus-Plain-save-name
    '~/News/Alt.andera-dworkin' のようなファイル名。

gnus-plain-save-name
    '~/News/alt.andera-dworkin' のようなファイル名。

gnus-sender-save-name
    '~/News/larsi' のようなファイル名。
```

連想リスト `gnus-split-methods` に正規表現を放り込むことによって、Gnus に記事を保存する場所をほのめかすことができます。例えば Gnus に関連する記事を `'gnus-stuff'` ファイルに、VM に関連する記事を `'vm-stuff'` ファイルに保存したければ、この変数を以下のようにすれば良いでしょう:

```
((("^Subject:.*gnus\\|^Newsgroups:.*gnus" "gnus-stuff")
  ("^Subject:.*vm\\|^Xref:.*vm" "vm-stuff")
  (my-choosing-function "../other-dir/my-stuff")
  (equal gnus-newsgroup-name "mail.misc") "mail-stuff"))
```

これはそれぞれの要素が、二つの要素—「合致」と「ファイル」を持つリストであるリストであるということがわかります。合致は文字列（この場合は記事のヘッダーに合致する正規表現として使われます）、シンボル（グループ名を引数として、関数として呼ばれます）およびリスト（これは評価（eval）されます）のどれかであることができます。これらの動作の一つでも `nil` でない結果を返すと、入力を求めるときのデフォルトとして「ファイル」が使われます。加えて、呼ばれた関数か式が文字列か文字列のリストを返したときは、演算の結果自体が使われます。

基本的には、現在の記事を保存するのに使われる可能性のあるファイル名のリストを手に入れることとなります。（すべての『合致』が使われます。）そして、実際に名前として使いたいものの入力を促されますが、その際、この変数を適用して得られた結果が、ファイル名を補完するときの候補になります。

この変数はデフォルトでは `((gnus-article-archive-name))` で、これは Gnus が保存する記事の `Archive-name` 行を調べて、それをファイル名の候補として使います。

これはファイル名を多少きれいにする関数の例です。`'nnml:mail.whatever'` のようなメールグループがたくさんあるとすると、保存するためのファイル名を作る前にそれらのグループ名の最初の方を切り落とす必要があるかもしれません。次の物はまさにそれをします:

```
(defun my-save-name (group)
  (when (string-match "^nnml:mail." group)
    (substring group (match-end 0))))

(setq gnus-split-methods
      '((gnus-article-archive-name)
        (my-save-name)))
```

最後に、`gnus-use-long-file-name` という変数があります。これが `nil` であると、すべての上記の関数はグループ名のすべてのピリオド（`'.'`）をスラッシュ（`'/'`）で置き換えます—つまり、すべてのファイルを一番上のディレクトリーに置くのではなく、それらの関数が階層的なディレクトリーを生成するということです（`'~/News/alt.andrea-dworkin'` ではなく `'~/News/alt/andrea-dworkin'` のように）。たいていのシステムにおいて、この変数のデフォルトは `t` です。しかし、歴史的な理由によって Xenix と `usg-unix-v` マシンでは `nil` がデフォルトになります。

この関数は削除とスコアのファイル名にも影響します。この変数がリストで、そのリストが `not-score` という要素を含んでいると、長いファイル名はスコアファイルには使われません。そのリストが `not-save` という要素を含んでいると、保存するときに長いファイル名は使われません。また、そのリストが `not-kill` という要素を含んでいると、長いファイル名は削除ファイルには使われません。

記事をスプールのような階層に保存したい場合は、次のようにして下さい。

```
(setq gnus-use-long-file-name '(not-save)) ; 階層にする
(setq gnus-default-article-saver
      'gnus-summary-save-in-file) ; エンコードしない
```

そうしたならば、`o` で記事を保存するだけです。すると、階層を `nneething` 一時グループによって読むことができます—グループバッファで `GD` をタイプして、一番上のディレクトリー (`~/News/`) を引数として渡して下さい。

3.16 記事のデコード

ときどき利用者は何らかの方法でエンコードされた記事（もしくは一連の記事群）を投稿します。Gnus はそれらをデコードすることができます。

訳注: この章では、複数に分割して送信された一つの巨大な記事を、再び一つにまとめ上げてデコードする処理について説明しています。現在では、そのような分割送信をメールサーバーが受け付けない等の理由によって、ほとんど目にすることはありません。分割して送信しないことを確実にするには、以下の設定を行なって下さい (see section “メール変数” in *The Message Manual*):

```
(setq message-send-mail-partially-limit nil)
```

これらすべての関数はプロセス/接頭引数の習慣 (see Section 8.1 [Process/Prefix], page 249) を、『一つの記事』を『一つの群』と解釈する拡張をして、どの記事に操作をするかを見つけるために使います。Gnus は自分自身でどの記事がその群に属しているかを判断し、すべての記事をデコードして、その結果のファイルを展開/表示/保存することができます。

Gnus は以下の簡単な規則に則ってどの記事が群に属するのかを推測します: 表題は行の最後の二つの数字を除いて (ほとんど) 同じである必要があります。(空白は大体無視されますが。)

例えば: `'cat.gif (2/3)'` というような表題を選ぶと、Gnus は正規表現 `'^cat.gif ([0-9]+/[0-9]+).*'` に合致するすべての記事を見つけようとします。

`'cat.gif (2/3) Part 6 of a series'` のような標準でない表題はどの自動表示命令によっても適切に認識されないため、手で記事に `#` の印を付けなければなりません。

3.16.1 uuencode された記事

`X u` 現在の群を `uudecode` します (`gnus-uu-decode-uu`)。

`X U` 現在の群を `uudecode` して保存します (`gnus-uu-decode-uu-and-save`)。

`X v u` 現在の群を `uudecode` して、表示します (`gnus-uu-decode-uu-view`)。

`X v U` 現在の記事を `uudecode` して、表示して保存します (`gnus-uu-decode-uu-and-save-view`)。

これらはすべて、プロセス印が付けられた記事に対して反応するということを覚えておいて下さい。例えばニュースグループ全体をデコードして保存したいのであれば、例によって `MP a` (`gnus-uu-mark-all`) に続いて `X U` (`gnus-uu-decode-uu-and-save`) を実行して下さい。

このすべては、白日の下にいちいちキーを打っていた GNUS 4.1 のときの gnus-uu の動作とはまったく違ってしています。一般にこの版の gnus-uu は、何かの方法 (see Section 3.7.6 [Setting Process Marks], page 60) で記事に印を付け、それから *X u* を押すことを前提としています。

注意: 定数 `gnus-uu-notify-files` (値が `'[Cc][Ii][Nn][Dd][Yy][0-9]+\.\.(gif\|jpg\|)'` にハードコードされています) に合致する名前を持つ記事をデコードしようとすると、あなたが問題の記事を今まに見たことをバラすために、gnus-uu は自動的に `'comp.unix.wizards'` に記事を投稿します。この機能を使わないようにすることはできません (訳注: そんな Cindy Crawford 嬢の写真がニュースで大量に流れていた、まだ WWW がロクに普及していなかった時代の産物です)。

3.16.2 シェルアーカイブ

シェルアーカイブ (『*shar* ファイル』) はソースを配布するための人気のある方法でしたが、今日ではそんなに使われていません。とにかくこれらを扱うための命令があります:

- X s* 現在の群を解凍します (`gnus-uu-decode-unshar`)。
- X S* 現在の群を解凍して保存します (`gnus-uu-decode-unshar-and-save`)。
- X v s* 現在の群を解凍して表示します (`gnus-uu-decode-unshar-view`)。
- X v S* 現在の群を解凍し、表示して保存します (`gnus-uu-decode-unshar-and-save-view`)。

3.16.3 ポストスクリプトファイル

- X p* 現在のポストスクリプト群を展開します (`gnus-uu-decode-postscript`)。
- X P* 現在のポストスクリプト群を展開して保存します (`gnus-uu-decode-postscript-and-save`)。
- X v p* 現在のポストスクリプト群を表示します (`gnus-uu-decode-postscript-view`)。
- X v P* 現在のポストスクリプト群を表示して保存します (`gnus-uu-decode-postscript-and-save-view`)。

3.16.4 他のファイル

- X o* 現在の群を保存します (`gnus-uu-decode-save`)。
- X b* 現在の記事を `binhex` で解凍します (`gnus-uu-decode-binhex`)。これは実際には動作しません。

3.16.5 デコードのための変数

形容詞です。動詞ではありません。

3.16.5.1 規則変数

Gnus はファイルをどうやって表示するかを決めるために「規則変数」を使います。これらの変数はすべて以下のような様式です。

```
(list '(regexp1 command2)
      '(regexp2 command2)
      ...)
```


gnus-uu-user-view-rules

この変数はファイルを表示するときに最初に調べられます。例えば、もし ‘.au’ 音響ファイルを変換するために sox を使いたいときは、次のように設定することができます:

```
(setq gnus-uu-user-view-rules
      (list '("\\\\.au$" "sox %s -t .aiff > /dev/audio")))
```

gnus-uu-user-view-rules-end

この変数は Gnus が利用者とデフォルトの表示規則から合致するものを見つけることができなかったときに調べられます。

gnus-uu-user-archive-rules

この変数はアーカイブを展開するときにどの命令が使われるべきかを決めるために使うことができます。

3.16.5.2 他のデコードのための変数**gnus-uu-grabbed-file-functions**

これは関数のリストです。すぐにファイルを移動したり表示することを可能にし、何かができるようになる前にすべてのファイルがデコードされるのを待つ必要が無いように、それぞれのファイルのデコードに成功した直後にそれらの関数が呼ばれます。このリストに入れることができる既製の関数は以下の通りです:

gnus-uu-grab-view

ファイルを表示します。

gnus-uu-grab-move

ファイルを移動します (もし保存関数を使っているのであれば)。

gnus-uu-be-dangerous

デコードの最中に異常な状況が起こったときに何をするかを指定します。もし nil であると、できるだけ保守的になります。もし t であると、動作しないものは無視して、現存するファイルを上書きします。その他の場合は、それぞれのときに尋ねます。

gnus-uu-ignore-files-by-name

この正規表現に合致する名前のファイルは表示されません。

gnus-uu-ignore-files-by-type

この変数に合致する MIME の型を持つファイルは表示されません。Gnus はファイル名に基づいて型を推測していることに注意して下さい。gnus-uu は (まだ) MIME パッケージではないので、これは少々お行儀が悪いものです。

gnus-uu-tmp-dir

gnus-uu がその仕事をする場所です。

gnus-uu-do-not-unpack-archives

nil でないと、gnus-uu は表示するためのファイルを探すためにアーカイブの中身までは見ません。

gnus-uu-view-and-save

nil でないと、利用者はファイルを表示した後で常に保存するかどうかを尋ねられます。

gnus-uu-ignore-default-view-rules

nil でないと、gnus-uu はデフォルトの表示規則を無視します。

`gnus-uu-ignore-default-archive-rules`

`nil` でないと、`gnus-uu` はデフォルトのアーカイブ展開命令を無視します。

`gnus-uu-kill-carriage-return`

`nil` でないと、`gnus-uu` は記事からすべてのキャリッジリターンを取り去ります。

`gnus-uu-unmark-articles-not-decoded`

`nil` でないと、`gnus-uu` はデコードに失敗した記事に未読の印を付けます。

`gnus-uu-correct-stripped-uucode`

`nil` でないと、`gnus-uu` は後続の空白が削除されてしまっている uuencode されたファイルを修復しようと 試み ます。

`gnus-uu-pre-uudecode-hook`

メッセージを uudecode に送る前に実行されるフックです。

`gnus-uu-view-with-metamail`

`nil` でないと、`gnus-uu` は規則変数で定義された表示命令を無視して、ファイル名に基づいた MIME Content-Type をでっちあげます。その結果は表示のために metamail にかかけられます。

`gnus-uu-save-in-digest`

`nil` でないと、デコードせずに保存することを指示されたときに、`gnus-uu` は要約 (digest) を保存します。この変数が `nil` であると、`gnus-uu` は何も加工を施さずにすべてを一つのファイルに保存します。要約の作成は概ね RFC1153 に準拠していますが、意味のある目次を付ける簡単な方法が見つからなかったため、私はそれらを単に落としました。

3.16.5.3 uuencode と投稿

`gnus-uu-post-include-before-composing`

`nil` でないと、`gnus-uu` は記事を作成する前にエンコードするファイルを尋ねます。この変数が `t` であると、`C-c C-i` によってエンコードされたファイルを取り込むか、記事を投稿するときに取り込むかのどちらかを行うことができます。

`gnus-uu-post-length`

記事の最大の長さです。エンコードされたファイルは全体のファイルを投稿するのに必要な量のファイルに分割されます。

`gnus-uu-post-threaded`

`nil` でないと、`gnus-uu` はエンコードされたファイルをスレッドで投稿します。これはあまり賢い方法ではないかもしれませんが。というのは、今まで私が見た中で uuencode された記事を集めるのに、スレッドを追っていくことのできる他のデコーダーが存在しないからです。(えーと、私はそれをする一つのパッケージを見たことがあります—`gnus-uu` です。しかしどうも、それが数のうちに入るとは思えないのです...) デフォルトは `nil` です。

`gnus-uu-post-separate-description`

`nil` でないと、説明文は別の記事で投稿されます。最初の記事は普通 (0/x) のように番号が付けられます。もしこの変数が `nil` であると、利用者の書いた説明文は最初のファイルの始めに取り込まれ、(1/x) の番号が付けられます。デフォルトは `t` です。

3.16.6 ファイルの表示

デコードした後でファイルが何らかのアーカイブである場合、Gnus はアーカイブを展開しようと試み、アーカイブの中に表示できるファイルがあるかどうかを調べます。例えば、gzip された tar ファイル ‘pics.tar.gz’ があって、ファイル ‘pic1.jpg’ と ‘pic2.gif’ を含んでいる場合、Gnus は主ファイルを解凍して tar を展開し、それから二つの絵を表示します。この展開の過程は再帰的なので、アーカイブにアーカイブのアーカイブがあると、それはすべて展開されます。

最後に、Gnus は普通はそれぞれの抽出された記事ごとに「疑似記事」を概略バッファに挿入します。これらの『記事』に移動した場合は、実行する命令（普通は Gnus が提案をします）を入力するように促され、それからその命令が実行されます。

gnus-view-pseudo-asynchronously が nil であると、Emacs は先へ進む前に表示の終了を待ちます。

gnus-view-pseudos が automatic であると、Gnus は概略バッファに疑似記事を挿入せず、それらをすぐに表示します。この変数が not-confirm であると、利用者は表示が済む前に確認さえも求められません。

gnus-view-pseudos-separately が nil でないと、表示されるそれぞれのファイルにつき一つの疑似記事が作成されます。nil であると、同じ表示命令を使うすべての命令がその命令の引数のリストとして渡されます。

gnus-insert-pseudo-articles が nil でないと、デコードのときに疑似記事を挿入します。デフォルトでは t です。

さて、あなたはそんなふうに 仮想サーバー の 仮想グループ にある 疑似記事 を読むことになるわけです。どうしてもすべてが現実ではなくなってしまうのでしょうか？ どうしてこんなところに来ってしまったのでしょうか？

3.17 記事のトリートメント

この巨大な説明文書を読んできて、人々の著作を読むことがニュースリーダーの本当の目的だったことを、すっかり忘れてしまったかもしれません。記事を読むことです。残念ながら人々は書くことがとても苦手ですが、記事を読みやすくするための関数と変数は山のようにあります。

3.17.1 記事のハイライト

記事バッファをフルーツサラダのように、いや総天然色のフルーツサラダのようにしたくありませんか。

W H a 現在の記事をもっとハイライトします。この関数は、ヘッダー、引用文、署名をハイライトし、本文とヘッダーにボタンを加えます。

W H h ヘッダーをハイライトします (gnus-article-highlight-headers)。ハイライトは変数 gnus-header-face-alist に従って行なわれ、それはそれぞれの要素が（正規表現 名前 内容）という様式のリストです。正規表現 はヘッダーに合致する正規表現、名前 はヘッダーの名前をハイライトするのに使われるフェース (see Section 8.6 [Faces and Fonts], page 258)、内容 はヘッダーの値をハイライトするフェースです。最初に合致したものが使われます。正規表現 の先頭に ‘^’ を付けてはいけないことに注意して下さい—Gnus がそれを付け加えます。

W H c 引用された文をハイライトします (gnus-article-highlight-citation)。引用文のハイライトをカスタマイズする変数は次の通りです：

- `gnus-cite-parse-max-size`
 記事の大きさがこの変数 (デフォルトでは 25000) のバイト数より大きい記事は、引用文のハイライトが行なわれません。
- `gnus-cite-max-prefix`
 引用符の最大の長さです (デフォルトでは 20 です)。
- `gnus-cite-face-list`
 引用文をハイライトするために使われるフェースのリストです (see Section 8.6 [Faces and Fonts], page 258)。同じメッセージの中に複数の記事からの引用があると、Gnus はそれぞれの記事からの引用をそれ用のフェースで表示しようとします。これにより、誰が何を書いたかが分かりやすくなるでしょう。
- `gnus-supercite-regexp`
 普通の Supercite 著者行に合致する正規表現です。
- `gnus-supercite-secondary-regexp`
 引き裂かれた Supercite 著者行に合致する正規表現です。
- `gnus-cite-minimum-match-count`
 それが引用文であると判定する前に調べなければならない引用符の最小の数です。
- `gnus-cite-attribution-prefix`
 著者行の始まりに合致する正規表現です。
- `gnus-cite-attribution-suffix`
 著者行の終りに合致する正規表現です。
- `gnus-cite-attribution-face`
 著者行に使われるフェースです。その著者が書いた文の引用のためのフェースと融合されます。
- `gnus-cite-ignore-quoted-from`
 非-nil だったら、‘>From’ で始まる行で引用文のハイライトは行なわれません。それらの行は、エンベロープ From 行と混同しないように、MTA がクオートした可能性があります。デフォルト値は t です。
- W H S* 署名 (signature) をハイライトします (`gnus-article-highlight-signature`)。 `gnus-signature-separator` (see Section 3.17.10 [Article Signature], page 93) の後のすべてのものは署名であると解釈され、`gnus-signature-face` でハイライトされます。それはデフォルトでは *italic* です。

記事を自動的にハイライトする方法については Section 4.3 [Customizing Articles], page 118 を参照して下さい。

3.17.2 記事中の文の強調表示

(訳注: Fontisize == Fontify + Emphasize)

人々はよくニュースの記事で ‘_これ_’ や ‘*これ*’ または ‘/これ/’ のようなものを使って単語を強調します。Gnus は記事を *W e* 命令 (`gnus-article-emphasize`) にかけることによって素敵に見えるようにできます。

強調がどのように処理されるかは変数 `gnus-emphasis-alist` によって制御されます。これは連想リストで、最初の要素は合致すべき正規表現です。二番目の要素は、正規表現の中のどのグループが強調語全体を見つけるために使われるかを示す数値です。三番目は正規表現のどのグループが表示されハイライトされるかを定める数値です。(この二つのグループの間にあるテキストは隠されます。) 四番目はハイライトさせるためのフェースです。

```
(setq gnus-emphasis-alist
      '(("_\\(\\w+\\)" 0 1 gnus-emphasis-underline)
        ("\\*\\(\\w+\\)\\*" 0 1 gnus-emphasis-bold)))
```

(訳注: 上記の変数の値は、デフォルトのままにしておくのが無難です。)

デフォルトでは七つの規則があり、それらは以下のフェースを用います:

`gnus-emphasis-bold`, `gnus-emphasis-italic`, `gnus-emphasis-underline`,
`gnus-emphasis-bold-italic`, `gnus-emphasis-underline-italic`, `gnus-emphasis-underline-bold`, `gnus-emphasis-underline-bold-italic`.

これらのフェースを変更したいのであれば、`M-x customize` か `copy-face` を使うことができます。例えば `gnus-emphasis-italic` が代わりに赤のフェースを使うようにしたいのならば、次のようにすれば良いでしょう:

```
(copy-face 'red 'gnus-emphasis-italic)
```

任意の語を強調表示させたいときは、`gnus-group-highlight-words-alist` 変数を使うことができます。これは `gnus-emphasis-alist` と同じ構文を使います。`highlight-words` グループパラメーター (see Section 2.10 [Group Parameters], page 23) を使うこともできます。

記事を自動的に強調表示させるやり方については Section 4.3 [Customizing Articles], page 118 を参照して下さい。

3.17.3 記事を隠す

と言うよりはむしろ、記事の中にある特定のものを隠すことです。たいていの記事には、普通はありすぎるくらいのごみがあります。

W W a 記事バッファでたくさんのを隠します (`gnus-article-hide`)。特にこの関数はヘッダー、PGP、引用文、それに署名を隠します。

W W h ヘッダーを隠します (`gnus-article-hide-headers`)。See Section 4.1 [Hiding Headers], page 115.

W W b あまり興味の持てないヘッダーを隠します (`gnus-article-hide-boring-headers`)。See Section 4.1 [Hiding Headers], page 115.

W W s 署名を隠します (`gnus-article-hide-signature`)。See Section 3.17.10 [Article Signature], page 93.

W W l `gnus-list-identifiers` で指定されているメーリングリストの標識を削除します。これらはいくつかのメーリングリストのサーバーがすべての Subject ヘッダーの最初に付ける文字列、例えば `'[zebra 4711]'` のようなものです。文字列の初めにある `'Re:'` は、削除を行なう前に跳び越されます。`gnus-list-identifiers` に `\\(..\\)` を含めてはいけません。

`gnus-list-identifiers`

表題から削除されるべきメーリングリストの標識に合致する正規表現です。これは正規表現のリストであることもできます。

W W P 不要な PEM (privacy enhanced messages (プライバシー拡張メッセージ)) の部分を隠します (gnus-article-hide-pem)。

W W B banner グループパラメーターで指定されたバナーを取り除きます (gnus-article-strip-banner)。これは主に、いくつかのメーリングリストや司会者付きのグループがすべての記事に追加する、鬱陶しいバナーと/ もしくは署名を隠すために使用されます。この関数を使う方法は banner グループパラメーター (see Section 2.10 [Group Parameters], page 23) をバナーを取り除きたいグループに追加することです。パラメーターは、消去されるテキストに合致する正規表現として解釈される文字列か、(最後の) 署名を消去するためのシンボル signature、または gnus-article-banner-alist の正規表現に対応した他のシンボルのいずれかであることができます。

グループにかかわらず、記事の送信者が gnus-article-address-banner-alist で設定されている特定のメールアドレスを持っているときだけ、広告のようなものを隠すことができます。

gnus-article-address-banner-alist

メールアドレスとバナーの連想リストです。それぞれの要素は (address . banner) の形式を持ち、ここで address は From ヘッダーにあるメールアドレスに合致する正規表現です。また、banner はシンボル signature、gnus-article-banner-alist の要素、正規表現または nil のうちのひとつです。address が著者のメールアドレスに合致すると、広告のようなものを消します。例えば、送信者が 'hail@yoo-hoo.co.jp' というメールアドレスを持っていて、彼が送信するすべての記事に 'Do You Yoo-hoo!?' のようなものがある場合、以下の要素でそれらを消すことができます。

```
("@yoo-hoo\\.co\\.jp\\'" . "\\n_+\\nDo You Yoo-hoo!\\|?\\n.*\\n.*\\n")
```

W W c 引用文を隠します (gnus-article-hide-citation)。隠蔽をカスタマイズするいくつかの変数は:

gnus-cited-opened-text-button-line-format

gnus-cited-closed-text-button-line-format

Gnus はどこの引用文が隠されているかを示すためにボタンを付け加え、文章の隠蔽を切り替えられるようにします。この変数の様式は、以下のフォーマットのような変数によって指定されます (see Section 8.4 [Formatting Variables], page 250)。次の指定が有効です:

- 'b' 隠された文の最初のポイントです。
- 'e' 隠された文の最後のポイントです。
- 'l' 隠されたリージョンの文字の数です。
- 'n' 隠された文の行の数です。

gnus-cited-lines-visible

隠さずに表示しておく、引用文の先頭からの行数です。これは、隠さずに表示する先頭からの行と、隠さずに表示する末尾からの行の、それぞれの数の cons セルであることもできます。

W W C-c 以下の二つの変数に依存して、引用文を隠します (gnus-article-hide-citation-maybe):

`gnus-cite-hide-percentage`

引用文の割合のパーセンテージが、この変数 (デフォルトは 50) より大きかったら、引用文を隠します。

`gnus-cite-hide-absolute`

隠される前に、引用文は少なくともこの長さ (デフォルトは 10) でなければなりません。

W W C 根本でない記事の引用文を隠します (`gnus-article-hide-citation-in-followups`)。これは対話的命令としてはあまり役に立たないかもしれませんが、自動的に実行させるには手軽な関数でしょう (see Section 4.3 [Customizing Articles], page 118)。

これらのすべての『隠蔽』命令は切り替え命令ですが、これらの命令に負の接頭引数を与えると、それらは前に隠されていたものを表示します。正の接頭引数を与えれば、それらは常に隠します。

引用文をカスタマイズするためのさらなる変数について、Section 3.17.1 [Article Highlighting], page 81 も参照して下さい。

自動的に記事の要素を隠すための方法は Section 4.3 [Customizing Articles], page 118 を参照して下さい。

3.17.4 記事の洗濯

私たちはこれをもっともな理由の下で『記事の洗濯』(article washing) と呼んでいます。A キーは使われていたので、代わりに *W* キーを使う必要がありました。

「洗濯」は『何かの何かを何か別のものに変換する』と定義されますが、普通はもっと良く見える何かに落ち着きます。もっときれいになります、たぶん。

Gnus が記事を表示するデフォルトのやり方を変えたいときは Section 4.3 [Customizing Articles], page 118 を参照して下さい。

C-u g これは洗濯ではなくて、その逆です。これをタイプすると、ディスクやサーバーにあるがままの記事が見えます。

g 現在の記事の再表示を強制します (`gnus-summary-show-article`)。これもまた本当の洗濯ではありません。これをタイプすると、以前に適用された対話的な洗濯機能はご破算にされ、すべてのデフォルトのトリートメントを施された記事が表示されます (see Section 4.3 [Customizing Articles], page 118)。

W l ページの区切りを現在の記事から取り除きます (`gnus-summary-stop-page-breaking`)。ページの区切りに付いては Section 4.5 [Misc Article], page 121 を参照して下さい。

W r 記事バッファでカエサル変換 (Caesar rotate, rot13) を行ないます (`gnus-summary-caesar-message`)。カエサル変換か rot13 を用いて読むことを指定する、判読不可能な記事です (典型的には攻撃的な冗談などです。)

普通は“rot13”と呼ばれています。それはアルファベットの位置が 13 個回転するからです。例えば、‘B’ (2 番目の文字) ‘O’ (15 番目の文字)。これは時々『カエサル変換』と呼ばれることもあります。というのは、カエサルがこの形式の、えーと、いささか貧弱な暗号化を採用したという噂があるからです。

W m 記事バッファをモールスでデコードします (`gnus-summary-morse-message`)。

- W i** 現在の記事にある IDNA エンコードされたドメイン名をデコードします。IDNA エンコードされたドメイン名は 'xn--bar' のように見えます。これを実行した後で文字列がデコードされないままだったら、おそらくそれは不正な IDNA 文字列でしょう ('xn--bar' は不正です)。このコマンドを動かすためには、GNU Libidn (<http://www.gnu.org/software/libidn/>) をインストールしていなければなりません。
- W t**
- t** 記事バッファにすべてのヘッダーを表示するかどうかを切り替えます (gnus-summary-toggle-header)。
- W v** 記事バッファにすべてのヘッダーを永続的に表示するかどうかを切り替えます (gnus-summary-verbose-headers)。
- W o** オーバーストライクを処理します (gnus-article-treat-overstrike)。
訳注: 以下のような重ね打ちを指示する文字列を bold や underline で表示します。
'B^HBo^Hol^Hld^Hd', 'U^H_n^H_d^H_e^H_r^H_l^H_i^H_n^H_e^H_'
- W d** gnus-article-dumbquotes-map に応じて、マ ソ sm*rtq**t*s を処理します。この関数は文字が sm*rtq**t* かどうかを推測するので、対話的にのみ使用されるべきであることに注意して下さい。
Sm*rtq**t*s はもっと多くの引用文字を提供するために、マ ソ が勝手に文字マップを拡張したものです。もし、アポストロフィ (') や引用記号 (") などがあるべきところに \222 や \264 のようなものが見えてしまったら、洗濯してみてください。
- W Y f** いかれた Outlook (Express) の記事を完全に醜くなくすること (訳注: de-ugly-fy) (sm*rtq**t*s を処理、行の折り返しを解除、著者行の修復と引用文の整頓) をします。(gnus-article-outlook-deuglify-article)。
- W Y u** 折り返された引用行のように見える行の折り返しを解きます。折り返しが解かれた行の最小および最大の長さを表す gnus-outlook-deuglify-unwrap-min および gnus-outlook-deuglify-unwrap-max を調整することによって、どんな行の折り返しが解かれるかを制御することができます。(gnus-article-outlook-unwrap-lines)。
- W Y a** 壊れた著者行を修復します。
(gnus-article-outlook-repair-attribution)。
- W Y c** 壊れた引用文を、テキストを整理し直すことによって修復します。(gnus-article-outlook-rearrange-citation)。
- W w** 行を折り返します (gnus-article-fill-cited-articles)。
折り返す幅を指定するために、命令に数値接頭引数を与えることができます。
- W Q** 長い行を折り返します (gnus-article-fill-mode-lines)。
- W C** それぞれの文の最初の語を大文字にします (gnus-article-capitalize-sentences)。
- W c** CRLF の組 (すなわち、行の最後の '^M') を LF に変換します (これは DOS の行末の世話をします)。そうしてから残りの CR を LF に変換します (これは MAC の行末の世話をします) (gnus-article-remove-cr)。
Quoted-printable を処理します (gnus-article-de-quoted-unreadable)。
Quoted-Printable は 非-ASCII (すなわち 8-bit) の記事を送るときに使われる一般

的な MIME エンコーディングです。それは概して ‘dø,Aijø,A‘ vu’ のようなものを ‘d=E9j=E0 vu’ に見せるので、とても読み辛くなります。問題の記事が、そのエンコーディングが行なわれたことを示す Content-Transfer-Encoding ヘッダーを持っていれば、通常それは Gnus によって自動的に行なわれることに注意して下さい。接頭引数が与えられると、文字セットが尋ねられます。

W 6 Base64 をデコードします (gnus-article-de-base64-unreadable)。Base64 は非-ASCII (すなわち 8-bit) の記事を送るときに使われる、一般的な MIME エンコーディングです。問題の記事が、そのエンコーディングが行なわれたことを示す Content-Transfer-Encoding ヘッダーを持っていれば、通常それは Gnus によって自動的に行なわれることに注意して下さい。接頭引数が与えられると、文字セットが尋ねられます。

W Z HZ または HZP を処理します。HZ (または HZP) は中国語の記事を伝送するときに使われる一般的な符号です。これは ‘~{<:Ky2;S{#,NpJ)16HK!#~}’ のような典型的な文字列を作ります。

W A ANSI SGR シーケンスを overlay または extent に変換します (gnus-article-treat-ansi-sequences)。ANSI シーケンスは中国語のニュースグループで強調表示に使われています。

W u URL に含まれる改行を削除します。いくつかのメイラーは、行を短くするために出ていくメールに改行を挿入しますが、これは長い URL を複数の行に分割してしまいます。改行を削除することによって、それらの URL を復旧させます (gnus-article-unsplit-urls)。

W h HTML を処理します。当該メッセージが HTML であることを示す Content-Type ヘッダーを持っていたならば、それは Gnus によって自動的に行なわれることに注意して下さい。

接頭引数が与えられると、文字セットを尋ねられます。それがもし数値だったら、gnus-summary-show-article-charset-alist (see Section 3.4 [Paging the Article], page 50) で定義されている文字セットが使われます。(訳注: 実質的には「文字セット」ではなくて coding-system です。)

デフォルトでは HTML の変換に mm-text-html-renderer (see section “表示のカスタマイズ” in *The Emacs MIME Manual*) で設定された関数を使いますが、変数 gnus-article-wash-function が設定されていると、記事の洗濯ではそれが優先されます。使うことができる、あらかじめ用意された関数は以下の通りです:

w3 Emacs/W3 を使います。

w3m emacs-w3m (<http://emacs-w3m.namazu.org/>) を使います。

w3m-standalone
w3m (<http://w3m.sourceforge.net/>) を使います。

links Links (<http://links.sf.net/>) を使います。

lynx Lynx (<http://lynx.isc.org/>) を使います。

html2text
html2text (シンプルな HTML コンバーターで、Gnus に含まれています) を使います。

訳注: 例えば `mm-text-html-renderer` を `w3m` などに設定してあって正しく動作するのならば、`gnus-article-wash-function` はデフォルトの `nil` のままにしておいて構いません。

- W b* クリックできるボタンを記事に加えます (`gnus-article-add-buttons`)。See Section 3.17.6 [Article Buttons], page 89.
- W B* クリックできるボタンを記事のヘッダーに加えます (`gnus-article-add-buttons-to-head`)。
- W p* 署名付きコントロールメッセージの認証を行ないます (`gnus-article-verify-x-pgp-sig`)。newgroup や checkgroups といったコントロールメッセージは、通常そのニュースグループ階層のメインテイナーによって署名されています。認証を行なうためには、メインテイナーの PGP 公開鍵をあなたのキーリングに追加しなければなりません。¹
- W s* 署名されたメッセージ (PGP, PGP/MIME または S/MIME によって) を検証します (`gnus-summary-force-verify-and-decrypt`)。See Section 3.30 [Security], page 111.
- W a* 記事の本文の先頭から X-No-Archive ヘッダーのようなヘッダーを取り除きます (`gnus-article-strip-headers-in-body`)。
- W E l* 記事の先頭にあるすべての空白行を取り除きます (`gnus-article-strip-leading-blank-lines`)。
- W E m* すべての空白行を空行で置き換えてから、すべての複数の空行を一つの空行で置き換えます (`gnus-article-strip-multiple-blank-lines`)。
- W E t* 記事の最後にあるすべての空白行を取り除きます (`gnus-article-remove-trailing-blank-lines`)。
- W E a* 上の三つの命令をすべて実行します (`gnus-article-strip-blank-lines`)。
- W E A* すべての空白行を取り除きます (`gnus-article-strip-all-blank-lines`)。
- W E s* 記事の本文のすべての行頭にあるすべての空白を取り除きます (`gnus-article-strip-leading-space`)。
- W E e* 記事の本文のすべての行末にあるすべての空白を取り除きます (`gnus-article-strip-trailing-space`)。

自動的に記事の洗濯を行なわせる方法は Section 4.3 [Customizing Articles], page 118 を参照して下さい (訳注: 実は多くの洗濯がデフォルトで自動的に行なわれます)。

3.17.5 記事ヘッダー

これらのコマンドは記事ヘッダーをいろいろに変形させます。

- W G u* 折り返されたヘッダー行を一行にします (`gnus-article-treat-unfold-headers`)。
- W G n* Newsgroups と Followup-To ヘッダーを折り返します (`gnus-article-treat-fold-newsgroups`)。

¹ 多くのニュースグループ階層のメインテイナーの PGP の鍵は <ftp://ftp.isc.org/pub/pgpcontrol/README.html> から入手することができます。

- W G f* すべてのメッセージヘッダーを折り返します (gnus-article-treat-fold-headers)。
- W E w* すべてのヘッダーから余分な空白を取り除きます (gnus-article-remove-leading-whitespace)。

3.17.6 記事のボタン

人々はよく記事の中に他の資料を参照するための案内を入れることがあります。それらの参照への案内の上で *RET* を打つか、マウスの真中のボタンを使ったときに、彼らが話題にしているのが何であれ、最小限の曖昧さで Gnus が取得することができれば素敵でしょう。

特定の標準的な参照に、Gnus はデフォルトで「ボタン」を付けます: ちゃんとした URL、メールアドレス、Message-ID、Info へのリンク、man ページ、それに関連する Emacs または Gnus の参考文献です。これは二つの変数によって制御されていて、その一つは記事の本文を扱い、もう一つは記事のヘッダーを扱います。

gnus-button-alist

それぞれの要素が次のような様式を持つ連想リストです:

(*regexp button-par use-p function data-par*)

regexp この正規表現 (大文字と小文字は区別されません) に合致するすべてのテキストは、外部への参照であるとみなされます。これは埋め込まれた URL に合致する典型的な正規表現です: '<URL:\\([^\n\r>]*\\)>'。これはまた正規表現の値を持つ変数であってもよく、有用な変数として *gnus-button-url-regexp* および *gnus-button-mid-or-mail-regexp* があります。

button-par

Gnus は合致したもののどの部分がハイライトされるのかを知らなければなりません。これは正規表現のどの副表現がハイライトされるかを指定する番号です。すべてをハイライトしたいのなら、ここで 0 を使って下さい。

use-p

この式は 評価 され、結果が *nil* でなかったら、これは合致であるとみなされます。これは間違った合致を避けるために特別な選別をしたいときに役に立ちます。ここではしばしば *gnus-button-*-level* のような名前の変数が使われますが、See Section 3.17.7 [Article Button Levels], page 91, 他のどんな形式でも使うことができます。

function

この関数が、このボタンをクリックしたときに呼ばれます。

data-par

button-par のように、これは部分表現の番号ですが、これは合致のどの部分が *function* にデータとして送られるかを指定します。

したがって URL をボタンにする完全な要素は、こうなります。

("<URL:\\([^\n\r>]*\\)>" 0 t *gnus-button-url* 1)

gnus-header-button-alist

これは他の連想リストと同じようなものですが、記事のヘッダーだけに適用されることが、それぞれの項目がどのヘッダーにボタンを付けるかを指示するための追加の要素を持っていることが異なります:

(*header regexp button-par use-p function data-par*)

header は正規表現です。

3.17.6.1 関連する変数と関数

`gnus-button-*-level`

Section 3.17.7 [Article Button Levels], page 91 を参照して下さい。

`gnus-button-url-regexp`

埋め込まれた URL に合致する正規表現です。上述の変数のデフォルトの値で使われます。

`gnus-button-man-handler`

Man ページの表示に使う関数です。少なくとも一つの引数として Man ページの名前の文字列を受け付けなければなりません。

`gnus-button-mid-or-mail-regexp`

Message-ID かメールアドレスに合致する正規表現です。

`gnus-button-prefer-mid-or-mail`

この変数は 'foo123@bar.invalid' のような文字列のボタンが押されたときに、何を行なうかを決める変数です。このような文字列は Message-ID かメールアドレスのいずれかです。もし mid か mail というシンボルのうちの一つだったら、Gnus は常にそれぞれ文字列が Message-ID またはメールアドレスであると仮定します。この変数が ask というシンボルに設定されると、Gnus はいつも利用者が何をしたいかを尋ねます。それが関数だった場合、たった一つの文字列を引数として呼ばれます。その関数は mid、mail、invalid または ask を返さなければなりません。デフォルト値は関数 `gnus-button-mid-or-mail-heuristic` です。

`gnus-button-mid-or-mail-heuristic`

その引数が Message-ID かメールアドレスであるかを推定する関数です。Message-ID だったら mid を、メールアドレスだったら mail を、不確かだったら ask を、そして無効な文字列だったら invalid を返します。

`gnus-button-mid-or-mail-heuristic-alist`

関数 `gnus-button-mid-or-mail-heuristic` で使われる (RATE . REGEXP) 対の連想リストです。

`gnus-button-ctan-handler`

CTAN リンクの表示に使う関数です。URL 名の文字列を、引数として一つ受け付けなければなりません。

`gnus-ctan-url`

`gnus-button-ctan-handler` で使われる CTAN (Comprehensive TeX Archive Network) アーカイブのディレクトリーです。

`gnus-article-button-face`

ボタンに使われるフェースです。

`gnus-article-mouse-face`

マウスのカーソルがボタンの上にあるときに使われるフェースです。

記事に自動的にボタンを付ける方法は、Section 4.3 [Customizing Articles], page 118 を参照して下さい。

3.17.7 Article button levels

変数 `gnus-button-*-level` の値が高いほど、より多くのボタンが現れます。レベルがゼロだったらボタンは表示されません。デフォルト値 (それは 5) では、とてもたくさんのボタンをすでにしているはずで、高いレベルではより多くのボタンを見ることになりますが、多くの要らないものも現れるかもしれません。それらを避けるために、特定のグループに対して変数 `gnus-button-*-level` を設定しても良いでしょう (see Section 2.10 [Group Parameters], page 23)。`gnus-parameters` 変数の例です:

```
;; いくつかのグループで gnus-button-*-level を増やす:
(setq gnus-parameters
      '(("<\\(emacs\\|gnus\\)\\>" (gnus-button-emacs-level 10))
        ("<\\(unix\\)" (gnus-button-man-level 10))
        ("<\\(tex\\)" (gnus-button-tex-level 10))))
```

`gnus-button-browse-level`

Message-ID、メールアドレスおよびニュースの URL を参照する案内の表示を制御します。関連する変数と関数には `gnus-button-url-regexp`、`browse-url` および `browse-url-browser-function` があります。

`gnus-button-emacs-level`

Emacs または Gnus への参照の表示を制御します。関連する関数は、`gnus-button-handle-custom`、`gnus-button-handle-describe-function`、`gnus-button-handle-describe-variable`、`gnus-button-handle-symbol`、`gnus-button-handle-describe-key`、`gnus-button-handle-apropos`、`gnus-button-handle-apropos-command`、`gnus-button-handle-apropos-variable`、`gnus-button-handle-apropos-documentation` および `gnus-button-handle-library` です。

`gnus-button-man-level`

(Unix の) man ページへの参照の表示を制御します。`gnus-button-man-handler` を見て下さい。

`gnus-button-message-level`

Message-ID、メールアドレスおよびニュースの URL の表示を制御します。関連する変数と関数には `gnus-button-mid-or-mail-regexp`、`gnus-button-prefer-mid-or-mail`、`gnus-button-mid-or-mail-heuristic` および `gnus-button-mid-or-mail-heuristic-alist` があります。

`gnus-button-tex-level`

T_EX または LaTeX への参照、例えば CTAN の URL の表示を制御します。変数 `gnus-ctan-url`、`gnus-button-ctan-handler`、`gnus-button-ctan-directory-regexp` および `gnus-button-handle-ctan-bogus-regexp` を見て下さい。

3.17.8 記事の日付

日付は聞いたことの無い何か辺鄙なタイムゾーンで作成されていることが良くあるので、記事が送られたときに何時だったかを知ることができるのはとても良いことです。

`W T u` UT (別名 GMT, ZULU) で日付を表示します (`gnus-article-date-ut`)。

`W T i` 日付を国際的な形式、ISO 8601 で表示します (`gnus-article-date-iso8601`)。

- W T l** 日付をローカル・タイムゾーンで表示します (gnus-article-date-local)。
- W T p** 日付を英語で楽に発音できる形式で表示します (gnus-article-date-english)。
- W T s** 日付を利用者定義の様式を使って表示します (gnus-article-date-user)。その様式は変数 `gnus-article-time-format` で指定される、`format-time-string` に渡される文字列です。指定することができる様式の一覧は、変数の説明文を見て下さい。
- W T e** 記事が投稿されてから今までどれくらいの時間が経過したかを表示します (gnus-article-date-lapsed)。こんなふうに。
- X-Sent: 6 weeks, 4 days, 1 hour, 3 minutes, 8 seconds ago
- gnus-article-date-lapsed-new-header の値で、このヘッダーを既存の Date の下に追加するか、置き替えるかを指定します。
- Gnus でメールを読むことの利点は、それが単純なバグを素晴らしい不条理に置き換えることです。
- この行が連続して更新されるようにしたいのであれば、
- (gnus-start-date-timer)
- を ‘~/gnus.el’ ファイルに入れるか、それを何かのフックで実行するようにすることができます。タイマーを止めたい場合は、gnus-stop-date-timer 命令を使って下さい。
- W T o** 本来の日付を表示します (gnus-article-date-original)。これはあなたが普段は他の変換関数を使っていて、それが完全に間違っているのではないかと心配になったときに役に立ちます。例えば、記事が 1854 年に投稿されたと主張したとしましょう。しかし、そのようなことは 完全に 不可能です。私が信用できませんか? *くすくす*

好みの書式で自動的に日付を表示する方法は Section 4.3 [Customizing Articles], page 118 を参照して下さい。

3.17.9 Article Display

これらのコマンドは、いろんな取るに足らないギミック (gimmicks) の表示を、それらをサポートしている Emacs の記事バッファに追加します。

X-Face ヘッダーは小さな白黒画像で、メッセージヘッダーから持ってきます (see Section 8.17.1 [X-Face], page 265)。

Face ヘッダーは小さなカラー画像で、メッセージヘッダーから持ってきます (see Section 8.17.2 [Face], page 267)。

スマイリーは、人々がメッセージに散らかしたがる小さな ‘:-)’ シンボルです。

一方 Picon はあなた自身のシステムに依存し、Gnus はヘッダーに合致するあなたの持ち物を探してみます (see Section 8.17.4 [Picons], page 268)。

これらすべての機能はトグルです。もしすでにそれらが存在していたならば、それらは削除されます。

- W D x** X-Face を From ヘッダーに表示します (gnus-article-display-x-face)。
- W D d** Face を From ヘッダーに表示します (gnus-article-display-face)。
- W D s** スマイリーを表示します (gnus-treat-smiley)。

- W D f* From ヘッダーを Picon 化します (gnus-treat-from-picon)。
- W D m* すべてのメールヘッダー (すなわち Cc、To) を Picon 化します (gnus-treat-mail-picon)。
- W D n* すべてのニュースヘッダー (すなわち Newsgroups と Followup-To) を Picon 化します (gnus-treat-newsgroups-picon)。
- W D D* 記事バッファからすべての画像を削除します (gnus-article-remove-images)。

3.17.10 記事の署名

それぞれの記事は二つの部分に分けられます—ヘッダーと本文です。本文は署名部分と文章部分に分けることができます。どれが署名とみなされるかを決める変数は `gnus-signature-separator` です。これは普通は son-of-RFC 1036 で規定されている標準の ‘^-- \$’ です。しかし、多くの人が標準ではない署名セパレーターを使うので、この変数は一つ一つ試される、正規表現のリストであることもできます。(探索は本文の最後から始めへとなされます。) よくありそうな値は:

```
(setq gnus-signature-separator
      '(("^-- $"           ; 標準
        "^-- *$"         ; 普通の崩し方
        "^-----*$"     ; 多くの人は長ーい横棒の
                        ; 行を使います。みっともない!
        "^ *-----*$"   ; 二倍みっともない!
        "^-----*$"     ; 下線も人気があります
        "^=====*$")) ; 邪道!
```

あなたが寛容であればあるほど、間違った結果を得ることになるでしょう。

`gnus-signature-limit` は記事を表示するときにどれが署名とみなされるかへの制限を提供します。

1. これが整数であれば、署名はこの整数より (文字数で) 長くなってはいけません。
2. これが浮動小数点数であれば、署名はその数値より (行数で) 長くなってはいけません。
3. これが関数であれば、その関数は引数なしで呼ばれ、それが `nil` を返せば、そのバッファには署名がありません。
4. これが文字列であれば、それは正規表現として使われます。もしそれが合致すれば、当のその文字列は署名ではありません。

この変数は、要素が上に列挙された型のリストであることもできます。例です:

```
(setq gnus-signature-limit
      '(200.0 "^----*Forwarded article"))
```

これは署名セパレーターの後に 200 を超える行があるか、セパレーターの後のテキストが正規表現 ‘^----*Forwarded article’ に合致すれば、結局それは署名ではないということです。

3.17.11 記事いろいろ

- A t* 記事のある言語から別のものへ変換します (gnus-article-babel)。

3.18 MIME コマンド

以下のコマンドはすべて数値接頭引数を理解します。例えば `3 K v` は「三番目の MIME パートを表示する」という意味です。

<code>b</code>	
<code>K v</code>	MIME パートを表示します。
<code>K o</code>	MIME パートを保存します。
<code>K O</code>	ファイル名の入力を求めてから MIME パートを保存し、それを記事から取り除きます。取り除かれた MIME オブジェクトは <code>message/external-body</code> 型の MIME 形式として参照されるようになります。
<code>K r</code>	MIME パートを外部にある本体で置き換えます。
<code>K d</code>	MIME パートを削除して、削除したことの案内を追加します。
<code>K c</code>	MIME パートをコピーします。
<code>K e</code>	MIME パートを外部コマンドで表示します。
<code>K i</code>	MIME パートをバッファ内に表示します。
<code>K </code>	MIME パートを外部コマンドにパイプします。

以降の MIME コマンドの残りは、数値接頭引数を同じやり方では使いません:

<code>K b</code>	すべての MIME パートの先頭にボタンを付加します。埋め込まれたパートをセーブ (または他の動作を実行) しようとするときに、たいてい便利です。
<code>K m</code>	ときたま、ヘッダーが無かったり間違ったヘッダーを持つマルチパートのメッセージが送信されてきます。このコマンドは、それらのメッセージがより快適に表示されるように「修復」を試みます (<code>gnus-summary-repair-multipart</code>)。
<code>X m</code>	MIME タイプに合致するすべてのパートを、ディレクトリーにセーブします (<code>gnus-summary-save-parts</code>)。プロセス/接頭引数の習慣を理解します (see Section 8.1 [Process/Prefix], page 249)。
<code>M-t</code>	記事バッファにボタンを表示するかしないかを切り替ええます (<code>gnus-summary-toggle-display-buttonized</code>)。
<code>W M w</code>	記事ヘッダーにある RFC 2047 でエンコードされた語をデコードします (<code>gnus-article-decode-mime-words</code>)。
<code>W M c</code>	エンコードされた記事の本文を、文字セットでデコードします (<code>gnus-article-decode-charset</code>)。 このコマンドは、文字セットを決めるために Content-Type ヘッダーを調べます。記事にそんなヘッダーが無い場合でも、接頭引数を与えることによって、デコードするための文字セットを入力することは可能です。ある共通のエンコーディングを使って (でも MIME ヘッダーは含めずに) 人々が記事を投稿する地域的なグループでは、charset グループ/トピック・パラメーターに必要な文字セットを設定すれば良いでしょう (see Section 2.10 [Group Parameters], page 23)。
<code>W M v</code>	現在の記事にある、すべての MIME パートを表示します (<code>gnus-mime-view-all-parts</code>)。

関連する変数:

gnus-ignored-mime-types

これは正規表現のリストで、これに含まれている正規表現に合致する MIME タイプは、Gnus によって完全に無視されます。デフォルト値は `nil` です。

すべての Vcard を無視させるには、こんなふうにして下さい:

```
(setq gnus-ignored-mime-types
      '("text/x-vcard"))
```

gnus-article-loose-mime

非-`nil` だったら、Gnus は記事を MIME メッセージとして解読する前に、`'MIME-Version'` があることを必要としません。これは、ある壊れたメール・ユーザー・エージェントからのメッセージを読むときに役立ちます。デフォルトは `nil` です。

gnus-article-emulate-mime

MIME ではない別のエンコーディングの手法があります。最も一般的なのは `'uuencode'` ですが、`yEncode` も普及してきています。この変数が非-`nil` になっていると、Gnus はメッセージの本文にそれらのエンコーディングが見つかるかどうかを調べ、もしあったならば、それらを Gnus の MIME 機構で処理します。デフォルトは `t` です。デコードできるのは単一の `yEnc` でエンコードされたパートだけです。Gnus はエンコードについてはサポートしません。

gnus-unbuttonized-mime-types

これは正規表現のリストで、これに含まれている正規表現に合致する MIME タイプには、ボタンが付加されません。ただし、それらが表示されないか、`gnus-buttonized-mime-types` 変数の方が優先される場合を除いて、ですが。デフォルト値は `(".*/.*)` です。この変数は `gnus-inhibit-mime-unbuttonizing` が `nil` のときだけ使われます。

gnus-buttonized-mime-types

これは正規表現のリストで、これに含まれている正規表現に合致する MIME タイプには、それらが表示されない場合を除いて、ボタンが付加されます。この変数は `gnus-unbuttonized-mime-types` よりも優先されます。デフォルト値は `nil` です。この変数は `gnus-inhibit-mime-unbuttonizing` が `nil` のときだけ使われます。

例えば、セキュリティのボタンだけを表示して、他のボタンを表示しないようにするには、この変数を `("multipart/signed")` に設定して、`gnus-unbuttonized-mime-types` はデフォルト値のままにしておいて下さい。

また、このリストに `"multipart/alternative"` を加えることによって、そういうメールに含まれている二つのメディア・タイプのうちの一つを選ぶことができる、ラジオボタンを表示させることができます。`mm-discouraged-alternatives` も参照して下さい (see section “表示のカスタマイズ” in *The Emacs MIME Manual*)。

gnus-inhibit-mime-unbuttonizing

これが非-`nil` だと、すべての MIME パートにボタンを付加します。デフォルト値は `nil` です。

gnus-article-mime-part-function

それぞれの MIME パートに対して、この関数が MIME ハンドル (訳注: パートのタイプや内容物を表現するために、Gnus の内部で使われるデータの構造体) を引数にして呼ばれます。この関数は、利用者が記事から情報を集め (例えば Vcard の情報を `bbdb`

のデータベースに加え) たり、パートに基づいて何かを起動 (例えば、すべての jpeg をあるディレクトリーにセーブ) するために使われることが意図されています。

後者を行なう関数の例です:

```
(defun my-save-all-jpeg-parts (handle)
  (when (equal (car (mm-handle-type handle)) "image/jpeg")
    (with-temp-buffer
      (insert (mm-get-part handle))
      (write-region (point-min) (point-max)
                    (read-file-name "Save jpeg to: "))))
  (setq gnus-article-mime-part-function
        'my-save-all-jpeg-parts))
```

gnus-mime-multipart-functions

MIME マルチパートの型と、それらを扱う関数の連想リストです。

gnus-mime-display-multipart-alternative-as-mixed

"multipart/alternative" のパートを "multipart/mixed" であるものとして表示します。

gnus-mime-display-multipart-related-as-mixed

"multipart/related" のパートを "multipart/mixed" であるものとして表示します。

もし "text/html" を表示するのが気に入らないのなら、`mm-discouraged-alternatives` を参照して下さい。ただし (それで "text/html" を表示しないように設定して、かつ) この変数が `nil` だと、"multipart/related" パートの中にある画像や他の資料を見逃してしまうかもしれません。See section “表示のカスタマイズ” in *The Emacs MIME Manual*.

gnus-mime-display-multipart-as-mixed

"multipart" のパートを "multipart/mixed" であるものとして表示します。もし `t` だと、`gnus-mime-display-multipart-alternative-as-mixed` および `gnus-mime-display-multipart-related-as-mixed` が `nil` であっても、この設定の方が優先されます。

mm-file-name-rewrite-functions

MIME パートのファイル名を書き換えるために使われる関数のリストです。それぞれの関数はファイル名を受け取って、ファイル名を返します。

出来合いの関数は

`mm-file-name-delete-whitespace`, `mm-file-name-trim-whitespace`, `mm-file-name-collapse-whitespace` および `mm-file-name-replace-whitespace` です。最後のものはファイル名に含まれるそれぞれの空白文字を、変数 `mm-file-name-replace-whitespace` の値で置き換えます。デフォルト値は "_" (単一の下線) です。

標準の関数である `capitalize`, `downcase`, `upcase` および `upcase-initials` も、役に立つでしょう。

ファイル名に含まれる空白文字が害をもたらすことは、みんなが知っています。ただし、気にかけない人たちを除いて、ですが。そんな蒙昧の人たちから、たくさんの添付ファイルを受け取るのであれば、こんなものを ‘`~/gnus.el`’ ファイルに追加することによって、安寧な生活を送ることができるでしょう。

```
(setq mm-file-name-rewrite-functions
      '(mm-file-name-trim-whitespace
        mm-file-name-collapse-whitespace
        mm-file-name-replace-whitespace))
```

3.19 文字セット

人々はいろいろな文字セットを使いますが、私たちは彼らが何の文字セットを使っているかを教えてくれる MIME を持っています。あるいはもっと正確に言えば、持っていたらいいなあと思います。多くの人たちが MIME を利用しないか理解しないニュースリーダーとメイラーを使って、何の文字セットを使うかを言わずに、単にメッセージを送出するのですが、これを少しばかり救済するために、いくつかの地域的なニュース階層には、何の文字セットがデフォルトであるかを宣言する取り決めがあります。例えば ‘fj’ 階層では iso-2022-jp を使っています。

この知識は `gnus-group-charset-alist` 変数にエンコードされています。これは正規表現 (グループのフルネームに合致した最初の項目を使います) と、それらのグループを講読するときに使われるデフォルトの文字セットの、連想リストです。

加えて、人々のいくらかは MIME を意識していると自称 (*soi-disant*) しているくせに、実はそうではないエージェントを使っています。それらは、実際にはメッセージが koi-8 なのに iso-8859-1 だと、陽気にメッセージに刻印するのです。ここでは救済のために `gnus-newsgroup-ignored-charsets` 変数を使うことができます。そのリストに連ねられた文字セットは無視されます。この変数は、グループパラメーター (see Section 2.10 [Group Parameters], page 23) を使って、グループ毎に設定することができます。デフォルト値は (`unknown-8bit x-unknown`) で、それはいくつかのエージェントが内蔵し、主張する値を含んでいます。

投稿する場合に、MIME でエンコードしてはいけない文字セットを判定するために、`gnus-group-posting-charset-alist` が使われます。例えばいくつかの階層では、`quoted-printable` でヘッダーをエンコードすることは嫌われます。

この変数は正規表現と、投稿に際してエンコードしなくても良いことを許された (またはエンコードすることが嫌われる) 文字セットの連想リストです。それぞれの要素は (*test header body-list*) の形式であり、それらは次の意味を持ちます。

- | | |
|------------------|---|
| <i>test</i> | Newsgroups ヘッダーに合致する正規表現、または変数シンボルのどちらかです。後者の場合は、その値を調べた結果が非-nil だったら、その要素が採用されることになります。 |
| <i>header</i> | ヘッダーをエンコードしなくても良い文字セットです (nil は、すべての文字セットをエンコードすることを意味します)。 |
| <i>body-list</i> | “ Content-Transfer-Encoding: 8bit ”でもって本文をエンコードしても良い (または <code>quoted-printable</code> や <code>base64</code> でエンコードすることが嫌われる) 文字セットのリスト、または特別な値の一つである nil (常に <code>quoted-printable</code> でエンコードする)、または <code>t</code> (常に “ Content-Transfer-Encoding: 8bit ”を使う) です。 |

メッセージを送信するときに何の文字セットが使われるかを制御する付加的な変数については、See section “エンコーディングのカスタマイズ” in *The Emacs MIME Manual*, を参照して下さい (訳注: 特に日本語のメッセージの文字セットについては、例えば変数 `mm-coding-system-priorities` を参照して下さい)。

Gnus 固有ではないけれど、役に立つかもしれない文字セットに関する他の秘訣:

もし、同一の Emacs の文字セットをエンコードする MIME の文字セットが複数あるのならば、以下の宣言を使うことによって、使う文字セットを選択することができます:

```
(put-charset-property 'cyrillic-iso8859-5
                      'preferred-coding-system 'koi8-r)
```

これは、ロシア語がデフォルトの iso-8859-5 MIME 文字セットの代わりに、koi8-r でエンコードされることを意味します。

メッセージを koi8-u で読みたいのであれば、以下のように騙すことができます。

```
(define-coding-system-alias 'koi8-u 'koi8-r)
```

これは、ほとんど正しいことをするでしょう。

そして最後に、windows-1251 のような文字セットを読むには、次のように宣言すれば良いでしょう (訳注: Emacs の版によっては、windows-1251 が最初から実装されています)。

```
(codepage-setup 1251)
(define-coding-system-alias 'windows-1251 'cp1251)
```

3.20 記事命令

A P 記事バッファのポストスクリプト (PostScript) イメージを作成して印刷します (gnus-summary-print-article)。gnus-ps-print-hook がバッファを印刷する直前に実行されます。他に Muttprint を使って印刷することもできます (see Section 3.15 [Saving Articles], page 73)。

3.21 概略の並べ替え

私はどうしてあなたがそうしたいのかはわからないのですが、それでもあなたはたくさんの方で概略バッファを並べ替えることができます。

C-c C-s C-n

記事番号によって並べ替えます (gnus-summary-sort-by-number)。

C-c C-s C-a

著者によって並べ替えます (gnus-summary-sort-by-author)。

C-c C-s C-t

受信者によって並べ替えます (gnus-summary-sort-by-recipient)。

C-c C-s C-s

表題によって並べ替えます (gnus-summary-sort-by-subject)。

C-c C-s C-d

日付によって並べ替えます (gnus-summary-sort-by-date)。

C-c C-s C-l

行数によって並べ替えます (gnus-summary-sort-by-lines)。

C-c C-s C-c

記事の長さ (文字数) で並べ替えます (gnus-summary-sort-by-chars)。

C-c C-s C-i

スコアによって並べ替えます (gnus-summary-sort-by-score)。

C-c C-s C-r

ランダムに並べ替えます (gnus-summary-sort-by-random)。

`C-c C-s C-o`

デフォルトの方法で並べ替えます (`gnus-summary-sort-by-original`)。

これらの関数はスレッドを使っているときと使っていないときの両方で動作します。後者では、すべての概略行が一行一行並べ替えられます。前者では根本だけに基づいて並べ替えられ、それはあなたが求めていることとは異なっているかもしれません。スレッドを使うかどうかを切り替えるには `T T` を打って下さい (see Section 3.9.2 [Thread Commands], page 68)。

3.22 親記事を探す

`^` 現在の記事の親記事を読みたいのに、それが概略バッファに表示されていないか、おそらくそれは可能でしょう。というのは、現在のグループが NNTP で取得されていて、親がまだ期限切れ消去されていない上、現在の記事の References がぶち壊されていないければ、ただ `^` か `A r` を押せば良いだけです (`gnus-summary-refer-parent-article`)。すべてがうまくいけば、親記事を取得できるでしょう。もし親記事がすでに概略バッファに表示されているのであれば、ポイントがその記事に移動するでしょう。正の数値接頭引数を与えられると、その数の祖先たちを遡って取得します。負の数値接頭引数を与えられた場合は、その数の世代だけ前の祖先の記事のみを取得します。ですから `3 ^` とすれば、Gnus は現在の記事の親と祖父母と曾祖父母を取得します。`-3 ^` とすれば、Gnus は現在の記事の曾祖父母だけを取得します。

`A R` (概略) 記事の References 欄にあるすべての記事を取得します (`gnus-summary-refer-references`)。

`A T` (概略) 現在の記事があるスレッドの、全部の記事を表示します (`gnus-summary-refer-thread`)。この命令は動作するために現在のグループのすべてのヘッダーを取得しなければならないので、普通は少し時間がかかります。これをしばしば行なうのであれば、`gnus-fetch-old-headers` を `invisible` に設定することを考えたほうが良いでしょう (see Section 3.9.1.2 [Filling In Threads], page 66)。これは普通は視覚的な効果はありませんが、この命令の動作をかなり速くします。もちろんグループに入るのはいくらか遅くなりますが。

変数 `gnus-refer-thread-limit` はこの命令を実行するときどのくらい古い (すなわち、現在のグループで最初に表示されたものよりも前の記事の) ヘッダーを取得するかを指定します。デフォルトは 200 です。もし `t` であれば、取得可能なすべてのヘッダーを取得します。`A T` 命令に数値接頭引数を与えると、代わりにそれが使われます。

`M-^` (概略)

どのグループに属しているかに関わらず、任意の記事を Gnus に要求することができます。`M-^` (`gnus-summary-refer-article`) は Message-ID、つまりあの長くてなかなか読むことのできない `<38o6up$6f2@hymir.ifi.uio.no>` のようなものをあなたに尋ねます。あなたはすべてを正確に打ち込まなければなりません。残念ながら、あいまいな検索はできないのです。

Gnus はすでに取得してあるヘッダーたちの中で Message-ID を探しますが、見つからなかったら `gnus-refer-article-method` に設定されているすべての選択方法を試してみます。

もしあなたの読んでいるグループが Message-ID での取得があまり良くできないようなバックエンド (`nnsPOOL` など) であるのなら、`gnus-refer-article-method` を NNTP の選択方法に設定すれば良いでしょう。おそらく、あなたが問い合わせる NNTP サーバーがあなたの読んでいるスプールを更新していると最も良いでしょう。しかし、それはどうしても必要なわけではありません。

それは選択方法のリストのみならず、現在の選択方法を意味する特別なシンボル `current` であることもできます。Gnus は合うものを発見するまでそれらすべての方法を試します。

これは現在の選択方法を試して、それが失敗した場合には Google に訊く設定の例です:

```
(setq gnus-refer-article-method
      '(current
        (nnweb "google" (nnweb-type google))))
```

ほとんどのメールバックエンドは Message-ID での取得が可能です。あまり優雅な方法でやっているわけではありません。nnmbox, nnbabyl, nnmaildir および nnml がどのグループからでも記事を検索できるのに対して、nnfolder と nnimap は現在のグループに投稿された記事しか探すことができません。(その他のものは時間がかかりすぎます。) nnmh ではまったく不可能です。

3.23 代替手段

ニュースを読む方法の好みは人それぞれです。これは Gnus なのですから、概略バッファのためのマイナーモードに少しばかり選択肢を設けます。

3.23.1 選んで読む

いくつかのニュースリーダー (nn や、ええと VM/CMS の Netnews など) は二段階の講読インターフェイスを使います。利用者はまず概略バッファで読みたい記事に印を付けます。それから、記事バッファだけを表示して記事を読みます。

Gnus はこれをするための概略バッファマイナーモードを提供します—`gnus-pick-mode` です。これは、基本的には簡単に印を付けられるように少数のプロセス印命令を一個のキーだけで済む命令にして、概略バッファへ切り替えるための追加の命令を一つ提供します。

訳注: Pick マイナーモードを有効にするには、以下のフックを使って下さい:

```
(add-hook 'gnus-summary-mode-hook 'gnus-pick-mode)
```

そうせずに、概略バッファに入ってから `M-x gnus-pick-mode` を実行しても、うまくいかないようです。

これらが `pick mode` で使うことができるキーです:

- . 現在の行の記事かスレッドを選択します (`gnus-pickd-article-or-thread`)。変数 `gnus-thread-hide-subtree` が非-nil だったら、このキーがスレッドの最初の記事で使われるとスレッド全体を選択します。そうでなければ、その記事だけを選択します。もし数値接頭引数を与えられると、その番号のスレッドか記事に移動して、それを選択します。(普通は行番号が概略行の最初に表示されます。)
- SPACE 概略バッファを一ページ次にスクロールします (`gnus-pick-next-page`)。もしバッファの最後であれば、選択した記事を読み始めます。
- u スレッドか記事を未選択にします (`gnus-pick-unmark-article-or-thread`)。変数 `gnus-thread-hide-subtree` が非-nil だったら、このキーがスレッドの最初で使われるとそのスレッドを未選択にします。そうでなければ、その記事だけを未選択にします。その行にあるスレッドか記事を未選択にするために、このキーに数値接頭引数を与えることができます。
- RET 選択された記事を読み始めます (`gnus-pick-start-reading`)。接頭引数を与えられると、最初にすべての未選択記事に既読の印を付けます。 `gnus-pick-display-summary` が nil でないと、概略バッファは読んでいる間も表示されます。

すべての普通の概略モード命令は `pick-mode` でも使用可能ですが、`u` は例外です。それでも、同じ関数 `gnus-summary-tick-article-forward` に割り当てられている `!` を使うことができます。

これが良さそうと思ったら、次のようにして下さい:

```
(add-hook 'gnus-summary-mode-hook 'gnus-pick-mode)
```

`gnus-pick-minor-mode-hook` は `pick` マイナーモードのバッファで実行されます。

`gnus-mark-unpicked-articles-as-read` が非-`nil` だったら、選択されなかったすべての記事に既読の印を付けます。デフォルトは `nil` です。

`pick` モードでの概略行の様式は標準の様式とは少し違います。それぞれの行の最初に行数が表示されます。`pick` モードの行の様式は変数 `gnus-summary-pick-line-format` で制御されます (see Section 8.4 [Formatting Variables], page 250)。これは `gnus-summary-line-format` と同じ様式指定を受け付けます (see Section 3.1.1 [Summary Buffer Lines], page 43)。

3.23.2 バイナリーグループ

多くの時間をバイナリーグループで過ごしているのなら、いつも `X u, n, RET` を叩くのが嫌になっているでしょう。`M-x gnus-binary-mode` は、単に記事を普通の方法で表示する代わりに、記事を選択するための普通の Gnus の関数を、一連の記事を `uudecode` してその結果を表示するように変更する、概略バッファのためのマイナーモードです。

現実には、このモードにしたときに、実際に記事を見るための唯一の命令が `g` です (`gnus-binary-show-article`)。

`gnus-binary-mode-hook` がバイナリーマイナーモードのバッファで呼ばれます。

3.24 木表示

もし普通の Gnus の概略表示を好きでないならば、`gnus-use-trees` を `t` に設定してみると良いかもしれません。これは (デフォルトで) 追加の「木バッファ」(`tree buffer`) を作成します。木バッファではすべての概略モード命令を実行することができます。

もちろん、木表示をカスタマイズする変数が少しあります:

`gnus-tree-mode-hook`

すべての木モードのバッファで実行されるフックです。

`gnus-tree-mode-line-format`

木モードのバッファにおけるモード行のためのフォーマット文字列です (see Section 8.4.2 [Mode Line Formatting], page 251)。デフォルトは `'Gnus: %b %S %Z'` です。使用可能な指定は Section 3.1.3 [Summary Buffer Mode Line], page 47 を参照して下さい。

`gnus-selected-tree-face`

木バッファで選択された記事をハイライトするために使われるフェースです。デフォルトでは `modeline` です。

`gnus-tree-line-format`

木の節のためのフォーマット文字列です。でもこれは少し誤った名称です—それは行ではなく、ただ節を定義するだけです。デフォルトの値は `'(%[%3,3n%])'` で、それは投稿者の名前の最初の三文字を表示します。すべての節が同じ長さであることが重要なので、`'%4,4n'` のような指定を 使わなければなりません。

有効な指定は:

‘n’ 投稿者の名前。
 ‘f’ From 欄。
 ‘N’ 記事の番号。
 ‘[’ 開き括弧。
 ‘]’ 閉じ括弧。
 ‘s’ 表題。

See Section 8.4 [Formatting Variables], page 250.

表示に関連した変数は:

`gnus-tree-brackets`

これは『本当の』記事と『まばら』な記事に違いを付けるために使われます。様式は

```
((本当の開 . 本当の閉)
 (まばら開 . まばら閉)
 (偽の開 . 偽の閉))
```

となっていて、デフォルトは ((?[. ?]) (?[. ?]) ({[. ?]) (<[. ?>)) です。

`gnus-tree-parent-child-edges`

これは親の節を子に接続するために使われる文字を含むリストです。デフォルトは (?- ?\ ?|) です。

`gnus-tree-minimize-window`

もしこの変数が `nil` でないと、他の Gnus ウィンドウがもっと場所を取るように Gnus は木バッファをできるだけ小さくします。もしこの変数が数値であると、木バッファの高さはその数値より大きくなることはありません。デフォルトは `t` です。フレームでいくつかのウィンドウが横に並んで表示されていて、木バッファがそのうちの一つである場合、木ウィンドウを最小化することはその隣に表示されているすべてのウィンドウの大きさをも変更することに注意して下さい。

以下のフックを追加して、いつでも木ウィンドウを最小化するようにしても良いでしょう。

```
(add-hook 'gnus-configure-windows-hook
          'gnus-tree-perhaps-minimize)
```

`gnus-generate-tree-function`

実際にスレッドの木を作成する関数です。二つの定義済みの関数 `gnus-generate-horizontal-tree` および `gnus-generate-vertical-tree` (これがデフォルトです) が利用可能です。

水平木バッファ (horizontal tree buffer) の例です:

```
{***}-(***)-[odd]-[Gun]
|      \[Jan]
|      \[odd]-[Eri]
|      \(***)-[Eri]
|      \[odd]-[Paa]
\[Bjo]
\[Gun]
```


\[Gun]-[Jor]

同じスレッドが垂直木バッファ (vertical tree buffer) で表示されたものです:

```
{***}
|-----\-----\-----\
(**)                                [Bjo] [Gun] [Gun]
|--\-----\-----\
[odd] [Jan] [odd] (**)                                [Jor]
|         |         |--\
[Gun]     [Eri] [Eri] [odd]
|
[Paa]
```

もし水平木を使っているのなら、概略バッファで木を隣り合わせで表示できれば嬉しいでしょう。次のようなものを ‘~/gnus.el’ ファイルに加えることができます:

```
(setq gnus-use-trees t
      gnus-generate-tree-function 'gnus-generate-horizontal-tree
      gnus-tree-minimize-window nil)
(gnus-add-configuration
 '(article
   (vertical 1.0
    (horizontal 0.25
     (summary 0.75 point)
     (tree 1.0))
    (article 1.0))))
```

See Section 8.5 [Window Layout], page 254.

3.25 メールグループ命令

いくつかの命令はメールグループでのみ意味を持ちます。これらの命令が現在のグループで有効でないなら、それらは大騒ぎをしてあなたに知らせるでしょう。

これらすべての命令は (期限切れ消去と編集命令は除く) プロセス/接頭引数の習慣を使います (see Section 8.1 [Process/Prefix], page 249)。

- B e* 現在のグループのすべての期限切れ消去可能な記事について、期限切れ消去の処理 (gnus-summary-expire-articles) を行ないます。これは、そのグループにしばらく存在していた期限切れ消去可能なすべての記事を消去するということです。 (see Section 6.3.9 [Expiring Mail], page 163)。
- B C-M-e* グループのすべての期限切れ消去可能な記事を削除します (gnus-summay-expire-articles-now)。これは、現在のグループにある すべての期限切れ消去可能な記事が、永遠に空の大きな ‘/dev/null’ へ消え去るということです。
- B DEL* メール記事を削除します。これは『あなたのディスクから永久に削除して二度と戻らない』の意味の『削除』です。注意して使って下さい (gnus-summary-delete-article)。
- B m* あるメールグループから別のメールグループへ記事を移動します (gnus-summary-move-article)。gnus-preserve-marks の値が nil でなければ (それがデフォルト)、印は保存されます。

- B c* あるグループ (メールグループや他のもの) からメールグループに記事をコピーします (gnus-summary-copy-article)。gnus-preserve-marks の値が nil でなければ (それがデフォルト)、印は保存されます。
- B B* 現在の記事を他のグループにクロスポストします (gnus-summary-crosspost-article)。これは他のグループの記事の新しい複製を作成し、記事の Xref 欄も適切に更新されます。
- B i* 任意のファイルを現在のメールグループに取り込みます (gnus-summary-import-article)。あなたはファイル名と、From 欄と Subject 欄の入力を促されます。
- B I* 空の記事を現在のメールグループに作ります (gnus-summary-create-article)。From ヘッダーと Subject ヘッダーの内容を尋ねられます。
- B r* メール記事をスプールし直します (gnus-summary-move-article)。gnus-summary-respool-default-method が再スプールするときのデフォルトの選択方法として使用されます。この変数はデフォルトでは nil で、その場合は現在のグループの選択方法が代わりに使われます。gnus-preserve-marks の値が nil でなければ (それがデフォルト)、印は保存されます。
- 訳注: 「スプールし直す」というのはメールの分割 (Section 6.3.3 [Splitting Mail], page 147 または Section 6.3.6 [Fancy Mail Splitting], page 157) の規則に基づいて、メールを適切なグループに入れ直すことです。そのグループに間違っ入ってしまったメールを、分割の規則を修正した後で、正しいグループに移動させる場合などに使います。この章の *B q* と *B t* も見て下さい。
- B w*
e 現在の記事を編集します (gnus-summary-edit-article)。編集を終了して変更を固定するには *C-c C-c* (gnus-summary-edit-article-done) を打ちます。もし *C-c C-c* 命令に接頭引数を与えると、Gnus は記事を再ハイライトしません。
- 訳注: 変更しないで編集を終るには、*C-c C-k* をタイプして下さい。
- B q* 記事を再スプールするときは、再スプールをする前にどのグループに記事が移るかを知りたいでしょう。この命令でそれがわかります (gnus-summary-respool-query)。
- B t* 同様に、この命令は再スプールするときに使われるすべての特級分割方式を、もしあれば表示します (gnus-summary-respool-trace)。
- B p* 一部の人たちには、あなたが投稿した記事にフォローアップするときに「親切な」複製を送る傾向があります。これらは普通はそこに Newsgroups ヘッダーが付いているのですが、いつもそうであるとは限りません。この命令 (gnus-summary-article-posted-p) は現在の記事をあなたのニュースサーバーから (というよりは、むしろ gnus-refer-article-method や gnus-select-method から) 取得しようとして、記事を発見できたかどうかを報告します。それが記事を発見しなかったとしても、それはとにかく投稿されているかもしれません—メールの伝達はニュースの伝達よりもずっと速いので、ニュースの複製がまだ到着していないだけかもしれないのです。
- 訳注: その「親切な」複製が、概略バッファで独立した記事として見えていないと検査することができないので、そうするために *A D* または *C-d* 命令 (see Section 3.26.4 [Really Various Summary Commands], page 108) を使う必要があるかもしれません。この命令はとにかくすべての選択方法を試すので、特にそれらに遅いものが含まれているときは、注意して使して下さい。

K E 記事の本文を暗号化します (gnus-article-encrypt-body)。本文は、変数 gnus-article-encrypt-protocol で指定されたプロトコルで暗号化されます。

いつも記事をどこかに移動 (もしくは複製) することを習慣にしているのならば、記事をどこに入れれば良いかを Gnus に提案してもらいたいと思うでしょう。gnus-move-split-methods は gnus-split-methods と同じ構文を使う変数です (see Section 3.15 [Saving Articles], page 73)。あなたが妥当だと思われる提案をするようにその変数をカスタマイズすることができます。(gnus-split-methods がファイル名を使うのに対して gnus-move-split-methods はグループ名を使うことに注意して下さい。)

```
(setq gnus-move-split-methods
      '(("From:.*Lars Magne" "nnml:junk")
        ("Subject:.*gnus" "nnfolder:important")
        (".*" "nnml:misc")))
```

3.26 概略のいろいろなもの

gnus-summary-display-while-building

非-nil だったら、構築中の概略バッファを更新しながら表示します。t だった場合は、行が挿入される度に毎回バッファを更新します。値が整数 *n* であった場合は、*n* 行毎に表示を更新します。デフォルトは nil です。

gnus-summary-display-arrow

非-nil だったら、現在の記事を指し示すためにフリッジに矢印を表示します。(訳注: フリッジとは Emacs 21 以上でウィンドウの左右に現れる余白のことです。)

gnus-summary-mode-hook

概略モードのバッファを作成するときにこのフックが呼ばれます。

gnus-summary-generate-hook

これはスレッド作成と概略バッファ作成の前に実行する最後のものとして呼ばれます。これはニュースグループの持っているデータに基づいてスレッドの変数をカスタマイズするのに非常に便利です。このフックはほとんどの概略バッファ変数が設定された後に概略バッファから呼ばれます。

gnus-summary-prepare-hook

これは概略バッファが作成された後に呼ばれます。例えば、これを何かしら神をも畏れぬ方法で行をハイライトしたり、バッファの見え方を修正したりするのに使ったりするかもしれません。

gnus-summary-prepared-hook

概略バッファが作成された後で一番最後に呼ばれるフックです。

gnus-summary-ignore-duplicates

Gnus が同じ Message-ID を持つ二つの記事を発見したときは、何か思い切ったことをしなければなりません。別の記事が同じ Message-ID を持つことは許されていませんが、それは何らかのソースからメールを読んでいるときに起こるかもしれません。この変数によって Gnus が何をするかをカスタマイズできるようになっています。nil だったら (それがデフォルトです)、Gnus は Message-ID を付け替えて (表示のためだけに) その記事を他の記事と同じように表示します。t にすると、それは記事を表示しません—最初から存在しなかったかのように。

`gnus-alter-articles-to-read-function`

この変数に設定した関数で、選択する記事のリストを変更することができます。関数は二つの引数 (グループ名と選択する記事のリスト) を受け付けます。

例えば以下の関数は、キャッシュされた記事のリストを、あるグループのリストだけに追加します。

```
(defun my-add-cached-articles (group articles)
  (if (string= group "some.group")
      (append gnus-newsgroup-cached articles)
      articles))
```

`gnus-newsgroup-variables`

ニュースグループ (その概略バッファの) のローカル変数、または変数とそれらの評価されるデフォルトの表現 (デフォルト値が `nil` でない場合) の `cons` セルのリストで、その概略バッファが活きている間はグローバル変数になります。(訳注: いわゆるバッファローカル変数ではありません。)

注: デフォルトの表現は単にローカル変数に設定されるのではなく、その前に (`eval` 関数を使って) 評価されます。デフォルトの表現が `global` というシンボルだった場合は評価されず、代わりにそのローカル変数のグローバル値が使われます。

これらグループパラメーターの値が他のバッファで行なわれる処理に影響するようになっていても、(訳注: その概略バッファの) グループパラメーターを設定するために使うことができます。例です:

```
(setq gnus-newsgroup-variables
      '(message-use-followup-to
        (gnus-visible-headers .
          "^From:\\|^Newsgroups:\\|^Subject:\\|^Date:\\|^To:")))
```

Section 2.10 [Group Parameters], page 23 も参照して下さい。

訳注: もっと良い例が必要です。 `gnus-newsgroup-variables` および `gnus-parameters` (see Section 2.10 [Group Parameters], page 23) の値を次のように設定したとしましょう:

```
(setq gnus-newsgroup-variables '((var . foo)))
(setq gnus-parameters
      '(("^fj\\." (var . bar))
        ("^japan\\." (var . baz))))
```

こうしておくと変数 `var` の値が、`'fj` 階層のニュースグループ (の概略バッファ) に入ると `bar` になり、`'japan` 階層のグループに入ると `baz` になります。グループを抜けても変数 `var` の値は変化しませんが、`'fj` または `'japan` 階層以外のグループに入ると変数 `var` の値は `foo` になります (正確には、`foo`, `bar` または `baz` の値は、本編で説明されているように `eval` した結果が使われます)。

通常のグループパラメーターは、そのグループの概略バッファでだけ値を知ることができるのに対して、`gnus-newsgroup-variables` で設定した変数は、同じ Emacs のどのバッファでも、現在選択されているグループ固有の値を持つ点が違います。異なる複数のグループの概略バッファを使う場合には、注意する必要があります。

特別な場合として `foo` が `nil` で良い場合は、次のように記述することができます:

```
(setq gnus-newsgroup-variables '(var))
(setq gnus-parameters
```

```
'(("^fj\\\" (var . bar))
  (^japan\\\" (var . baz))))
```

gnus-newsgroup-variables および gnus-parameters はどちらもリストなので、setq よりはむしろ add-to-list や push などを使って、値を「追加」した方が便利かもしれません。

3.26.1 概略グループ情報

- H f* 現在のグループの FAQ (frequently asked questions (頻繁にされる質問) のリスト) を取得しようとします (gnus-summary-fetch-faq)。Gnus は gnus-group-faq-directory (通常これは遠隔マシンのディレクトリー) から FAQ を取得しようとします。この変数はディレクトリーのリストであることもできます。その場合、この命令に接頭引数を与えることによっていろいろなサイトから選ぶことができます。おそらく ange-ftp もしくは efs がファイルの取得に使われるでしょう。
- H d* 現在のグループの簡潔な説明を表示します (gnus-summary-describe-group)。接頭引数が与えられると、サーバーから強制的に説明の再読み込みをします。
- H h* 最も重要な概略コマンドの、非常に簡潔な説明を表示します (gnus-summary-describe-briefly)。
- H i* Gnus の info の節 (node) に移動します (gnus-info-find-node)。

3.26.2 記事を探す

- M-s* それ以降のすべての (生の) 記事を正規表現で検索します (gnus-summary-search-article-forward)。
- M-r* それ以前のすべての (生の) 記事を正規表現で検索します (gnus-summary-search-article-backward)。
- M-S* 前回の前方検索を繰り返します (gnus-summary-repeat-search-article-forward)。
- M-R* 前回の後方検索を繰り返します (gnus-summary-repeat-search-article-backward)。
- &* この命令は、ヘッダー、そのヘッダーの内容に合致する正規表現、および合致したときに実行されるコマンドの入力を要求します (gnus-summary-execute-command)。ヘッダーが空文字列だったら、記事全体で合致するものを探します。接頭引数を与えられると、代わりに後ろ向きに探します。
- 例えば *& RET* 何かの *.**文字列 *RET #* は、ヘッダーか本文に「何かの *.**文字列」を持つすべての記事にプロセス印を付けます。
- M-&* この命令に続けて入力する命令を、プロセス印が付けられているすべての記事で実行します (gnus-summary-universal-argument)。

3.26.3 概略生成命令

- Y g* 現在の概略バッファを再作成します (gnus-summary-prepare)。
- Y c* (現在のグループのために) キャッシュされたすべての記事を概略バッファに挿入します (gnus-summary-insert-cached-articles)。

- Y d* (現在のグループのための) すべての保留記事を概略バッファに挿入します (gnus-summary-insert-dormant-articles)。
- Y t* (現在のグループのための) すべての可視記事を概略バッファに挿入します (gnus-summary-insert-ticked-articles)。

3.26.4 本当にいろいろな概略命令

A D

C-d 現在の記事が別の記事を寄せ集めたもの (例えばダイジェスト) であるならば、それらの記事でできているグループに入るためにこの命令を使うことができます (gnus-summary-enter-digest-group)。この命令に接頭引数を与えないと Gnus はどのような型の記事が現在表示されているかを推測しようとし、実際にはそれが『ダイジェスト』であるものとして強引に解釈します。基本的に、ある様式で寄せ集められた別のメッセージを見るときはいつでも、*C-d* を使うことによって、もっと便利なやり方でそれらのメッセージを読むことができます。

C-M-d この命令は上のものによく似ていますが、いくつかの文書を一つのおおきなグループに集めます (gnus-summary-read-read-document)。それを実現するために、この命令はそれぞれの文書のための nndoc グループを開いてから、それら複数の nndoc グループのてっぺんで nnvirtual グループを開きます。この命令はプロセス/接頭引数の習慣を理解します (see Section 8.1 [Process/Prefix], page 249)。

C-t 長い概略行を切り詰めるかどうかを切り替えます (gnus-summary-toggle-truncation)。これはおそらく概略バッファで行を中央に表示する機能を混乱させるので、記事を読んでいるときに行の切り詰めを off にするのは良い考えではないでしょう。

= 概略バッファのウィンドウを拡大します (gnus-summary-expand-window)。接頭引数を与えられると、記事バッファのためのウィンドウの配置の設定を強制します (訳注: デフォルトでは記事バッファのためのウィンドウの配置の設定には概略バッファを表示することも含まれているので、普通に記事を読んでいるときと同じになるでしょう)。

C-M-e 現在のグループのグループパラメーター (see Section 2.10 [Group Parameters], page 23) を編集します (gnus-summary-edit-parameters)。

C-M-a 現在のグループのグループパラメーター (see Section 2.10 [Group Parameters], page 23) をカスタマイズします (gnus-summary-customize-parameters)。

3.27 概略バッファを抜ける

概略バッファから抜けると、普通はグループのすべての情報を更新してグループバッファに戻ります。

Z Z

Z Q

q 現在のグループを出て、グループのすべての情報を更新します (gnus-summary-exit)。抜け出るための多くの処理を行なう前に gnus-summary-prepare-exit-hook が呼ばれ、それはデフォルトで gnus-summary-expire-articles を呼びます。抜け出るための処理を終えた後で gnus-summary-exit-hook が呼ばれます。グループモードに戻るときに (未読の) グループが残っていなかったら gnus-group-no-more-groups-hook が実行されます。

Z E

Q グループのどんな情報も更新せずに現在のグループを抜け出ます (gnus-summary-exit-no-update)。

Z c

c グループのすべての可視ではない (unticked) 記事に既読の印を付けてから抜けます (gnus-summary-catchup-and-exit)。

Z C

C 可視記事さえも含むすべての記事に既読の印を付けてから抜けます (gnus-summary-catchup-all-and-exit)。

Z n

n すべての記事に既読の印を付けて次のグループへ移動します (gnus-summary-catchup-and-goto-next-group)。

Z p

p すべての記事に既読の印を付けて前のグループへ移動します (gnus-summary-catchup-and-goto-prev-group)。

Z R

C-x C-s 現在のグループを出て、それから入り直します (gnus-summary-reselect-current-group)。接頭指数が与えられると、既読と未読の両方のすべての記事を選択します。

Z G

M-g グループを抜け、そのグループの新しい記事を調べてから、再びそのグループを選択します (gnus-summary-rescan-group)。接頭指数が与えられると、既読と未読の両方のすべての記事を選択します。

Z N

N グループを抜けて、次のグループへ移動します (gnus-summary-next-group)。

Z P

P グループを抜けて、前のグループへ移動します (gnus-summary-prev-group)。

Z s

s 現在の既読と印付き記事の数をドリブルバッファ (dribble buffer) に保存し、それからドリブルバッファを保存します (gnus-summary-save-news). 接頭指数が与えられると '.newsrsrc' ファイル (と '.newsrsrc.eld' ファイル) も保存します。この命令を使うと、更新なしで抜け出すこと (**Q** 命令) は意味が無くなります。

グループのすべての情報を「更新」して現在のグループを抜けるときに gnus-exit-group-hook が呼ばれます。例えば **Q** 命令 (gnus-summary-exit-no-update) はこのフックを呼びません。

グループを抜けた後でそれを後悔する癖があるのなら、gnus-kill-summary-on-exit を nil に設定と良いかもしれませんが。そうすると Gnus は抜け出るときに概略バッファを削除しません。(何という驚き!) 代わりに、それはバッファの名前を '*Dead Summary ... *' のようなものに変更して、gnus-dead-summary-mode というマイナーモードを導入します。今やそのバッファに切り替えると、すべてのキーが関数 gnus-summary-wake-up-the-dead に割り当てられていることに気付くでしょう。死んだ概略バッファ (dead summary buffer) でどんなキーでも叩くと、それは生きた普通の概略バッファになります。

死んだ概略バッファは同時に一つしか存在することはできません。

概略バッファを抜け出ると、現在のグループのデータは更新されます (どの記事を読んで、どの記事に返答したか、などなど。) もし変数 gnus-use-cross-reference が t であると (それがデフォルトです)、そのグループに相互参照された (cross referenced) 記事には、それがクロスポストされた他の購読しているグループにあっても、既読の印が付きます。この変数が nil でも t でもなければ、記事には購読しているグループと購読していないグループの両方で既読の印が付きます (see Section 3.28 [Crosspost Handling], page 110)。

3.28 クロスポストの扱い

クロスポストされた記事に既読の印を付けることによって、同じ記事を二回以上読まないで済むことを保証します。もちろん、だれかがそれを複数のグループに別々に投稿しない限りは。同じ記事を複数のグループに（クロスポストではなく）投稿することは *spamming* と呼ばれ、あなたはそのような憎むべき犯罪を行なうものに対して、法律によって不快な記事を送ることが義務づけられています。spam を振り落とすために、NoCeM で処理することを試してみる必要があるかもしれません (see Section 8.12 [NoCeM], page 262)。

覚えておいて下さい: クロスポストはまあ構いませんが、同じ記事を別々に複数のグループに投稿するのは許されません。大量のクロスポスト (*velveeta* として知られているもの) は何としても避けられるべきで、過剰なクロスポストに対して不満を言うために `gnus-summary-mail-crosspost-complaint` 命令を使うことさえできます。

Gnus にクロスポストを正しく扱えなくさせる原因の一つは、XOVER (これは非常に良いです、というのはそれは速度をとて速くするからです) をサポートしているけれども NOV 行に Xref 欄を含めない NNTP サーバーを使っていることです。これは害悪です。でも、ああ、悲しいかな、非常に良くあることなのです。Gnus はあなたが読んだすべての記事に Xref 行を記録することによって The Right Thing (正しいこと) をしようとしませんが、記事を削除したり単に読まないで既読の印を付けると、Gnus がこれらの記事の Xref 行をのぞきまわる機会が無くなってしまいますので、相互参照 (cross reference) の機構を使えなくなってしまいます。

あなたの NNTP サーバーがその概観ファイル (overview file) に Xref 欄を含めるかどうかを調べるには、`'telnet your.nntp.server nntp'` をタイプして、inn サーバーでは `'MODE READER'` コマンドを与えてから、`'LIST overview.fmt'` を試して下さい。これは動作しないかもしれません。しかし、もし動作して、取得した最後の行が `'Xref:full'` でないならば (訳注: 最後の行ではないかもしれません)、ニュースの管理者が概観ファイルに Xref 欄を含めるようにしてくれるまで、彼女に向かって叫び、泣き付くべきでしょう。

Gnus にいつでも正しい Xref を取得するようにさせたいのであれば、`gnus-nov-is-evil` を `t` にする必要があり、それは非常に速度を遅くします。

ま、人生はそのようなものです。

代替手段に付いては Section 3.29 [Duplicate Suppression], page 110 を参照して下さい。

3.29 重複の抑制

デフォルトでは Gnus はクロスポスト機構を利用することによって、同じ記事を二回以上読まないようにしようとします (see Section 3.28 [Crosspost Handling], page 110)。しかし、その単純で効果的な方法は、いろいろな理由により、満足する結果をもたらさないかもしれません。

1. NNTP サーバーは Xref 欄の生成に失敗するかもしれません。これは悪いことで、あまり起こりません。
2. NNTP サーバーは `'overview'` データベースに Xref 欄を含めるのに失敗するかもしれません。これは悪いことで、非常に良くあることです、ああ悲しい。
3. 同じグループ (もしくはいくつかの関連したグループ) を違った NNTP サーバーから読んでいるかもしれません。
4. グループに投稿された記事と重複するメールを受け取ったかもしれません。

Xref の扱いに失敗する状況は確かに他にもありますが、これら四つが最も良くある状況です。

もし、本当にもしも Xref の扱いに失敗したら、「重複抑制」に切り替えることを考慮する必要があるかもしれません。そうすれば、Gnus はあなたが読んだすべての記事、あるいは既読の印を付け

たすべての記事の Message-ID を記憶し、そしてまるで魔法のように、以後それらを読むときはいつでも既読の印が付いているようにします—すべてのグループで。この機構を使うのは何だかとても非効率になりそうですが、過度に非効率なわけではありません。同じ記事を二回以上読むよりは、間違い無く望ましいです。

重複抑制はあまり精密な道具ではありません。どちらかというと大槌のようなものです。それは非常に単純なやり方で動作しています—あなたが記事に既読の印を付けると、その Message-ID をキャッシュに加えます。次にその Message-ID に出会うと、‘M’ 印によって記事に既読の印を付けます。その記事をどのグループで見たかは気にしません。

`gnus-suppress-duplicates`

`nil` でなければ、重複抑制をします。

`gnus-save-duplicate-list`

`nil` でなければ、重複のリストをファイルに保存します。これは起動と終了の時間を長くするので、デフォルトは `nil` です。しかし、これは Gnus を一回実行したときに読まれた重複記事だけが抑制されるということです。

`gnus-duplicate-list-length`

この変数はどのくらい多くの Message-ID を重複抑制リストに保っておくかを決定します。デフォルトは 10000 です。

`gnus-duplicate-file`

重複抑制のリストを格納しておくファイルの名前です。デフォルトは `~/News/suppression` です。

何度も Gnus を終了して起動する傾向があるのであれば、おそらく `gnus-save-duplicate-list` を `t` にするのは良い考えでしょう。もし Gnus を続けて何週間も走らせておくのであれば、それを `nil` にした方が良くもありません。一方、リストを保存することは起動と終了をずっと遅くするので、頻繁に Gnus を終了して起動するのであれば、`gnus-save-duplicate-list` を `nil` に設定するべきです。うーむ。私はあなたがどうするか任せようと思います。

3.30 セキュリティー

Gnus は署名されたメッセージを検証したり、暗号化されたメッセージをデコードすることができます。PGP, PGP/MIME および S/MIME の形式をサポートしますが、それらを動作させるためには、いくつかの外部プログラムを必要とします:

1. PGP と PGP/MIME のメッセージを扱うには、OpenPGP の実装である GnuPG のようなものをインストールしなければなりません。Gnus に含まれている GnuPG へのインターフェースは PGG というもの (see section “PGG” in *PGG Manual*) ですが、Mailcrypt と `gpg.el` もサポートします。
2. S/MIME のメッセージを扱うには、OpenSSL をインストールする必要があります。OpenSSL 0.9.6 か、それより新しいものがお勧めです。

以下は、メッセージを読む場合のセキュリティーの機能を制御するための変数です:

`mm-verify-option`

署名されたパートを検証するためのオプション。`never` は検証しない、`always` はいつも検証する、`known` は知られたプロトコルの場合だけ検証する、の意味です。それら以外の場合は、どうするかを利用者に尋ねます。

mm-decrypt-option

暗号化されたパートをデコードするためのオプション。never はデコードしない、always はいつもデコードする、known は知られたプロトコルの場合だけデコードする、の意味です。それら以外の場合は、どうするかを利用者に尋ねます。

mml1991-use

PGP のメッセージのための、OpenPGP の実装への elisp インターフェースを示すシンボルです。デフォルトは pgg ですが、反論があるものの mailcrypt と gpg もサポートします。

mml2015-use

PGP/MIME のメッセージのための、OpenPGP の実装への elisp インターフェースを示すシンボルです。デフォルトは pgg ですが、反論があるものの mailcrypt と gpg もサポートします。

デフォルトではセキュリティの情報を表示するボタンが現れません。それらは実際にメールを読む際に邪魔になるからです。K b をタイプすれば、その情報を表示することができます。これを恒久的に行なわせるには、gnus-buttonized-mime-types および gnus-unbuttonized-mime-types 変数を使って下さい。これらの変数の詳細と、常にセキュリティの情報を表示させるためにカスタマイズする方法は、Section 3.18 [MIME Commands], page 94 を参照して下さい。

メニュー項目やコマンドから OpenPGP の鍵を取得 (snarf) する (すなわち、記事から鍵を鍵束に輸入 (import) する) 機能は、明示的にはサポートされません。というよりはむしろ、あなたが適切だと思うどんな動作をも通常の MIME の機構を介して指定できるように、Gnus は 'application/pgp-keys' として鍵を検出し、ラベルを付けます。MIME ボタンをクリック (see Section 4.2 [Using MIME], page 116) したときに、GNU Privacy Guard を使って鍵を輸入してくれるようにするには、以下のような行を '~/.mailcap' ファイル (see section "mailcap" in *The Emacs MIME Manual*) に記入して下さい。

```
application/pgp-keys; gpg --import --interactive --verbose; needsterminal
```

これは、たまたま mailcap-mime-data ですでに定義されている、デフォルトの動作でもあります。

送信するメッセージに署名したり暗号化するために、どうやって設定するかについてのもっと詳しい情報が、message マニュアル (see section "セキュリティ" in *The Message Manual*) で見つかるでしょう。

3.31 メーリングリスト

Gnus は RFC 2369 で規定された各種のメーリングリストで使われるフィールドを理解します。これを有効にするには概略バッファで A M (gnus-mailing-list-insinuate) を使うなどして、to-list グループパラメーター (see Section 2.10 [Group Parameters], page 23) を追加して下さい。

これによって概略バッファでの以下の命令が使えるようになります。

- C-c C-n h List-Help フィールドがあったら、メーリングリストのヘルプを取り寄せるためのメッセージを送信します。
- C-c C-n s List-Subscribe フィールドがあったら、メーリングリストの講読を始めるためのメッセージを送信します。
- C-c C-n u List-Unsubscribe フィールドがあったら、メーリングリストの講読をやめるためのメッセージを送信します。

- C-c C-n p* List-Post フィールドがあったら、メーリングリストに投稿します。
- C-c C-n o* List-Owner フィールドがあったら、メーリングリストの管理者宛てにメッセージを送信します。
- C-c C-n a* List-Archive フィールドがあったら、メーリングリストのアーカイブを閲覧します。

4 記事バッファ

記事は一つしかない記事バッファに表示されます。すべての概略バッファは (Gnus に指示しない限り) 同じ記事バッファを共有します。

4.1 余分なヘッダーを隠す

各記事の頭の部分はヘッダー (*head*) と呼ばれます。(残りの部分はボディ (*body*) です。すでにお気づきでしょう。)

ヘッダーにはたくさんの有益な情報が含まれています。記事を書いた人の名前、それが書かれた日付、および記事の表題です。これはとても良いのですが、ヘッダーには大部分の人にとっては見たくない情報—記事があなたのところに着くまでにどんなシステムを経由してきたか、Message-ID、References などなど...もううんざりするくらい—たくさん含まれています。たぶんあなたはこれらの行のいくつかは取り除いてしまいたいと思うでしょう。もしこれらの行をすべて記事バッファ内に残しておきたければ、`gnus-show-all-headers` を `t` に設定して下さい。

Gnus はヘッダーを選り分けるために二つの変数を用意しています:

`gnus-visible-headers`

この変数が `nil` 以外であれば、どのヘッダーを記事バッファに残したいかを指定する正規表現であるとみなされます。この変数に合致しないヘッダーはすべて隠されます。

例えば、記事を書いた人の名前と表題のみを見たいければ、こう指定します:

```
(setq gnus-visible-headers "^From:\\|^Subject:")
```

この変数は、表示させたいヘッダーに合致する正規表現をリストで指定することもできます。

`gnus-ignored-headers`

この変数は `gnus-visible-headers` の反対です。この変数が設定されていれば (かつ `gnus-visible-headers` が `nil` であれば)、これは隠したいヘッダー行すべてに合致する正規表現であるとみなされます。この変数に合致しないすべてのヘッダー行が表示されます。

例えば、単に References 欄と Xref 欄のみを消し去りたければ、以下のようにします:

```
(setq gnus-ignored-headers "^References:\\|^Xref:")
```

この変数は消したいヘッダーに合致する正規表現のリストでも構いません。

なお、`gnus-visible-headers` が `nil` 以外の場合は、この変数には効果が無いことに注意して下さい。

Gnus はヘッダーの並べ替え (`sort`) も行ないます (これはデフォルトで行なわれます)。この並べ替えは `gnus-sorted-header-list` 変数を設定することで制御することができます。これはヘッダーをどういう順序で表示するかを指定する正規表現のリストです。

例えば、記事の著者名を最初に、次に表題を表示したければ、こんな風になるでしょう。

```
(setq gnus-sorted-header-list '("^From:" "^Subject:"))
```

表示するようになっているヘッダーでこの変数に指定されていないものは、この変数に指定されているすべてのヘッダーの後に、適当な順序で表示されるでしょう。

`gnus-treat-hide-boring-headers` を `head` に設定することによって、もっとつまらないヘッダーを隠すことができます。この関数が何をするかは `gnus-boring-article-headers` 変数に依存します。この変数はリストですが、このリストには実際のヘッダーの名前が入るものではありません。

ん。代わりに Gnus がチェックして視界から消し去るためのさまざまな「つまらない条件」(*boring conditions*) のリストを指定します。

この条件には以下のようなものがあります。

empty 空のヘッダーをすべて消去します。

followup-to Followup-To 欄が Newsgroups 欄と同一である場合には消去します。

reply-to Reply-To 欄が From 欄と同じアドレスを示しているか、**broken-reply-to** グループパラメーターが設定されている場合には消去します。

newsgroups Newsgroups 欄が現在のグループ名しか含んでいない場合には消去します。

to-address To 欄が現在のグループの **to-address** パラメーターと同じものしか含んでいない場合には消去します。

to-list To 欄が現在のグループの **to-list** パラメーターと同じものしか含んでいない場合には消去します。

cc-list Cc 欄が現在のグループの **to-list** パラメーターと同じものしか含んでいない場合には消去します。

date その記事が過去三日以内のものであれば、Date 欄を消去します。

long-to To 欄および/または Cc 欄があまりにも長い場合には消去します。

many-to To 欄および/または Cc 欄が一つよりも多ければ、それらをすべて消去します。

これらのうちの三つの要素を入れたければ、こんな風になります:

```
(setq gnus-boring-article-headers
      '(empty followup-to reply-to))
```

これはこの変数のデフォルト値でもあります。

4.2 MIME を使う

パントマイム (*mime*) は、観客があくびをしながらぼんやりしているのにもかかわらず、意味も無く空中で手を振るものの標準として広く知られています。

一方 MIME は、そのためにすべてのニュースリーダーが恐怖で死んでしまうのにもかかわらず、意味も無く記事をエンコードする標準です。

MIME は記事がどんな文字セットを使うか、文字をどうエンコードするかを指定することができ、さらには絵やその他のみだらなものを無邪気な格好の記事に埋め込むことさえ可能にします。

Gnus は MIME パートを表示するために、**gnus-display-mime-function** によって MIME 記事进行处理します。これはデフォルトでは **gnus-display-mime** で、MIME オブジェクトを表示し、セーブし、かつ操作するために使うことができる、ひとかたまりのクリック可能なボタンを作成します。

MIME ボタンの上にポイントを置いたならば、以下のコマンドが利用できます:

RET (記事)

BUTTON-2 (記事)

MIME オブジェクトの表示をトグルで切り替えます (`gnus-article-press-button`)。そのオブジェクトを内蔵のビューワーで表示できないときは、Gnus は `'mailcap'` ファイルにある外部のビューワーに助けを求めます。ビューワーが `'copiousoutput'` 仕様になっている場合は、オブジェクトはインラインで (訳注: Emacs の表示に埋め込まれて) 表示されます。

M-RET (記事)

v (記事) 手段を尋ね、その手段を使って MIME オブジェクトを表示します (`gnus-mime-view-part`)。

t (記事) MIME オブジェクトを、異なる MIME メディア・タイプであるかのように表示します (`gnus-mime-view-part-as-type`)。

C (記事) 文字セットを尋ね、その文字セットを使って MIME オブジェクトを表示します (`gnus-mime-view-part-as-charset`)。

o (記事) ファイル名を尋ねて MIME オブジェクトをセーブします (`gnus-mime-save-part`)。

C-o (記事)

ファイル名を尋ね、MIME オブジェクトをセーブして、それを記事から取り外します (記事を編集することによって行なわれます)。取り外された MIME オブジェクトは `message/external-body` MIME タイプとして参照されるようになります (`gnus-mime-save-part-and-strip`)。

r (記事) ファイル名の入力を求めて、MIME オブジェクトを `message/external-body` 型の MIME 形式のファイルとして参照される外部にある本体で置き換えます。 (`gnus-mime-replace-part`)。

d (記事) 記事から MIME オブジェクトを取り外し、取り外したことを表す告知で置き換えます (`gnus-mime-delete-part`)。

c (記事) MIME オブジェクトを新たに作ったバッファにコピーして、それを表示します (`gnus-mime-copy-part`)。接頭引数が与えられると、デコードせずに生の内容物をコピーします。数値の接頭引数を与えると、文字セットによるデコードを半手動で切り替えることができます (Section 3.4 [Paging the Article], page 50 で述べられている `gnus-summary-show-article-charset-alist` を参照して下さい)。 `auto-compression-mode` (see section “Accessing Compressed Files” in *The Emacs Editor*) が設定されていると、`'gz'` や `'bz2'` のような圧縮されたファイルを自動的に解凍します。

p (記事) MIME オブジェクトを印刷します (`gnus-mime-print-part`)。このコマンドは `'mailcap'` ファイルで定義された `'print='` 仕様に従います。

i (記事) MIME オブジェクトの内容物を、その記事バッファに `text/plain` として挿入します (`gnus-mime-inline-part`)。接頭引数が与えられると、デコードせずに生の内容物を挿入します。数値の接頭引数を与えると、文字セットによるデコードを半手動で切り替えることができます (Section 3.4 [Paging the Article], page 50 で述べられている `gnus-summary-show-article-charset-alist` を参照して下さい)。 `auto-compression-mode` (see section “Accessing Compressed Files” in *The Emacs Editor*) の設定とは無関係に、`'gz'` や `'bz2'` のような圧縮されたファイルを `jka-compr` を使って自動的に解凍します。

- E (記事) 内部ビューワーで MIME オブジェクトを表示します。内部ビューワーが使えないときは、外部ビューワーを使います (gnus-mime-view-part-internally)。
- e (記事) 外部ビューワーで MIME オブジェクトを表示します (gnus-mime-view-part-externally)。
- l (記事) MIME オブジェクトをプロセスに出力します (gnus-mime-pipe-part)。
- . (記事) MIME オブジェクトをどう処理するかを、対話的に決めて実行します (gnus-mime-action-on-part)。

Gnus はいくつかの種類の MIME オブジェクトを自動的に表示します。どのパートに対してそうするかを Gnus が決めるやり方については、Emacs MIME マニュアルで述べられています。

不愉快なものでびっくりさせられるのを避けるには、トグルで切り替える関数を使うのが最も良いでしょう。(例えば、‘alt.sing-a-long’ グループに入ると、あなたの気づかないうちに MIME は記事中のサウンドファイルをデコードして、何やら怪しげな長い長い歌があなたのスピーカーから大音響で流れ出し、あなたはボリュームボタンを見つけられず、というのはそんなものはもともと付いていないからで、みんなはあなたの方を睨みはじめ、あなたはプログラムを止めようとするけれどもできなくて、ボリュームを制御するプログラムも見つけられなくて、そして部屋中の全員は突然あなたのことを軽蔑の眼差しで見えるようになってしまい、あなたはちょっと面白くない思いをする、とか)。

現実の出来事と実在の人物に類似しているかもしれませんが、これはすべてホントのことです。げほげほ。

Section 3.18 [MIME Commands], page 94 も見て下さい。

4.3 記事のカスタマイズ

記事をどのように見せるかをカスタマイズするためのたくさんの関数が存在しています。これらの関数を対話的に呼ぶこともできるし (see Section 3.17.4 [Article Washing], page 85)、記事を選択したときに自動的に選択することもできます。

自動的に呼ばれるようにするためには、対応するトリートメント変数を設定しなければなりません。例えばヘッダーを隠すためには、gnus-treat-hide-headers を設定します。以下は設定できる変数の一覧ですが、まずこれらの変数の取り得る値について話しましょう。

注意: いくつかの値は、有効な値であってもほとんど意味を無しません。実用的な値は下の一覧を調べて下さい。

1. nil: このトリートメントをしません。
2. t: このトリートメントをすべての本文のパートで行ないます。
3. head: ヘッダーでそのトリートメントをします。
4. first: このトリートメントを最初の本文のパートで行ないます。
5. last: このトリートメントを最後の本文のパートで行ないます。
6. 整数: このトリートメントをこの数値より短いすべての本文のパートで行ないます。
7. 文字列のリスト: このリストに含まれている正規表現に合致する名前のグループで読まれた記事の、すべての本文のパートでこのトリートメントを行ないます。
8. 最初の要素が文字列でないリストです:
リストは再帰的に評価されます。リストの最初の要素は述語です。以下の述語が認識されます:
or, and, not, typep. 例です:

```
(or last
  (typep "text/x-vcard"))
```


ここで「パート」という語が使われていることに気付いたと思います。これはメッセージには MIME マルチパート記事があり、いくつかのパートに分割されているかもしれないという事実に関連しています。マルチパートでない記事は一つのパートのみであるとみなされます。

このトリートメントはすべてのマルチパートのパートたちに適用されるのでしょうか？ はい、そうしなければそうなります。ですが、デフォルトでは ‘text/plain’ パートだけにトリートメントが施されます。これは `gnus-article-treat-types` 変数で制御され、これはパートの型に合致する正規表現のリストです。制御変数の値が、上で説明されているように述語のリストであるときは、この変数は無視されます。

以下のトリートメントのオプションが使用可能です。これをカスタマイズするための最も簡単な方法は `gnus-article-treat` カスタマイズグループを調査することです。丸括弧の中の値は提案されている意味のある値です。他のものも可能ですが、ほとんどの人にとってはおそらくここに一覧表示されているもので十分でしょう。

```
gnus-treat-buttonize (t, integer)
gnus-treat-buttonize-head (head)
```

See Section 3.17.6 [Article Buttons], page 89.

```
gnus-treat-capitalize-sentences (t, integer)
gnus-treat-overstrike (t, integer)
gnus-treat-strip-cr (t, integer)
gnus-treat-strip-headers-in-body (t, integer)
gnus-treat-strip-leading-blank-lines (t, first, integer)
gnus-treat-strip-multiple-blank-lines (t, integer)
gnus-treat-strip-pem (t, last, integer)
gnus-treat-strip-trailing-blank-lines (t, last, integer)
gnus-treat-unsplit-urls (t, integer)
gnus-treat-wash-html (t, integer)
```

See Section 3.17.4 [Article Washing], page 85.

```
gnus-treat-date-english (head)
gnus-treat-date-iso8601 (head)
gnus-treat-date-lapsed (head)
gnus-treat-date-local (head)
gnus-treat-date-original (head)
gnus-treat-date-user-defined (head)
gnus-treat-date-ut (head)
```

See Section 3.17.8 [Article Date], page 91.

```
gnus-treat-from-picon (head)
gnus-treat-mail-picon (head)
gnus-treat-newsgroups-picon (head)
```

See Section 8.17.4 [Picons], page 268.

```
gnus-treat-display-smileys (t, integer)
gnus-treat-body-boundary (head)
```

ヘッダーと本文の間に境界線を追加します。境界線には `gnus-body-boundary-delimiter` に設定された文字列が使われます。

See Section 8.17.3 [Smileys], page 267.

`gnus-treat-display-x-face (head)`

See Section 8.17.1 [X-Face], page 265.

`gnus-treat-display-face (head)`

See Section 8.17.2 [Face], page 267.

`gnus-treat-emphasize (t, head, integer)`

`gnus-treat-fill-article (t, integer)`

`gnus-treat-fill-long-lines (t, integer)`

`gnus-treat-hide-boring-headers (head)`

`gnus-treat-hide-citation (t, integer)`

`gnus-treat-hide-citation-maybe (t, integer)`

`gnus-treat-hide-headers (head)`

`gnus-treat-hide-signature (t, last)`

`gnus-treat-strip-banner (t, last)`

`gnus-treat-strip-list-identifiers (head)`

See Section 3.17.3 [Article Hiding], page 83.

`gnus-treat-highlight-citation (t, integer)`

`gnus-treat-highlight-headers (head)`

`gnus-treat-highlight-signature (t, last, integer)`

See Section 3.17.1 [Article Highlighting], page 81.

`gnus-treat-play-sounds`

`gnus-treat-translate`

`gnus-treat-ansi-sequences (t)`

`gnus-treat-x-pgp-sig (head)`

`gnus-treat-unfold-headers (head)`

`gnus-treat-fold-headers (head)`

`gnus-treat-fold-newsgroups (head)`

`gnus-treat-leading-whitespace (head)`

See Section 3.17.5 [Article Header], page 88.

もちろん、`gnus-part-display-hook` から呼ばれる自分用の関数を書くこともできます。関数はそのパートに範囲が狭められた状態で呼ばれ、ほとんどなんでも好きなことができます。バッファに保存しておかなければならない情報はありません—何でも変えることができます。

4.4 記事のキーマップ

概略バッファにおけるキー操作のほとんどは記事バッファでも使用できます。これらは概略バッファでそれらを押したかのように動作するはずです。つまり記事を読んでいる間、実際に概略バッファを表示させておく必要がありません。すべての操作は記事バッファから行なうことができるのです。

`v` キーはユーザー用に予約されています。そのまま何かの機能に割り当てても構いませんが、接頭キーとして使う方が良いでしょう。

他にもいくつかのキーが利用できます:

`SPACE` 記事を一ページ先にスクロールします。(gnus-article-next-page)。`h SPACE h` とまったく同じです。

<i>DEL</i>	記事を一ページ前にスクロールします (gnus-article-prev-page)。 <i>h DEL h</i> とまったく同じです。
<i>C-c ^</i>	カーソルが Message-ID の近辺にあるときに <i>C-c ^</i> を押すと、Gnus はサーバーからその記事を取ってこようとします (gnus-article-refer-article)。
<i>C-c C-m</i>	カーソルの近くにあるアドレスに返信を送ります (gnus-article-mail)。接頭引数を与えると、そのメールを引用します。
<i>s</i>	バッファを再配置して、概略バッファが見えるようにします (gnus-article-show-summary)。
<i>?</i>	利用できるキー操作のごく簡単な説明を出します (gnus-article-describe-briefly)。
<i>TAB</i>	次のボタンがあればそこに移動します (gnus-article-next-button)。これは記事にボタンを付ける機能をオンにしているときのみ意味を持ちます。
<i>M-TAB</i>	一つ前のボタンがあればそこに移動します (gnus-article-prev-button)。
<i>R</i>	現在の記事に元記事を含んだ返答のメールを送ります (gnus-article-reply-with-original)。接頭引数を与えると広い返答 (wide reply) になります。もし領域が活性化されていたならば、その領域にあるテキストだけを yank します。
<i>F</i>	現在の記事に元記事を含んでフォローアップをします (gnus-article-followup-with-original)。接頭引数を与えると広い返答 (wide reply) になります。もし領域が活性化されていたならば、その領域にあるテキストだけを yank します。

4.5 記事のその他

gnus-single-article-buffer

nil 以外であれば、すべてのグループに対して同じ記事バッファを使用します (これはデフォルトです)。nil であれば、各グループ毎の固有の記事バッファを持つようになります。

gnus-article-decode-hook

MIME 記事をデコードするときに使用されるフックです。デフォルト値は (article-decode-charset article-decode-encoded-words) です。

gnus-article-prepare-hook

このフックは記事が記事バッファに挿入された直後に呼び出されます。これは主に、何か記事の内容に依存する処理をする関数のために用意されています。記事バッファの内容を変更するような目的で使うべきではないでしょう。

gnus-article-mode-hook

記事モードのバッファで呼び出されるフックです。

gnus-article-mode-syntax-table

記事バッファで用いられる構文テーブル (syntax table) です。これは text-mode-syntax-table をもとに初期化されます。

gnus-article-over-scroll

非-nil にすることによって、それ以上スクロールする新しいテキストが無くても記事バッファをスクロールできるようにします。デフォルトは nil です。(訳注: 記事の最下行が見えているときに、nil だと *RET* キーでそれ以上スクロールしませんが、非-nil にすると記事が見えなくなるまでスクロールします。)

`gnus-article-mode-line-format`

この変数は `gnus-summary-mode-line-format` と同じ仕様に沿った様式文字列です (see Section 3.1.3 [Summary Buffer Mode Line], page 47)。これは、その変数と同じ様式指定および二つの拡張を受付けます。

- ‘w’ 記事の「洗濯状態」(*wash status*)。これは記事に対して行なわれたであろう洗濯操作を、それぞれ一文字で示す短い文字列になります。文字とそれらの意味は次の通りです:
 - ‘c’ 記事バッファにおいて、引用された文が隠されているかもしれない場合に表示されます。
 - ‘h’ 記事バッファにおいて、ヘッダーが隠されている場合に表示されます。
 - ‘p’ 記事が電子署名または暗号化されていて、Gnus がセキュリティのためのヘッダーを隠していると表示されます。(注: 署名が正しいか間違っているかを表すものではありません。)
 - ‘s’ 記事バッファにおいて、署名が隠されている場合に表示されます。
 - ‘o’ 記事バッファにおいて、Gnus が重ね打ち文字のトリートメントを行なった場合に表示されます。
 - ‘e’ 記事バッファにおいて、Gnus が強調された文字のトリートメントを行なった場合に表示されます。
- ‘m’ 記事の MIME パートの数です。

`gnus-break-pages`

改ページ (*page breaking*) を行なうかどうかを制御します。この変数が `nil` 以外であれば、記事中にページ区切り文字が現れるごとにページ分割をします。この変数が `nil` であればページ分けは行なわれません。

`gnus-page-delimiter`

これが上で触れた区切り文字です。デフォルトでは ‘`^L`’ (フォームフィード) です。

`gnus-use-idna`

この変数は ‘From:’、‘To:’ および ‘Cc:’ ヘッダーにある国際化ドメイン名を、Gnus が IDNA デコードするかどうかを制御します。そのようなメッセージの作り方については See section “国際化ドメイン名” in *The Message Manual*, を参照して下さい。これには GNU Libidn (<http://www.gnu.org/software/libidn/>) が必要で、この変数はそれをインストールしてある場合だけ有効になります。

5 メッセージの作成

すべての投稿とメールを送るためのコマンドは、あなたをメッセージバッファに導きます。ここでは `C-c C-c` を押すことによって記事を送信する前に、記事を好きなように編集することができます。See section “概要” in *The Message Manual*. メッセージはあなたの設定に基づいて投稿またはメールとして送信されます (see Section 5.2 [Posting Server], page 123)。

投稿するべきでなかった記事を削除するための情報について Section 3.5.4 [Canceling and Superseding], page 54 も参照して下さい。

5.1 メール

出て行くメールをカスタマイズする変数です:

`gnus-uu-digest-headers`

要約メッセージ (digested message) に含まれるヘッダーに合致する正規表現のリストです。ヘッダーは合致した順に取り込まれます。`nil` だったら、すべてのヘッダーを含みます。

`gnus-add-to-list`

`nil` でなければ、`a` を押したときに、`to-list` グループパラメーターをその無いメールグループに付け加えます。

`gnus-confirm-mail-reply-to-news`

非-`nil` だったら、あなたがニュース記事への返答をメールでしようとするとき Gnus は確認を求めます。`nil` ならば、あなたがやりたいことに何も口出ししません。これは関数か正規表現であることもできます。関数は唯一のパラメーターとしてグループ名を受け取り、確認する必要がある場合に非-`nil` を返します。これを正規表現にすると、それに合致する名前のグループで確認を求めます。

メールで返信する気は無いのに時々まぞんざいに `R` を押してしまう癖があるならば、この変数はそんなあなたのためにあります。

`gnus-confirm-treat-mail-like-news`

非-`nil` だったら、Gnus はメールに返信する時にも `gnus-confirm-mail-reply-to-news` に基づいた確認を求めます。これはメーリングリストをニュースグループのように扱うのに便利です。

5.2 投稿するサーバー

最新の (もちろん、非常に知的な) 記事を送り出すために、あの魔法のような `C-c C-c` キーを押した時、それはどこにいくのでしょうか?

尋ねてくれてありがとう。あなたを恨みます。

それは非常に複雑になり得ます。

ニュースを投稿するとき、通常 Message は `message-send-news` を呼び出します (see section “ニュース変数” in *The Message Manual*)。普通は、Gnus は講読用と同じ選択方法を使って投稿します (このことは、あなたがたくさんのグループを異なったサーバーで講読している場合に、たぶん都合が良いのです)。しかし、あなたが講読しているそのサーバーが投稿を許可せず、読むことのみを許可しているのならば、おそらくあなたの (非常に知的でとんでもなく興味深い) 記事を投稿するために、他のサーバーを使いたいと思うでしょう。そうならば `gnus-post-method` を他の方法に設定することができます:

```
(setq gnus-post-method '(nnsPOOL ""))
```

さて、この設定をした後でサーバーがあなたの記事を拒否したり、サーバーが落ちていたりしたら、どうしたらよいのでしょうか？ この変数よりも優先させるために `C-c C-c` 命令にゼロでない数の接頭引数を与えることによって、投稿に“current”(現在の)サーバーを使わせること、すなわちデフォルトの動作 (訳注: `gnus-post-method` のデフォルト値は `current`) に戻すことができます。

もし、ゼロを接頭引数としてその命令に与えたなら (すなわち、`C-u 0 C-c C-c`)、Gnus は投稿にどの方法を使うかをあなたに尋ねます。

`gnus-post-method` を選択方法のリストにすることもできます。その場合は、Gnus は常に投稿にどの方法を使うかをあなたに尋ねます。

最後に、あなたがいつでも基本の選択方法を使って投稿したいのならば、この変数を `native` にして下さい。

メールを送信するときに、Message は `message-send-mail-function` を呼び出します。デフォルトの関数 `message-send-mail-with-sendmail` は記事を順番待ちに入れ (queuing) たり送信するために、記事を `sendmail` コマンドにパイプします。ローカルシステムが `sendmail` でメールを送信するように設定されていなくても、あなたが遠隔 SMTP サーバーに接続する権利を持っているならば、`message-send-mail-function` を `smtpmail-send-it` に設定することができます。smtpmail パッケージを使うために正しい設定がなされているかどうか確認して下さい。例です:

```
(setq message-send-mail-function 'smtpmail-send-it
      smtpmail-default-smtp-server "YOUR SMTP HOST")
```

これと似たものに `message-smtpmail-send-it` があります。ISP が POP-before-SMTP の認証を要求している場合に有用です。See Section 5.3 [POP before SMTP], page 124.

他の可能な `message-send-mail-function` の選択肢には `message-send-mail-with-mh`, `message-send-mail-with-qmail` および `feedmail-send-it` があります。

5.3 POP before SMTP

あなたの ISP は POP-before-SMTP 認証を要求しますか？ それは、メールを送信する前の一定時間以内に POP メールサーバーに接続しなければならないかどうかです。もしそうならば、便利な手があります。それには ‘`~/gnus.el`’ ファイルに以下の行を入れて下さい:

```
(setq message-send-mail-function 'message-smtpmail-send-it)
(add-hook 'message-send-mail-hook 'mail-source-touch-pop)
```

これは、メールを送信するときはいつでも Gnus に前もって POP メールサーバーに接続させることを意味します。関数 `mail-source-touch-pop` は、メールを送信する直前に、メールを取得せずに `mail-sources` の値に従って POP 認証だけを行ないます。smtpmail-send-it ではなくて、`message-send-mail-hook` を実行する `message-smtpmail-send-it` を使わなければならないことと、POP 接続のために `mail-sources` の値を正しく設定しなければならないことに注意して下さい。See Section 6.3.4 [Mail Sources], page 148.

もし `mail-sources` に二つ以上の POP メールサーバーを設定しているならば、それらの一つを POP-before-SMTP 認証に使われる POP メールサーバーとして `mail-source-primary-source` に設定する必要があるでしょう。それが第一 POP メールサーバーならば (すなわち、主にそのサーバーからメールを取得しているならば)、それを以下のように恒久的に設定することができます:

```
(setq mail-source-primary-source
      '(pop :server "pop3.mail.server"
            :password "secret"))
```

さもなければ、POP-before-SMTP 認証を行なうときだけ、それを以下のように動的に束縛して下さい:

```
(add-hook 'message-send-mail-hook
  (lambda ()
    (let ((mail-source-primary-source
          '(pop :server "pop3.mail.server"
                :password "secret"))))
      (mail-source-touch-pop))))
```

5.4 メールと投稿

これはメールの送信とニュースの投稿の両方に関連する変数のリストです:

gnus-mailing-list-groups

あなたのニュースサーバーが、本当にメーリングリストの記事を NNTP サーバーに流し込むゲートウェイによって、それらがニュースグループの記事として見えるようにしているのであれば、それらのグループは問題なく読めるでしょう。しかしいくらか面倒なことを克服すること無しに、それらに投稿またはフォローアップすることはできません。一つの解決法は、グループパラメーター (see Section 2.10 [Group Parameters], page 23) に to-address を加えることです。簡単にできるのは、gnus-mailing-list-groups を、本当はメーリングリストであるグループに合致する正規表現に設定することです。そうすれば、少なくともメーリングリストへのフォローアップはたいいていに行なうことができるでしょう。これらのグループに投稿すること (a) は、それでも苦痛を引き起こすでしょうけれど。

gnus-user-agent

この変数は、どの情報が User-Agent ヘッダーに陳列されるかを制御します。シンボルのリスト、または文字列です。有効なシンボルは gnus (Gnus のバージョン) および emacs (Emacs のバージョン) です。Emacs のバージョンには codename ((S)XEmacs のコードネーム)、または config (system-configuration の値) か type (system-type の値) のどちらか一方を加えることができます。これを文字列にするときは、正しいフォーマットを使って下さい (RFC2616 参照)。

あなたは自分が送るメッセージで、綴りをチェックしたいかもしれません。もしくは手で綴りのチェックをしたくないのであれば、自動綴りチェックを ispell パッケージを使うことによって付け加えることができます:

```
(add-hook 'message-send-hook 'ispell-message)

ispell の辞書をグループに応じて切り替えたいならば、以下のようにすれば良いでしょう。

(add-hook 'gnus-select-group-hook
  (lambda ()
    (cond
      ((string-match
        "^de\\\\" (gnus-group-real-name gnus-newsgroup-name))
        (ispell-change-dictionary "deutsch"))
      (t
        (ispell-change-dictionary "english")))))
```

あなたの必要に応じて変更して下さい。

`gnus-message-highlight-citation` を `t` に設定すれば、`message` モードのバッファでも記事バッファと同様に、引用された文のレベルの違いに応じたハイライトが行なわれます。

5.5 メッセージの保管

Gnus はあなたが送ったメールとニュースを貯めておくためのいくつかの違った方法を提供します。デフォルトの方法はメッセージを保存するために「アーカイブ仮想サーバー」を使うことです。これを完全に禁止したいのであれば、変数 `gnus-message-archive-group` を `nil` にしなければなりません。それがデフォルトです。

グループで読んだ興味のあるメッセージの保存については、`B c` (`gnus-summary-copy-article`) コマンドを参照して下さい (see Section 3.25 [Mail Group Commands], page 103)。

`gnus-message-archive-method` は、送ったメッセージを保存するためにどの仮想サーバーを Gnus が使うかを指定します。デフォルトは:

```
(nnfolder "archive"
  (nnfolder-directory "~/Mail/archive")
  (nnfolder-active-file "~/Mail/archive/active")
  (nnfolder-get-new-mail nil)
  (nnfolder-inhibit-expiry t))
```

しかし、どのメール選択方法でも使うことができます (`nnml`, `nnmbox` などなど)。しかし `nnfolder` はこのようなことをするのにとても好ましい選択方法です。デフォルトで選択されるディレクトリーを好きでないならば、次のようにできます:

```
(setq gnus-message-archive-method
  '(nnfolder "archive"
    (nnfolder-inhibit-expiry t)
    (nnfolder-active-file "~/News/sent-mail/active")
    (nnfolder-directory "~/News/sent-mail/")))
```

訳注: 上記のような例は「意図した通りに動作しない」FAQ のネタになりつつあり、不具合の原因が特定できない事例が少なくありません。例えば、同じ `"archive"` という名前の仮想サーバーを過去に使ったことがあると、それが `~/newsrsrc.eld` ファイルの中で `gnus-server-alist` 変数に登録されているかもしれません。それを削除するには `~/newsrsrc.eld` ファイルを手作業で書き換えなければなりませんが、かなり危険です。あるいは単に、同名の仮想サーバーを現在も使っているかもしれません。そのような場合は、別の名前を使う必要があります。

Gnus は外へ出て行くすべてのメッセージに、`gnus-message-archive-method` で指定されたアーカイブ仮想サーバーにある (あるいはそれ以外のサーバーにある) 一つかそれ以上のグループに保存することを意図した `Gcc` 欄を挿入します。どのグループを使うかは変数 `gnus-message-archive-group` によって決まります。

この変数 (`gnus-message-archive-group`) は次のようなことをするために使うことができます:

文字列 メッセージはそのグループに保存されます。

グループ名に選択方法を含めることができますが、そうするとそのメッセージは `gnus-message-archive-method` で指定した選択方法ではなくて、代わりにグループ名の選択方法で保存されることに注意しましょう。 `gnus-message-archive-method` は、上に示したようなデフォルト値を持つためのものであると考えて下さい。ですから `gnus-message-archive-group` を `"foo"` にしておけば、外へ出て行くメッセージは `'nnfolder+archive:foo'` に保存されますが、`"nnml:foo"` という値を使うと、外へ出て行くメッセージは `'nnml:foo'` に保存されるでしょう。

文字列のリスト

メッセージはそれらのすべてのグループに保存されます。

正規表現、関数、Lisp フォームの連想リスト

キーが『合致』すると、その結果が使われます。

訳注: 正確には以下の三種類です。

- 正規表現とグループ名 (または複数のグループ名リスト) の連想リスト。最初に正規表現が合致した要素のグループ名 (またはグループ名のリスト) が使われます。
- 関数のリスト。それぞれの関数には現在のグループ名が引数として与えられ、最初に返ってきた nil 以外の値が使われます。
- Lisp フォームのリスト。それぞれのフォームが評価され、最初に返ってきた nil 以外の値が使われます。

nil メッセージの保存は行なわれません。これがデフォルトです。

例をあげてみましょう:

‘MisK’ という単一のグループに保存するだけならば:

```
(setq gnus-message-archive-group "MisK")
```

二つのグループ、‘MisK’ と ‘safe’ に保存するならば:

```
(setq gnus-message-archive-group '("MisK" "safe"))
```

どのグループにいるかによって違ったグループに保存するなら:

```
(setq gnus-message-archive-group
      '(("^alt" "sent-to-alt")
        ("mail" "sent-to-mail")
        (".*" "sent-to-misc")))
```

もっと複雑なもの:

```
(setq gnus-message-archive-group
      '((if (message-news-p)
            "misc-news"
            "misc-mail")))
```

すべてのニュースメッセージを一つのファイルに保存して、メールメッセージを一月につき一つのファイルに保存するというのはどうでしょう:

```
(setq gnus-message-archive-group
      '((if (message-news-p)
            "misc-news"
            (concat "mail." (format-time-string
                        "%Y-%m" (current-time))))))
```

さあ、メッセージを送ると適切なグループに保存されるようになりました。(もし特定のメッセージを保存をしたくないのであれば、挿入された Gcc 欄を取り除いて下さい。) 保管グループは次に Gnus を起動したときか、次にグループバッファで F を押したときにグループバッファに現れます。他のグループと同じように、そのグループに入って記事を読むことができます。そのグループが本当に大きくて悩ましくなったら、なにか良いものにその名前を変更することができます (グループバッファで Gr を使うことによって) — ‘misc-mail-september-1995’ その他何でも。新しいメッセージは古い (今は空になった) グループに溜められます。

以上が送ったメッセージを保管するデフォルトの方法です。Gnus はデフォルトの方法を好きではない人には違ったやり方を勧めています。そのような場合は、`gnus-message-archive-group` を `nil` に設定するべきです。これは保管をしないようにします。

`gnus-outgoing-message-group`

すべての外にいくメッセージはこのグループに入れられます。もしすべての外に行くメールと記事を `'nnml:archive'` グループに保管したいのであれば、この変数をその値に設定して下さい。この変数はグループ名のリストであることもできます。

もしそれぞれのメッセージをどのグループに入れるかをもっと制御したいのであれば、この変数を現在のニュースグループ名を調べて、適切なグループ名 (もしくは名前のリスト) を返す関数に設定することができます。

この変数は `gnus-message-archive-group` の代わりに使うことができますが、後者の方が好ましい方法です。

(訳注: 「後者」とは `gnus-message-archive-group` のこと。前者より四ヶ月遅れて、1996 年 1 月に新設されました。)

`gnus-gcc-mark-as-read`

もし非-`nil` なら、Gcc の記事に既読の印を付けます。

`gnus-gcc-externalize-attachments`

`nil` だったら、ファイルを通常のパートとして Gcc で保存する記事のコピーに添付します。それが正規表現で Gcc のグループ名に合致する場合は、外部パートとしてファイルを添付します。`all` だったらローカルファイルを参照する外部パートとして添付します。それが別の非-`nil` だった場合の動作は `all` のときと同じですが、将来は変わるかもしれません。

(訳注: 送信したメッセージと同じものを Gcc で保存する代わりに、添付ファイルをメッセージから切り離して、別にセーブするかどうかを制御する変数です。)

5.6 投稿様式

それらはすべて変数で、私の頭をくらくらさせます。

投稿するグループによって違った Organization と署名を付けたいんですか? そして、家のマシンと職場のマシンの両方から投稿するけれども、違った From 行などを使いたいんですか? そんなこと、どうでもいいじゃありませんか。

そのようなことをする方法の一つは、変更する必要がある変数を変更する賢いフックを書くことです。それは少し退屈なので、利用者にこれらのことを手軽な連想リストで指定するというすばらしい着想にたどり着いた人がいました。これが変数 `gnus-posting-styles` の例です:

```
(((".".*"
  (signature "Peace and happiness")
  (organization "What me?"))
  ("^comp"
  (signature "Death to everybody"))
  ("comp.emacs.i-love-it"
  (organization "Emacs is it"))))
```

この例から推測されるように、この連想リストはいくつかの「様式」(style) からなっています。それぞれの様式は最初の要素が何らかの形で「合致」したときに適用されます。連想リスト全体は最初から最後まで反復して実行され、それぞれの合致が適用されます。これは、後の様式の属性が前に

合致した様式の属性よりも優先されるということです。ですから ‘comp.programming.literate’ は、‘Death to everybody’ という署名と ‘What me?’ という Organization ヘッダーを持ちます。

それぞれの様式の最初の要素は マッチ (match) と言います。もしそれが文字列であれば、Gnus はそれを正規表現であるものとして、グループ名に合致するかどうかを調べます。(header 合致 正規表現) という形式であれば、Gnus は元記事の中からその名前が 合致 であるヘッダーを探し、それを 正規表現 と比較します。合致 と 正規表現 は文字列です。(元記事とは、あなたがそれに対して返信またはフォローアップしようとしている対象の記事です。返信あるいはフォローアップを作成していなければ、合致するものは何もありません。) もし マッチ が関数のシンボルであれば、その関数が引数無しで呼ばれます。それが変数のシンボルであれば、その変数が参照されます。それがリストであれば、そのリストが 評価 されます。どの場合でも、これが nil でない値を返せば、様式は 合致した と言います。

それぞれの様式は任意の量の「属性」を持つことができます。それぞれの属性は (name value) の対により成り立っています。加えて (name :file value) の形式か (name :value value) の形式を使うこともできます。ここで :file は value がファイル名を表して、その内容が属性値として使用されるべきであることを示し、:value は value がファイル名を表わさないことを明示的に示します。属性名 (name) は、以下のどれかであることができます。

- signature
- signature-file
- x-face-file
- address (user-mail-address よりも優先されます)
- name ((user-full-name) よりも優先されます)
- body

signature-file 属性は message-signature-directory 変数を見ることに注意して下さい。

属性名は文字列またはシンボルであることもできます。その場合それはヘッダー名として使われ、その値が記事のヘッダーに挿入されます。もし属性名が nil だったら、そのヘッダー名は削除されます。もし属性名が eval だったらその様式が評価され、結果は捨てられます。

属性値は文字列 (そのまま使われます)、引数の無い関数 (返り値が使われます)、変数 (その値が使われます) またはリスト (それは 評価 されて、返り値が使われます) であることができます。関数と S 式 (sexp) はセットアップされつつあるメッセージバッファで呼ばれるか評価されます。現在の記事のヘッダー群は変数 message-reply-headers から得られます。これは number subject from date id references chars lines xref extra の各ヘッダーから成るベクトルです。

作成しようとしているメッセージがニュース記事かメールメッセージであるかを調べたいときは、関数 message-news-p と message-mail-p の戻り値を調べて下さい。

そして、これは例です:

```
(setq gnus-posting-styles
  '((".*"
     (signature-file "~/signature")
     (name "User Name")
     (x-face-file "~/xface")
     (x-url (getenv "WWW_HOME"))
     (organization "People's Front Against MWM"))
    ("^rec.humor"
     (signature my-funny-signature-randomizer)))
```

```

(equal (system-name) "gnarly") ;; 様式
(signature my-quote-randomizer))
(message-news-p ;; 関数シンボル
(signature my-news-signature))
(window-system ;; 変数シンボル
("X-Window-System" (format "%s" window-system)))
;; Lars さんに返事をするときは
;; Organization ヘッダーを付けよう。
(header "to" "larsi.*org")
(Organization "Somewhere, Inc.))
(posting-from-work-p) ;; 利用者が定義した関数
(signature-file "~/work-signature")
(address "user@bar.foo")
(body "You are fired.\n\nSincerely, your boss.")
(organization "Important Work, Inc"))
("nnml:.*"
(From (save-excursion
(set-buffer gnus-article-buffer)
(message-fetch-field "to"))))
("^nn.+:"
(signature-file "~/mail-signature"))))

```

‘nnml:.*’の規則は、あなたが出すすべての返事の To アドレスを From アドレスとして使うことを意味します。これは、あなたがたくさんのメーリングリストに参加している場合に便利でしょう。代わりに message-alternative-emails を使うこともできます。See section “メッセージヘッダー” in *The Message Manual*.

5.7 下書き

メッセージ（メールもしくはニュース）を書いているときに、オープンにステーキが入っている（もしくはあなたがとってもすごい菜食主義者で、何かのペーストがフードプロセッサに入っている）ことを突然思い出したなら、書いているメッセージを保存する方法があれば良いと思うでしょう。いつか別の日に編集を続けることができ、それが完成したと思ったときに送ることができるように。

ええ、心配しないで下さい。メールかニュースを送信するための Gnus の命令を使って何らかのメッセージを書き始めたときにあなたが手にするバッファは、自動的に特別な *draft* グループに関連付けられます。普通の方法（例えば *C-x C-s*）でバッファを保存すれば、その記事はそこに保存されます。（自動保存（auto-save）ファイルも下書きグループ（draft group）に行きます。）

下書きグループは ‘nndraft:drafts’ と呼ばれる特別なグループです（あなたが絶対に知っていなければならないのであれば、それは nndraft グループとして実装されています）。変数 nndraft-directory は nndraft がそのファイルをどこに保管するかを指定します。このグループを特別なものにしているのは、その中の記事に可視や既読の印を付けることができないことです—そのグループのすべての記事は永久に未読です。

もしグループが存在しないと、それは作成され、購読させられます。グループバッファからそれを消し去る唯一の方法は、それを購読しないようにすることです。下書きグループの特別の特性はグループの特性（see Section 2.10 [Group Parameters], page 23）によって生じ、それが失われてしまうと他のグループのように振る舞うようになります。これは（グループの特性を消してしまうことは）以下のコマンドが使えないことを意味します。そのグループの特別の特性を復活させる最も簡単

な方法は、*C-k* でそのグループを削除してから Gnus を再起動することです。そのグループの内容物は失われません。

記事の編集を続けたいときは、下書きグループに入って *De* (gnus-draft-edit-message) を押すだけです。編集を中断したときの状態のバッファーに移動します。

送信を拒否された記事も、この下書きグループに入れられます (see Section 5.8 [Rejected Articles], page 131)。

送信を拒否されたメッセージがたくさんあって、それ以上編集せずにそれらを送信したい場合は、*Ds* 命令を使うことができます。この命令はプロセス/ 接頭引数の習慣を理解します (see Section 8.1 [Process/Prefix], page 249)。*DS* 命令 (gnus-draft-send-all-messages) はバッファーのすべてのメッセージを送り出します。

送りたいくないメッセージがいくつかあるのであれば、*Dt* 命令 (gnus-draft-toggle-sending) を使ってメッセージに送信不可の印を付けることができます。これは切り替え命令です。

5.8 拒否された記事

時々ニュースサーバーは記事を送信することを拒否します。おそらくサーバーはあなたの顔を好きではないのでしょう。おそらく落ち込んでいるのでしょう。おそらく 悪魔 (*demon*) がいるのでしょう。おそらく引用文を入れすぎたのでしょう。おそらくディスクが一杯だったのでしょう。おそらくサーバーが落ちていたのでしょう。

もちろんこれらの状況は完全に Gnus の扱える範囲外です。(もちろん Gnus はあなたの風貌を愛しているし、いつも機嫌が良いし、中を飛び回る天使がいて、どれくらい引用文が含まれていようと気にせず、一杯になったり、落っこちたりしません。) ですから Gnus はこれらの記事を後でサーバーの機嫌が良くなるまで保存します。

拒否された記事は自動的に特別な下書きグループ (see Section 5.7 [Drafts], page 130) に入れます。サーバーが復旧した暁には、普通あなたはそのグループに入って、すべての記事を送ることになるでしょう。

5.9 署名と暗号化

素の PGP 形式、PGP/MIME または S/MIME を使って、Gnus はメッセージに電子署名したり暗号化することができます。そのようなメッセージのデコードに関しては、*mm-verify-option* オプションおよび *mm-decrypt-option* オプション (see Section 3.30 [Security], page 111) を参照して下さい。

署名したメッセージを送ってきた人たちに、署名した返信を返したいことはしばしばあります。さらに暗号化されたメッセージへの返信を暗号化したいことは、もっとたびたびあるかもしれません。Gnus は前者のために *gnus-message-replysign* の機能を、後者のために *gnus-message-replyencrypt* の機能を提供します。さらに *gnus-message-replysignencrypt* を設定することによって (デフォルトで on になっています)、暗号化したメッセージに自動的に署名もします。

MIME パートに対してセキュリティーの操作を行なうための MML への指示は、以下のように署名の場合は *C-c C-m s* キーマップを使って、暗号化の場合は *C-c C-m c* キーマップを使って行ないます。

C-c C-m s s

S/MIME を使って現在のメッセージに電子署名します。

C-c C-m s o

PGP を使って現在のメッセージに電子署名します。

C-c C-m s p

PGP/MIME を使って現在のメッセージに電子署名します。

C-c C-m c s

S/MIME を使って現在のメッセージを電子暗号化します。

C-c C-m c o

PGP を使って現在のメッセージを電子暗号化します。

C-c C-m c p

PGP/MIME を使って現在のメッセージを電子暗号化します。

C-c C-m C-n

メッセージから、セキュリティー関連の MML タグを外します。

もっと詳しいことは See section “セキュリティー” in *The Message Manual*, を参照して下さい。

6 選択方法

「外部グループ」(foreign group) とは、普通 (もしくはデフォルト) の方法で読めないグループのことです。例えばそれは別の NNTP サーバーのグループであったり、仮想グループであったり、個人的なメールグループであったりするでしょう。

外部グループ (あるいは実際にどんなグループでも) は「名前」と「選択方法」で指定されます。先に後者を例に出すと、選択方法はリストで、最初の要素がどのバックエンドを使うか (例えば `nntp`, `nnspool`, `nnml`) を、二つめの要素が「サーバー名」を表します。選択方法には、その当のバックエンドにとって特別の意味を持つ値である追加の要素があるかもしれません。

選択方法とは「仮想サーバー」を定義することだ、とすることができます—ですから私たちはまさにそれをしました (see Section 6.1 [Server Buffer], page 133)。

グループの「名前」は、バックエンドがそのグループを認識する名前です。

例えば `'some.where.edu'` という NNTP サーバーにある `'soc.motss'` グループは、名前 `'soc.motss'` と選択方法 (`nntp "some.where.edu"`) を持ちます。`nntp` バックエンドはこのグループを `'soc.motss'` として知っているだけですが、Gnus はこのグループを `'nntp+some.where.edu:soc.motss'` と呼びます。

もちろん、違った方法はすべてそれ特有の要素を持っています。

6.1 サーバーバッファ

伝統的に、「サーバー」は誰かがそれに接続して、それからの情報を要求するマシンかソフトウェアの断片です。Gnus は実際のどんなサーバーにも直接には接続せず、何かのバックエンドを通してすべての処理を行ないます。しかしそれはまさしく実際の媒体と Gnus の間に一つ以上の階層を置くことであって、ちょうどそれぞれのバックエンドが疑似的なサーバーに相当すると言っても良いでしょう。

例えば `nntp` バックエンドは、複数の別々に実在する NNTP サーバー、あるいは実在する同じ NNTP サーバーの異なるポートに接続するために用いられます。あなたはどのバックエンドを使うか、そしてどんなパラメーターを設定するかを選択方法 (*select method*) に設定して Gnus に指示します。

選択方法の指定は、ときに極めて面倒なものになります—えーと、例えば `'news.funet.fi'` という NNTP サーバーのポート 13 を読みたいのだけれど、NOV ヘッダーを取り寄せようとするとうまく固まってしまうし、間違った記事を選択してしまうような場合です。うおっほん。とにかくこのサーバーを使うそれぞれのグループについてそういうことを設定しなければならないとしたら、大変な作業になってしまうでしょう。そこで Gnus は、そういう作業をサーバーバッファで行なうために、選択方法に名前を付ける手段を設けているのです。

サーバーバッファに入るためには、グループバッファで `^` (`gnus-group-enter-server-mode`) コマンドを使って下さい。

サーバーバッファを作成するときに `gnus-server-mode-hook` が実行されます。

6.1.1 サーバーバッファの表示様式

サーバーバッファの行の外見を、変数 `gnus-server-line-format` 変数を変更することによって変えることができます。これは `format` のような変数で、少しばかり単純な拡張がなされています:

- 'h' どのようにニュースが取得されるか—バックエンドの名前。
- 'n' サーバーの名前。
- 'w' どこからニュースが取得されるか—アドレス。

- ‘s’ サーバーの接続の 開いた/閉じた/拒否された 状態。
- ‘a’ そのサーバーがエージェント化されているかどうか。

モード行も変数 `gnus-server-mode-line-format` を使うことによってカスタマイズすることができます (see Section 8.4.2 [Mode Line Formatting], page 251)。

[訳注: 現在この変数は使われていません。]

以下の仕様が理解されます:

- ‘S’ サーバー名。
- ‘M’ サーバーの選択方法。

Section 8.4 [Formatting Variables], page 250 も参照して下さい。

6.1.2 サーバー命令

- `v` `v` キーはユーザー用に予約されています。そのまま何かの機能に割り当てても構いませんが、接頭キーとして使う方が良いでしょう。
- `a` 新しいサーバーを追加します (`gnus-server-add-server`)。
- `e` サーバーを編集します (`gnus-server-edit-server`)。
- `SPACE` 現在のサーバーを眺めます (`gnus-server-read-server`)。
訳注: 実際には `gnus-server-read-server-in-server-buffer` 命令を呼びますが、`gnus-server-browse-in-group-buffer` の値がデフォルトの `nil` であれば `gnus-server-read-server` と同じです。`gnus-server-browse-in-group-buffer` を `nil` 以外の値にすることはまったくお勧めできませんが、あなたが何をするのも自由です。詳細はソースコードを読むか、実際に試して痛い目に会って下さい。;-p
- `q` グループバッファに戻ります (`gnus-server-exit`)。
- `k` 現在のサーバーを切り取ります (kill します) (`gnus-server-kill-server`)。
- `y` 先ほど切られた (killed) サーバーを貼り付けます (yank します) (`gnus-server-yank-server`)。
- `c` 現在のサーバーを複写します (`gnus-server-copy-server`)。
- `l` すべてのサーバーの一覧を表示します (`gnus-server-list-servers`)。
- `s` サーバーにそのソースから新しい記事を調べるように要求します (`gnus-server-scan-server`)。主にメールサーバーが意味のある動作をします。
- `g` サーバーにすべてのデータ構造を再作成させます (`gnus-server-regenerate-server`)。これは同期が外れてしまったメールバックエンドがあるときに役に立ちます。
- `z`
現在の位置のサーバーのすべてのグループを圧縮します。今のところ `nnml` (see Section 6.3.13.3 [Mail Spool], page 169) だけに実装されています。これは記事番号のすきまを取り除くので、正しい全記事数を得ることができるようになります。

6.1.3 方法の例

ほとんどの選択方法は、説明する必要が無いくらいにかなり単純です:

```
(nntp "news.funet.fi")
```

直接スプールから読むのはもっと単純です:

```
(nnspool "")
```

見ての通り、選択方法の最初の要素はバックエンドの名前で、二番目は「アドレス」(address)、もしくはそう呼びたいのであれば「名前」です。

これらの二つの要素の後には、任意の数の(変数 様式)の対を置くことができます。

最初の例に戻りましょう—そのマシンのポート 15 から読みたいのだと思って下さい。これがその時に、そうなるはずの選択方法です:

```
(nntp "news.funet.fi" (nntp-port-number 15))
```

どの変数が関連するかを見つけ出すために、それぞれのバックエンドの説明文書を読むべきでしょうが、これは nnmh の例です。

nnmh はスプールのような構造を読むためのメールバックエンドです。例えばアクセスしたい二つの構造があるとしましょう: 一つはあなたの私的なメールスプールで、他方は公的なものです。これは私的なメールのために使うことができる指定です:

```
(nnmh "private" (nnmh-directory "~/private/mail/"))
```

(それでこのサーバーは 'private' と呼ばれますが、あなたはすでに推測していたかもしれませんね。)

これは公的なスプールのための方法です:

```
(nnmh "public"
  (nnmh-directory "/usr/information/spool/")
  (nnmh-get-new-mail nil))
```

あなたが防壁 (firewall) の中にいて、防壁マシンを通して NNTP サーバーに接続するしかないのであれば、防壁マシンに rlogin して、そこから NNTP サーバーに telnet をするように Gnus に指示することができます。こんなことをするのはいささかばかげているのですが、でも仮想サーバーの定義はおそらくこのようなものになるはずです:

```
(nntp "firewall"
  (nntp-open-connection-function nntp-open-via-rlogin-and-telnet)
  (nntp-via-address "the.firewall.machine")
  (nntp-address "the.real.nntp.host")
  (nntp-end-of-line "\n"))
```

あの素敵な ssh プログラムを、モデムを経由する通信を圧縮するために使いたいのであれば、上記の例に以下の設定を加えることができます。

```
(nntp-via-rlogin-command "ssh")
```

nntp-via-rlogin-command-switches も参照して下さい。間接的に接続する場合の例です:

```
(setq gnus-select-method
  '(nntp "indirect"
    (nntp-address "news.server.example")
    (nntp-via-user-name "intermediate_user_name")
    (nntp-via-address "intermediate.host.example")
    (nntp-via-rlogin-command "ssh")))
```

```
(nntp-end-of-line "\n")
(nntp-via-rlogin-command-switches ("-C" "-t" "-e" "none"))
(nntp-open-connection-function nntp-open-via-rlogin-and-telnet)))
```

防壁の中にいたとしても "runsocks" のようなラッパーコマンドを通して外の世界に直接アクセスできるのならば、以下のように socks 化された telnet でニュースサーバーに接続することができるとでしょう:

```
(nntp "outside"
  (nntp-pre-command "runsocks")
  (nntp-open-connection-function nntp-open-via-telnet)
  (nntp-address "the.news.server")
  (nntp-end-of-line "\n"))
```

もちろん、自動認証を行なわせるためには ssh-agent を適切に設定しなければなりません。加えて、通信内容を圧縮するためには、ssh の 'config' ファイルに 'Compression' オプションが存在しなければなりません。

6.1.4 仮想サーバーを作成する

永続記事を使ってたくさんの記事をキャッシュに保存しているのであれば、キャッシュを読むための仮想サーバーを作る必要があるでしょう。

最初に新しいサーバーを追加する必要があります。それをするのは a 命令です。おそらくキャッシュを読むためには nnml を使うのが一番良いでしょう。nnspool や nnmh も使えるでしょうけれど。

a nnml RET cache RET とタイプして下さい。

今やあなたは真新しい 'cache' という nnml の仮想サーバーを手に入れたはずで、次はそれを編集して、正しい定義を与えましょう。サーバーを編集するには e をタイプして下さい。あなたは以下のものを含むバッファに入ります:

```
(nnml "cache")
```

それを次のように変更して下さい:

```
(nnml "cache"
  (nnml-directory "~/News/cache/")
  (nnml-active-file "~/News/cache/active"))
```

サーバーバッファに戻るには C-c C-c をタイプして下さい。今ではこの仮想サーバーで RET を押すと、閲覧バッファに入って、表示されているどのグループにでも入ることができるはずです。

6.1.5 サーバー変数

変数を (バックエンドと Emacs 一般の両方で) 定義する際の一つのやっかいな点は、いくつかの変数は、概してその変数の定義がロードされるときに他の変数で初期化されることです。「基」になる変数がロードされた後でそれを変更しても、「派生」した変数は変更されません。

これは一般にディレクトリーやファイルの変数に影響します。例えば nnml-directory はディフォルトでは '~/Mail/' で、また、すべての nnml ディレクトリー変数はその変数によって初期化されるので、nnml-active-file は '~/Mail/active' になります。新しい nnml 仮想サーバーを定義する場合、nnml-directory を設定するだけでは十分ではありません—あなたはすべてのファイル変数を、そうしたいと望んだ値に明示的に設定しなければなりません。それぞれのバックエンドのための完全な変数のリストを見るには、このマニュアルの後に続くそれぞれのバックエンドの部分を読んで下さい。でも nnml の定義の例はここにあります:

```
(nnml "public"
  (nnml-directory "~/my-mail/")
  (nnml-active-file "~/my-mail/active")
  (nnml-newsgroups-file "~/my-mail/newsgroups"))
```

サーバー変数はしばしば「サーバーパラメーター」と呼ばれます。

6.1.6 サーバーと選択方法

普通に選択方法を使う（例えば外部サーバーから記事を読むときにグループを選択する手段として `gnus-secondary-select-method` 使う）場面ではどこでも、代わりに仮想サーバーの名前を使うことができます。これによって、たくさんキーボードを叩かなくて済むかもしれません。そして、どんなときでもその方が良いです。

6.1.7 使用不可能なサーバー

あるサーバーに接続することができないように見えるとき、Gnus はそのサーバーに拒否された (denied) ことを記録します。その後でそのサーバーと接続しようとするどんな試みも、単に無視されます。実際にそうかどうかを少しも確かめずに、Gnus は「接続を開くことができません」と (英語で) 告げます。

それはずいぶんお行儀が悪いと思うかもしれませんが、たいていの場合は有意義なのです。例えば `'nephelococcdyia.com'` というサーバーで十個のグループを購読しているとしましょう。サーバーはどこかとても遠いところにあって、そのマシンはとても遅いので、今日それが接続を拒否するかどうかを調べるだけでも一分かかります。もし Gnus がそれを十回試すようになっていたとすると、とても煩わしいでしょう。ですから Gnus はそれを試そうとはしません。一度でも「接続が拒否された」 (connection refused) という結果を受け取ったなら、それはサーバーが「落ちている」 (down) のだ、とみなします。

では、一時的にそのマシンの機嫌が悪いだけだったら何が起こるのでしょうか？ マシンが復活したかどうかをどうすれば調べることができるのでしょうか？

それには、サーバーバッファーに移動して (see Section 6.1 [Server Buffer], page 133)、以下の命令で突いてみて下さい:

- D* 現在の行のサーバーとの接続を確立しようとします (`gnus-server-open-server`)。
- C* サーバーとの接続 (もしあれば) を閉じます (`gnus-server-close-server`)。
- D* 現在のサーバーに接続不可の印を付けます (`gnus-server-open-all-server`)。
- M-o* バッファーにあるすべてのサーバーとの接続を開きます (`gnus-server-open-all-servers`)。
- M-c* バッファーにあるすべてのサーバーとの接続を閉じます (`gnus-server-close-all-servers`)。
- R* Gnus が接続を拒否されたすべてのサーバーの、すべての印を消去します (`gnus-server-remove-denials`)。
- L* サーバーの状態をオフラインにします (`gnus-server-offline-server`)。

6.2 ニュースの取得

ニュースリーダーは普通はニュースを読むために使われます。Gnus は現在はニュースを取得するための二つの方法だけを提供しています—NNTP サーバーから、またはローカルスプールから読むことができます。

6.2.1 NNTP

NNTP サーバーから外部グループを購読するのは比較的簡単です。単に選択方法として `nntp` を指定し、NNTP サーバーのアドレスを、うーん、アドレスとして指定するだけです。

NNTP サーバーが標準ではないポート (port) に設置されているときは、選択方法の三番目の要素をこのポートの番号に設定すれば、正しいポートに接続することができますでしょう。そのためにはグループ情報を編集しなければなりません (see Section 2.9 [Foreign Groups], page 21)。

外部グループの名前は基本グループと同じでも構いません。実際、あなたの思うままに同じグループを可能な限りの違ったサーバーから購読することができます。名前の衝突は起こりません。

以下の変数は仮想 `nntp` サーバーを作るために使われます:

`nntp-server-opened-hook`

は接続ができた後に実行されます。それは NNTP サーバーに接続した後に、それに命令を送るために使うことができます。デフォルトでは `MODE READER` 命令が、`nntp-send-mode-reader` 関数によってサーバーに送られるようになっています。この関数は常にこのフックにあるべきです。

`nntp-authinfo-function`

この関数は NNTP サーバーに 'AUTHINFO' を送るために使われます。デフォルトの関数は `nntp-send-authinfo` で、適切な記載事項を探すために '~/.authinfo' (もしくは `nntp-authinfo-file` 変数に設定した何でも) を調べます。もし一つも見つからなかったら、ログイン名とパスワードの入力を要求します。 '~/.authinfo' ファイルの様式は ftp のための '~/.netrc' ファイルと (ほとんど) 同じです。それは ftp のマニュアルページで定義されていますが、ここに顕著な実例があります:

1. ファイルは一つ以上の行を含み、それぞれは一つのサーバーを定義します。
2. それぞれの行は任意の数の標章 (token) と値の対を含むことができます。

有効な標章は 'machine', 'login', 'password', 'default' です。加えて、Gnus は '.netrc'/ftp の構文の原型には現れない二つの新しい標章、名付けて 'port' と 'force' を導入します。(これが '.authinfo' ファイルの様式が '.netrc' ファイルの様式から逸脱する唯一の方法です。) 'port' はサーバーのどのポートを認証に用いるかを示し、'force' は以下で説明します。

これがそのファイルの例です:

```
machine news.uio.no login larsi password geheimnis
machine nntp.ifi.uio.no login larsi force yes
```

標章と値の対はどんな順番でも現れることができます。例えば 'machine' が最初でなければならぬ必要はありません。

この例では、前者のサーバーにログイン名とパスワードの両方が与えられているのに対して、後者にはログイン名だけがあり、利用者はパスワードの入力を求められるでしょう。後者は 'force' タグも持っていて、これによって接続時に `nntp` サーバーに認証情報 (authinfo) が送られます。デフォルト (すなわち、'force' タグが無いとき) では、`nntp` サーバーが認証情報を尋ねない限りそれを `nntp` サーバーに送りません。

'machine' 行に合致しないすべてのサーバーに適用される 'default' 行を追加することもできます。

```
default force yes
```

これは、それ以前に書かれていないすべてのサーバーに 'AUTHINFO' 命令を強制的に送ります。

‘~/.authinfo’ ファイルを世界中が読めるような設定のままで放置しないように注意して下さい。

nntp-server-action-alist

これはサーバーの型に合致する正規表現と、合致が起こったときに取られる動作の連想リストです。例えば、Gnus に innd に接続したときに毎回ビープ音を鳴らしたいのであれば、次のようにすることができます:

```
(setq nntp-server-action-list
      '(("innd" (ding))))
```

まあ、そんなことをしたいとは思わないでしょうけれどね。

デフォルトの値は

```
'(("nntpd 1\\.5\\.11t"
  (remove-hook 'nntp-server-opened-hook
                'nntp-send-mode-reader)))
```

で、これは nntpd 1.5.11t には MODE READER 命令を確実に送らないようにします。なぜなら、その命令はサーバーの息の根を止めると聞いているからです。

nntp-maximum-request

もし NNTP サーバーが NOV ヘッダーをサポートしていないのであれば、このバックエンドは head 命令をいくつも送って、ヘッダーを集めます。この動作を速くするために、バックエンドは返答を待たずにこの命令をたくさん送り、それからすべての返答を読みます。これは変数 nntp-maximum-request によって制御され、デフォルトで 400 です。もしネットワークの具合が良くないようなら、この変数を 1 に設定するべきでしょう。

nntp-connection-timeout

定期的に接続している外部 nntp グループがたくさんあると、ちゃんと応答しなかったり常識的な時間内に返答できないくらいの負荷がかかっている NNTP サーバーの問題があるはずです。これはやっかいな問題をもたらしますが、nntp-connection-timeout を設定することによってある程度解消することができます。これは接続を諦める前に、nntp バックエンドが何秒待つかを示す整数です。もしこれが nil であると、それがデフォルトですが、時間切れによる切断は行ないません。

nntp-nov-is-evil

NNTP サーバーが NOV をサポートしていない場合は、この変数を t に設定すれば良いでしょう。でも nntp は普通は NOV が使えるかどうかを自動的に調べます。(訳注: ですから、わざわざ設定しなくても構いません。)

nntp-xover-commands

サーバーから NOV 行を取得するための命令として使われる文字列のリストです。この変数のデフォルトの値は ("XOVER" "XOVERVIEW") です。(訳注: それらを順に試します。)

nntp-nov-gap

nntp は、普通はサーバーに NOV 行のための一つの大きな要求を送ります。サーバーは一つの巨大な行のリストで応答します。しかし、グループの 2-5000 の記事を読んだ後で 1 と 5001 を読みたいだけだとしても、nntp は必要の無い 4999 個の NOV 行を取得することになります。この変数は、どれくらい大きな二つの連続した記事群の間の隔たり (gap) まで XOVER の要求を分割せずに送るかを決定します。ネットワークが

速い場合に、この変数を本当に小さな数値に設定してしまうと、おそらく取得が遅くなることに注意して下さい。この変数が `nil` ならば、`nntp` は要求を分割しません。デフォルトは 5 です。

`nntp-prepare-server-hook`

NNTP サーバーに接続を試みる前に実行するフックです。

`nntp-record-commands`

これを `nil` でない値にすると、`nntp` は NNTP サーバーに送ったすべての命令を（時刻と共に）‘`*nntp-log*`’ バッファに記録します。これは動作していないように見える Gnus の NNTP 接続をデバッグしているときに役に立ちます。

`nntp-open-connection-function`

どのように NNTP サーバーと接続するかをカスタマイズすることができます。`nntp-open-connection-function` パラメーターを設定しておく、Gnus は接続を確立するためにその関数を使います。そのために七つの関数があらかじめ用意されています。それらは二種類に分類することができ、直接接続するための関数群（四つ）と間接的に接続するためのもの（三つ）があります。

`nntp-never-echoes-commands`

非-`nil` で NNTP サーバーがコマンドをエコーバックしないことを意味します。報告によると、ある種の NNTPS サーバーはコマンドをエコーバックしないそうです。したがって、例えば `nntp-open-connection-function` を `nntp-open-ssl-stream` に設定してあるそのようなサーバーのための選択方法の中で、この変数を非-`nil` に設定する必要があるでしょう。デフォルト値は `nil` です。この変数の値 `nil` は、`nntp-open-connection-functions-never-echo-commands` 変数でくつがえされることに注意して下さい。

`nntp-open-connection-functions-never-echo-commands`

コマンドをエコーバックしない関数のリストです。`nntp-open-connection-function` に設定した関数がコマンドをエコーバックしないならば、それをこのリストに加えて下さい。`nntp-never-echoes-commands` 変数の `nil` でない値が、この変数をくつがえすことに注意して下さい。デフォルト値は (`nntp-open-network-stream`) です。

`nntp-prepare-post-hook`

記事を投稿する直前に実行されるフックです。もし記事に Message-ID ヘッダーがなくてニュースサーバーが推奨 ID を提供してくれるならば、このフックが実行される前にそれが記事に加えられます。これは、もしあなたが Gnus が Message-ID ヘッダーを付けないようにしていても、Cancel-Lock ヘッダーを作るために利用することができます。それにはこうすれば良いでしょう：

```
(add-hook 'nntp-prepare-post-hook 'cancellock-insert-header)
```

すべてのサーバーが推奨 ID をサポートしているわけではないことに注意して下さい。これは例えば INN 2.3.0 以上で動作します。

6.2.1.1 直接接続するための関数

これらの関数は、あなたのマシンと NNTP サーバーを接続するために直接呼ばれます。また、それらの動作はそれらが共通に参照する変数に影響されます (see Section 6.2.1.3 [Common Variables], page 143)。

`nntp-open-network-stream`

これはデフォルトで、単純に遠隔システムの何らかのポートに接続します。

nntp-open-tls-stream

「安全な」チャンネルを使ってサーバーに接続します。これを使うためには GNUTLS (<http://www.gnu.org/software/gnutls/>) をインストールしておかなければなりません。それからサーバーを次のように定義します:

```
;; ポート 563 が "nntp" として '/etc/services' で定義済み
;; であっても、'gnutls-cli -p' でその名前は使えません。
;;
(nntp "snews.bar.com"
  (nntp-open-connection-function nntp-open-tls-stream)
  (nntp-port-number )
  (nntp-address "snews.bar.com"))
```

nntp-open-ssl-stream

「安全な」チャンネルを使ってサーバーに接続します。これを使うためには OpenSSL (<http://www.openssl.org>) または SSLeay (<ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL>) をインストールしておかなければなりません。それからサーバーを次のように定義します:

```
;; ポート 563 が "snews" として '/etc/services' で定義済みで
;; あっても、'openssl s_client -port' でその名前は使えません。
;;
(nntp "snews.bar.com"
  (nntp-open-connection-function nntp-open-ssl-stream)
  (nntp-port-number 563)
  (nntp-address "snews.bar.com"))
```

nntp-open-telnet-stream

単に 'telnet' して NNTP サーバーに接続します。デフォルトの nntp-open-network-stream がそれをするのにもかかわらず、なぜこの関数があるのか不思議に思うかもしれません。その理由 (の一つ) は、もしあなたが防壁の中にいたとしても runsocks のようなコマンドラッパーのおかげで外の世界を直接アクセスできるならば、それをこのように使うことができるのです:

```
(nntp "socksified"
  (nntp-pre-command "runsocks")
  (nntp-open-connection-function nntp-open-telnet-stream)
  (nntp-address "the.news.server"))
```

Emacs のセッション全体をラップして、デフォルトのメソッドを使うというのは、良い考えではありません。

6.2.1.2 間接的に接続するための関数

これらの関数は、実際に NNTP サーバーに接続する前に中間のホストに接続するために間接的に呼ばれます。すべてのこれらの関数と関連する変数は“ via ”接続の仲間に属しているとも言えるので、それを明確にするためにすべて“ via ”という接頭語が付けられます。また、それらの動作はそれらが共通に参照する変数に影響されます (see Section 6.2.1.3 [Common Variables], page 143)。

nntp-open-via-rlogin-and-telnet

遠隔システムに 'rlogin' して、そこから本当の NNTP サーバーに 'telnet' します。これは、例えばあなたが始めに防壁マシンに接続しなければならない場合に便利です。

nntp-open-via-rlogin-and-telnet-用の変数:

`nntp-via-rlogin-command`

中間のホストにログインするために使われるコマンドです。デフォルトは `'rsh'` ですが、`'ssh'` が人気のある代替手段です。

`nntp-via-rlogin-command-switches`

`nntp-via-rlogin-command` のコマンドのスイッチとして使われる文字列のリストです。デフォルトは `nil` です。もし `'ssh'` を `nntp-via-rlogin-command` の値として使うならば、データ接続を圧縮するために `'("-C")'` を使うことができます。あるいは、もし中間のホストで `telnet` コマンドが疑似端末を必要とするならば、これを `'("-t" "-e" "none")'` または `'("-C" "-t" "-e" "none")'` にして下さい。

`nntp-end-of-line` の値を `'\n'` に変更する必要があるであろうことに注意して下さい (see Section 6.2.1.3 [Common Variables], page 143)。

`nntp-open-via-rlogin-and-netcat`

本質的には同じことなのですが、中間のホストから本当の NNTP サーバーに接続するために、`'telnet'` の代わりに `netcat` (<http://netcat.sourceforge.net/>) を使います。

`nntp-open-via-rlogin-and-netcat`-用の変数:

`nntp-via-netcat-command`

中間のホストから本当の NNTP サーバーに接続するために使われるコマンドです。デフォルトは `'nc'` です。代わりに `connect` (<http://www.imasy.or.jp/~gotoh/ssh/connect.html>) のような、他のコマンドを使うこともできます。

`nntp-via-netcat-switches`

`nntp-via-netcat-command` のコマンドのスイッチとして使われる文字列のリストです。デフォルトは `nil` です。

`nntp-via-rlogin-command`

中間のホストにログインするために使われるコマンドです。デフォルトは `'rsh'` ですが、`'ssh'` が人気のある代替手段です。

`nntp-via-rlogin-command-switches`

`nntp-via-rlogin-command` のコマンドのスイッチとして使われる文字列のリストです。デフォルトは `nil` です。

`nntp-open-via-telnet-and-telnet`

これもまた本質的には同じことなのですが、中間のホストに接続するために `'rlogin'` の代わりに `'telnet'` を使います。

`nntp-open-via-telnet-and-telnet`-用の変数:

`nntp-via-telnet-command`

中間のホストに `telnet` するために使われるコマンドです。デフォルトは `'telnet'` です。

`nntp-via-telnet-switches`

`nntp-via-telnet-command` のコマンドのスイッチとして使われる文字列のリストです。デフォルトは `'("-8")'` です。

`nntp-via-user-password`

中間のホストにログインするときに使われるパスワードです。

`nntp-via-envuser`

もし非-nil なら、中間の telnet のセッション (クライアントとサーバーの両方) で ENVIRON オプションをサポートし、ログイン名の入力を要求しません。これは例えば Solaris の telnet で動作します。

`nntp-via-shell-prompt`

中間のホストでのシェルのプロンプトに合致する正規表現です。デフォルトは `'bash\\|\\$ *\\r?\\$\\|> *\\r?'` です。

`nntp-end-of-line` の値を `'\n'` に変更する必要があるであろうことに注意して下さい (see Section 6.2.1.3 [Common Variables], page 143)。

これらは上記のすべての関数が参照する付加的な変数です:

`nntp-via-user-name`

中間のホストに接続するときに使う利用者名です。

`nntp-via-address`

接続する中間のホストのアドレスです。

6.2.1.3 共通の変数

以下の変数は、すべての、またはいくつかのあらかじめ用意されている関数の動作に影響を及ぼします。設定されていなければ、すべての関数が影響されます (それぞれの仮想サーバーにおいて、サーバー変数として個々に値が設定されていない場合に、以下の値がデフォルトで使われます)。

`nntp-pre-command`

素の接続用の関数ではないもの (`nntp-open-network-stream`、`nntp-open-tls-stream` または `nntp-open-ssl-stream` 以外のすべて) を通して接続するときに使うコマンドラッパーです。例えばあなたは `'SOCKS'` ラッパーを割り当ててください。(訳注: telnet などの外部コマンドに被せて使われます。)

`nntp-address`

NNTP サーバーのアドレスです。

`nntp-port-number`

接続する NNTP サーバーのポート番号です。デフォルトは `'nntp'` です。TLS/SSL を介した NNTP を使うには、ポートの名前ではなくて整数 (つまり `'snews'` や `'nntps'` ではなくて `'563'`) を指定する必要があります。外部の TLS/SSL ツールはポートの名前では動作しないからです。

`nntp-end-of-line`

NNTP サーバーとお話をしているときに行の終わりの印として使われる文字列です。これはデフォルトで `'\r\n'` ですが、素ではない接続用の telnet 同等の関数を使っているときは `'\n'` であるべきです。

`nntp-telnet-command`

`'telnet'` を通して NNTP サーバーと接続するときに使うコマンドです。これは中間のホストと接続するためのものではありません。これはまさに本当の NNTP サーバーと接続するためのものです。デフォルトは `'telnet'` です。

`nntp-telnet-switches`

`nntp-telnet-command` に渡すスイッチのリストです。デフォルトは `'("-8")'` です。

6.2.1.4 NNTP marks

Gnus は NNTP サーバーのための記事の印 (marks) (see Section 3.7 [Marking Articles], page 56) を印ファイルに保存します。印ファイルはあるグループで設定した印を記録し、それぞれのファイルは、対応するサーバーに対して専用です。印ファイルは、ニュースサーバーに似ている古典的な階層で ‘~/News/marks’ (nntp-marks-directory) に保存されます。例えば news.gmane.org サーバーにおける ‘gmane.discuss’ グループのための印ファイルは ‘~/News/marks/news.gmane.org/gmane/discuss/.marks’ に保存されます。

印ファイルは役に立ちます。‘~/News/marks’ ディレクトリーは (rsync、scp または他の何かを使って) Gnus を走らせる別のホストにコピーすることができ、どの記事を読んで印を付けたかをそちらで再現します。‘~/News/marks’ のデータは、‘~/newsrc.eld’ にある同じものよりも優先されます。

印ファイルは、それぞれのサーバーでそれ専用非常に特化されることに注意して下さい。Gnus は記事番号を記憶するので、両方のホストで同じサーバーを使っていないと、ものごとは壊れてしまうでしょう (大抵の NNTP サーバーは他のどんなサーバーとも同じ記事番号を使いません)。しかし、あるホストでサーバー A、B、C を使い、別のホストでサーバー A、D、E を使う場合には、A のための印ファイルを同じにすることができるので、二つのホスト間でそのサーバーは同期するでしょう。

NNTP 印の使用は性能の劣化を招き、Gnus をのろく感じさせる可能性があります。そういう場合は nntp-marks-is-evil 変数を t に設定してみてください。すると、印は ‘~/newsrc.eld’ (だけ) に格納されるようになるでしょう。

関連する変数:

nntp-marks-is-evil

非-nil だったら、このバックエンドは印ファイルを無視します。デフォルトは nil です。

nntp-marks-directory

NNTP グループの印が格納されるディレクトリーです。

6.2.2 ニューススプール

ローカルスプールから外部グループを購読することは極めて簡単だし便利かもしれませんが。非常に大きな記事があるグループ—例えば ‘alt.binaries.pictures.furniture’ を読む速度が速くなります。

とにかく、nnsPOOL を選択方法として、かつ "" (もしくは何でも) をアドレスとして指定するだけです。

もしローカルスプールにつなぐことが可能なら、おそらくそれを基本選択方法として使うべきでしょう (see Section 1.1 [Finding the News], page 3)。それは普通は nntp 選択方法を使うより速いですが、そうでないかもしれません。それは場合によります。何があなたのサイトで一番良いかを見つけるために、いろいろと試してみなければなりません。

nnsPOOL-inews-program

記事を投稿するために使われるプログラムです。

nnsPOOL-inews-switches

記事を投稿するときに inews プログラムに与えられるパラメーターです。

nnsPOOL-sPOOL-directory

nnsPOOL が記事を探すところです。これは普通は ‘/usr/spool/news/’ です。

nnspool-nov-directory

nnspool が NOV ファイルを探すところです。これは普通は `‘/usr/spool/news/over.view/’` です。

nnspool-lib-dir

ニュースのライブラリーが置かれているディレクトリーの場所です (デフォルトで `‘/usr/lib/news/’` です)。

nnspool-active-file

アクティブファイルの絶対パス名です。

nnspool-newsgroups-file

`‘newsgroups’` ファイルの絶対パス名です。

nnspool-history-file

`‘history’` ファイルの絶対パス名です。

nnspool-active-times-file

`‘active.times’` ファイルの絶対パス名です。

nnspool-nov-is-evil

`nil` でないと、nnspool はそれが見つけたどんな NOV ファイルも使おうとはしません。

nnspool-sift-nov-with-sed

`nil` でないと、これがデフォルトですが、概観ファイル (overview) から関連する部分を得るために `sed` を使います。もし `nil` だと、nnspool はファイル全体をバッファに読み込んで、そこで実行します。

6.3 メール取得

ニュースリーダーでメールを読むなんて実に奇妙ですよ? いや、もちろんできるのですが。

6.3.1 ニュースリーダーでメール

使い慣れた伝統的なメールリーダーから Gnus に乗り換えることを決断したならば、かなりのカルチャーショックを経験することになるでしょう。

Gnus は伝統的なメールリーダーのようなふるまいをしません。あなたが望むならそのようにもできますが、それは骨折り損のくたびれ儲けです。

Gnus はふつう同じ手法ですべてのグループを扱います。あるグループを選んで新しい、または未読のメッセージを読むと、それらには既読の印が付けられ、(意図的に要求しなければ) 以後はそれらを目にすることはありません。これってとてもニュースリーダー的でしょ。

メッセージを消すために、取り立てて何かを行なうことはありません。

このことは既読のメッセージはすべて消されてしまうことを意味するのかって? そりゃあんまりですよ!

しかし、そうではありません。古いメッセージは何らかの仕組みによって期限切れ消去 (*expire*) されるのです。ニュースのメッセージはニュースの管理人 (が管理しているサーバー) によって期限切れ消去の処理が制御され、メールの期限切れ消去の処理はあなたが制御します。メールの期限切れ消去については、Section 6.3.9 [Expiring Mail], page 163 で徹底的に網羅されています。

多くの Gnus の利用者が、それをニュースとメールの両方でしばらく使ってみた後で気が付くのは、その配送の機構がメッセージの扱い方に関して行なうことが、ほんの少ししか無いことです。

多くの人たちが複数のメーリングリストを講読しています。それらは SMTP で配送されるもの、すなわちメールです。それらのメッセージに返答をしないまま、あるいはさらに、それらを非常に注意深くは読まないままに、私たちは何週間も過ごすかもしれません。でも、そういうメッセージを保存しておく必要はありません。なぜならば、もう一度読む必要が生じたとしても、それらはどこかに保存されているからです。

ある人たちは小人数に利用されているローカルニュースグループを講読しています。それらは NNTP で配送されるもの、すなわちニュースです。私たちは自分の仕事に役立てるために、それらの膨大なメッセージの断片を読んだり返事をしなければなりません。しかもそれらがどこかに保存されているとは限らないので、興味のあるメッセージを個人メールと同じように保存しなければなりません。

配送の仕組みの違いはいつでもよいことで、大事なのはいかに主題に興味を持っているかと、もう一度読みたいときにいかに簡単に呼び出せるかなのです。

Gnus はメールをニュースグループのように「グループ」に並べ変えて、各々のグループ（メールがニュース）を別個に扱うための豊富な機能を提供します。

ある人たちは Gnus（えっへん）のやりかたに満足できなくて、Gnus が男（male）になること、もとい、メールリーダーになることを欲します。Gnus をもっとメールリーダー的なものにするために鞭打つことは可能ではあるのですが、前にも言ったように簡単ではありません。いわゆるメールリーダーが好みならば VM を使いましょう。これは優秀な、厳密な意味でのメールリーダーです。

脅かすわけではないのですが、はっきりさせておきたいのは、あなたにメッセージについての新しいやり方を修得して欲しいということです。あなたが Gnus のやり方を受け入れてくれた暁には、きっとあなたは Gnus が好きになるでしょう。請け合いますよ。（少なくとも、私が Gnus に入れた Emacs のサブリミナル脳味噌洗濯関数を売ってくれた人はそれを保証しています。あなたも同化します。あなたは Gnus を愛します。あなたは Gnus でのメールの方法を愛します。絶対に。）

6.3.2 メールを読むことを始める

Gnus を使って新しいメールを読むことはまったく簡単です。あなたが選んだメールバックエンドを `gnus-secondary-select-methods` に放り込むだけで、自動的に読むことができるようになります。

例えば `nnml`（これは「メールにつきーファイル」のバックエンドです）を使いたいなら、次のものを「`~/gnus.el`」ファイルに入れれば良いでしょう：

```
(setq gnus-secondary-select-methods '((nnml "")))
```

そうすると、次に Gnus を起動したときにこのバックエンドは新しい記事を求め、すべてのメッセージをスプールファイルからそのディレクトリー（ディフォルトでは「`~/Mail/`」）に移します。新しいグループ（「`mail.misc`」）が作られて、他のグループと同じように読むことができるようになります。

たぶんメールをいくつかのグループに分割したいでしょうけれど：

```
(setq nnmail-split-methods
  '(("junk" "^From:.*Lars Ingebrigtsen")
    ("crazy" "^Subject:.*die\\`Organization:.*flabby")
    ("other" "")))
```

これは三つの新しい `nnml` メールグループ「`nnml:junk`」、`nnml:crazy` および「`nnml:other`」を作ることになります。初めの二つのグループにふさわしくないすべてのメールは、最後のグループに置かれます。

Gnus でメールを読むにはこれで十分なはずですが。マニュアルのこの部分の他の章を熟読する必要があるかもしれませんが。特に Section 6.3.13 [Choosing a Mail Back End], page 168 と Section 6.3.9 [Expiring Mail], page 163 を。

6.3.3 メール分割

訳注: このマニュアルの多方面で使われている「分割」という語のうち、受信したメールをいろいろなグループに「区分け」することを意味するものは“ split ”という語に対応します。ある一つのメールを「分解」するのではなくて、外からやって来た複数のメールをそれぞれの格納先に一通ずつ「振り分ける」意味で使っています。

変数 `nnmail-split-methods` は入ってくるメールをどのようにグループ分けするかを指定します。

```
(setq nnmail-split-methods
  '(("mail.junk" "^From:.*Lars Ingebrigtsen")
    ("mail.crazy" "^Subject:.*die\\|\\^Organization:.*flabby")
    ("mail.other" "")))
```

この変数はリストのリストで、それぞれのリストの最初の要素はメールグループの名前、二つめの要素はそれぞれのメールがそのグループに属するかどうかをヘッダーで判定するために使う正規表現です (ところで、メールグループの名前が ‘mail’ で始まる必要はありません)。最初の文字列は、`replace-match` が合致した文章から取り出した副表現を挿入するために使われるような、‘\\1’ の様式を含むかもしれません。例えば:

```
("list.\\1" "From:.* \\(.*\\)-list@majordomo.com")
```

この場合、挿入されるテキストを小文字にすべきかどうかを `nnmail-split-lowercase-expanded` が制御します。See Section 6.3.6 [Fancy Mail Splitting], page 157.

二番目の要素は関数でも構いません。その場合、それは規則の最初の要素 (メールグループの名前) を引数として、ヘッダーだけに範囲を狭められたバッファーで呼ばれます。メールがそのグループに属すると判断したら、その関数は `nil` でない値を返す必要があります。

これらのグループの最後は常に総合的なものであるべきで、その正規表現は他の正規表現に合致しなかったメールに合致するために、いつも ‘’ でなければなりません。(これらの規則は連想リストの初めから終わりまで順番に処理されます。クロスポストを有効にしていない限り、最初に合致した規則が「勝ち」ます。クロスポストを有効にしている場合は、すべての合致した規則が「勝ち」ます。) 合致する規則がなかったら、メールは最後に ‘bogus’ グループで終わります。メール分割によって新しいグループが作られた場合は、それらを見るために `gnus-group-find-new-groups` を実行する必要があるでしょう。これは ‘bogus’ グループにも当てはまります。

あなた自身でこれをいじくりまわしたいときは、あなたの選んだ関数をこの変数に設定することができます。この関数は入って来たメールメッセージのヘッダーに範囲を狭められたバッファーで、引数無しで呼ばれます。この関数は、そのメールメッセージをが行くべきだと判断するグループ名のリストを返さなければなりません。

すべてのメールバックエンドは、入って来た気の毒な無実のヘッダーを乱暴に扱っても良いことに注意して下さい。それらはすべて Lines ヘッダーを追加します。いくつかは X-Gnus-Group ヘッダーを加えます。たいいていのは Unix の mbox の From<SPACE> 行を何か別のものに変えます。

すべてのメールバックエンドはクロスポストをサポートします。複数の正規表現が合致すると、メールはそれらすべてのグループに「クロスポスト」されます。`nnmail-crosspost` はこの機能を使うかどうかを指定します。どの記事も総合の (‘’) グループにクロスポストされないことに注意して下さい。

`nnmh` と `nnml` はクロスポストされた記事にハードリンク (hardlink) を作ることによってクロスポストを行ないます。しかし、すべてのファイルシステムがハードリンクの機能を提供しているわけではありません。もしあなたがその場合に当てはまるのであれば、`nnmail-crosspost-link-function` を `copy-file` に設定して下さい。(この変数はデフォルトで `add-name-to-file` です。)

以前に行なわれたメール分割がメッセージをどこに入れたかを見たい場合は、`M-x nnmail-split-history` 命令を使って下さい。これからスプールし直そうとするメッセージがどこに入るかを見たい場合は、`gnus-summary-respool-trace` および関連する命令 (see Section 3.25 [Mail Group Commands], page 103) を使って下さい。

`nnmail-split-header-length-limit` の制限より長いヘッダー行は、分割関数の処理対象から除外されます。

デフォルトでは分割の処理においてヘッダーを MIME デコードするので、非-ASCII 文字列に合致させることができます。変数 `nnmail-mail-splitting-charset` で、デコードするときのデフォルトの文字セットを指定します。`nnmail-mail-splitting-decodes` を `nil` に束縛することによってまったくデコードを行なわないようにすることができます、それは、生のヘッダーのデータを元に記事の合致を判定したい場合には役立つでしょう。

デフォルトでは入ってくるすべてのメッセージに対して分割の処理が行なわれます。しかし、もし `mail-sources` 変数 (see Section 6.3.4.1 [Mail Source Specifiers], page 148) に `directory` の項目を設定すると、デフォルトでは分割は行なわれません。変数 `nnmail-resplit-incoming` を `nil` ではない値に設定すれば、この場合でも分割を起こさせることができます。(この変数は他の種類の項目に対しては効果がありません。)

Gnus はあなた自身に災いが及ぶ可能性あっても、あなたが望むすべての機会を提供します。例えば、あなたの上司からくるすべてのメールを入れるグループを作ったとしましょう。その後、偶発的にそのグループの購読をやめてしまうとどうなるでしょう。それでも Gnus は上司からのすべてのメールを未購読のグループに入れるので、上司が「月曜日までにその報告書を準備しないと首だ!」というメールをあなたに送っても、あなたはそれを見ることはなく、火曜日になって本当は翌月の家賃を払うために空のボトルを集めるべきであっても、まだ有給で雇われていると信じているかもしれません。

6.3.4 メールソース

メールはたくさんの別のソース (source) から取得することができます—例えばメールスプールから、POP メールサーバーから、`procmail` ディレクトリーから、`maildir` から。

6.3.4.1 メールソース指示子

「メールソース指示子」に メールソース (see Section 6.3.4.4 [Fetching Mail], page 155) を設定して、Gnus にメールを取得する方法を指示しましょう。

例です:

```
(pop :server "pop3.mailserver.com" :user "myname")
```

ご覧のように、メールソース指示子はリストで、最初の要素は「メールソースの型」、それに任意の数の「キーワード」が続きます。明示的に指定されていないキーワードはデフォルト値になります。

以下のメールソースの型が使用可能です:

file	単一のファイルからメールを取得します。普通はメールスプールです。 キーワード: :path ファイルの名前です。デフォルトは MAIL 環境変数の値か <code>rmail-spool-directory</code> の値 (普通は <code>'usr-mail/spool/user-name'</code> のようなもの) です。 :prescript :postscript それぞれのメールを取得する前と後で実行するスクリプトです。
-------------	---

ファイルメールソースの例:

```
(file :path "/usr/spool/mail/user-name")
```

もしくは、ディフォルトのファイル名を使うと:

```
(file)
```

メールスプールファイルがローカルマシンに無い場合は、POP や IMAP などメールを取得するのが最善です。ここでは `ange-ftp` のファイル名は使用できません—メールを移動しているときにメールスプールをロックする方法がありません。

適当なサーバーを設置することが不可能なら、代わりに `ssh` を使うことができます。

```
(setq mail-sources
      '((file :prescript "ssh host bin/getmail >%t")))
```

‘getmail’ スクリプトは以下のようなものになるでしょう:

```
#!/bin/sh
# getmail - move mail from spool to stdout
# flu@iki.fi

MOVEMAIL=/usr/lib/emacs/20.3/i386-redhat-linux/movemail
TMP=$HOME/Mail/tmp
rm -f $TMP; $MOVEMAIL $MAIL $TMP >/dev/null && cat $TMP
```

あなたが使いたい ‘movemail’ と一時ファイルに合わせて、スクリプトを書き換えて下さい。

directory

特定のディレクトリーにある複数のファイルからメールを取得します。これは普通は `procmail` に新しいメールをいくつかのファイルに分割させているときに使われます。すなわち、そのディレクトリーにあるファイルとグループは一対一で対応しているので、‘foo.bar.spool’ ファイルにあるメールは `foo.bar` グループに置かれます (接尾語の `.spool` は変更可能です)。nnmail-scan-directory-mail-source-once を `nil` 以外の値にすると、Gnus に新しいメールソースを一回だけ調べさせることができます。これは特に、指定したレベルのメールグループだけを調べたいときに便利です。

`nnmail-resplit-incoming` という変数もあり、これを非-`nil` にすると通常の分割処理がそのディレクトリーにあるすべてのファイルに対して行なわれます (see Section 6.3.3 [Splitting Mail], page 147)。

キーワード:

:path ファイルがあるディレクトリーの名前です。これにはディフォルト値はありません。

:suffix この接尾語で終わる名前ファイルだけが使われます。ディフォルトは ‘.spool’ です。

:predicate

この述語が `nil` でない値を返すファイルだけが使われます。ディフォルトは `identity` です。これは追加の選別器として使用されます—正しい接尾語で、かつ この述語を満足するファイルだけが対象になります。

訳注: この場合の述語は関数で、正しい接尾語を持つファイルの名前が引数として渡されます。

```
:prescript
:postscript
```

それぞれのメールを取得する前と後で実行するスクリプトです。

ディレクトリーメールソースの例です:

```
(directory :path "/home/user-name/procmail-dir/"
           :suffix ".prcml")
```

pop

POP サーバーからメールを取得します。

キーワード:

:server POP サーバーの名前です。デフォルトは MAILHOST 環境変数から取得されます。

:port POP サーバーのポート番号です。これは数値 (例えば ‘:port 1234’) か文字列 (例えば ‘:port "pop3"’) です。もし文字列なら Unix システムにおける ‘/etc/services’ に載っているサービス名でなければなりません。デフォルトは “pop3” です。システムによっては “pop-3” としなければならないかもしれません。

:user POP サーバーに与える利用者名です。デフォルトはログイン名です。

:password POP サーバーに与えるパスワードです。設定しないと利用者は入力を求められます。

:program POP サーバーからメールを取得するために使用されるプログラムです。これは format で使うような文字列でなければなりません。例です:

```
fetchmail %u@s -P %p %t
```

有効な書法仕様指示子は:

‘t’ メールがそこに移動させられるファイルの名前です。これは常にこの文字列に含まれていなければなりません。

‘s’ サーバーの名前です。

‘p’ サーバーのポート番号です。

‘u’ 使用する利用者名です。

‘p’ 使用するパスワードです。

これらの仕様で使われる値は、対応するキーワードに与えた値から取られます。

```
:prescript
```

メールを取得する前に実行されるスクリプトです。構文は :program キーワードと同じです。これは実行される関数であることもできます。

```
:postscript
```

メールを取得した後で実行されるスクリプトです。構文は :program キーワードと同じです。これは実行される関数であることもできます。

```
:function
```

POP サーバーからメールを取得するために使う関数です。その関数は一つのパラメーター (メールがそこへ移動されるべきファイルの名前) とともに呼ばれます

:authentication

これは password かシンボル apop のどちらかで、何の認証方式を使うかを指示します。デフォルトは password です。

:program と :function キーワードが指定されていない場合は pop3-movemail が使われます。pop3-movemail を使う場合に pop3-leave-mail-on-server が非-nil だったら、メールは取得した後も POP サーバーに残されます。POP サーバーはセッションとセッションの間の状態の情報を維持しないので、そこにあるクライアントが信頼できる情報と、実際にそこにあるものは一致しないかもしれないことに注意して下さい。それらが一致しないと、メールをダブって受け取るか、またはすべてが崩壊して、あなたは壊れたメールボックスとともに置き去りにされる可能性があります。

訳注: Gnus に含まれている 'pop3.el' を使う場合に pop3-leave-mail-on-server を非-nil に設定するのは、あまり意味がありません。サーバーに残されたメールは、次回に (何度でも) 再び取り込まれてしまいます。一度取り込んだメールを二度と取得しないようにする機能を持つ 'pop3.el' は T-gnus に含まれています。これは XEmacs 用に開発されたものが元になっています。しかし、残念ながら誰が開発したかがわかりません。したがって FSF への正式な権利譲渡が行われていないので、Gnus に含めることができないのです。

参考: 以下は T-gnus の 'pop3.el' を使う場合に自動的に追加されるメールソース用のキーワードです:

:connection

サーバーに接続するときに使うストリームで、ssl、tls または nil を指定することができます。デフォルトは nil で、安全ではない接続を用います。ssl と tls では外部プログラムとライブラリーが必要であることに注意して下さい:

ssl SSL を使います。OpenSSL ('openssl' プログラム) か SSLeay ('s_client') と外部ライブラリー 'ssl.el' が必要です。

tls STARTTLS (SSL に類似) を使います。外部ライブラリー 'starttls.el' と 'starttls' プログラムが必要です。

:leave 非-nil でメールをサーバーに残し、メッセージの取得に UIDL を使います。デフォルトは nil です。

メールを POP サーバーから取得するための、いくつかの例を挙げます。デフォルトの利用者名を使って、デフォルトの POP サーバーから取得し、デフォルトの取得方法を使用します:

```
(pop)
```

指名したサーバーから、指名した利用者とパスワードで取得します:

```
(pop :server "my.pop.server"
     :user "user-name" :password "secret")
```

メールの移動に 'movemail' を使います:

```
(pop :program "movemail po:%u %t %p")
```

maildir Maildir からメールを取得します。これは少なくとも qmail と postfix によってサポートされているメールボックスの形式で、特別のディレクトリーにあるそれぞれのファイルは、厳密に一通のメールを含んでいます。

キーワード:

:path メールが保存されるディレクトリーの名前です。デフォルトは環境変数 MAILDIR から取得した値か、または ‘~/Maildir/’ です。

:subdirs Maildir のサブディレクトリーです。デフォルトは ‘(“new” “cur”)’ です。

リモートマシンからメールを取り寄せることも出来ます。(というのも、maildir はロックの問題を気にせずに済むからです。)

Maildir メールソースの例をふたつ:

```
(maildir :path "/home/user-name/Maildir/"
         :subdirs ("cur" "new"))

(maildir :path "/user@remotehost.org:~/Maildir/"
         :subdirs ("new"))
```

imap IMAP サーバーからメールを取得します。何らかの理由で、IMAP をそれが意図されたようなネットワーク上でメールを読むプロトコルとしては (すなわち nnimap で) 使いたくないときは、Gnus では POP サーバーと同様に扱って、指定された IMAP メールボックスから記事を取得することができます。詳しくは Section 6.5 [IMAP], page 185 を参照して下さい。

Kerberos, GSSAPI, TLS/SSL および STARTTLS のための外部プログラムとライブラリーが必要であることに留意して下さい。See Section 6.5 [IMAP], page 185.

キーワード:

:server IMAP サーバーの名前。デフォルトは環境変数 MAILHOST から得ます。

:port IMAP サーバーのポート番号。普通はデフォルトは ‘143’ で、TLS/SSL 接続には ‘993’ です。

:user IMAP サーバーに渡す利用者名です。デフォルトはログイン名です。

:password IMAP サーバーに渡すパスワードです。指定されていないときは、利用者は入力することを促されます。

:stream サーバーに接続するときに使うストリーム。imap-stream-alist にあるシンボルの中のひとつを設定します。現状では ‘gssapi’, ‘kerberos4’, ‘starttls’, ‘tls’, ‘ssl’, ‘shell’ またはデフォルトの ‘network’ になります。

:authentication サーバーでの認証にどの認証法を使うか。これには imap-authenticator-alist で定義されているシンボルの一つを設定します。現状では ‘gssapi’, ‘kerberos4’, ‘digest-md5’, ‘cram-md5’, ‘anonymous’ またはデフォルトの ‘login’ になります。

:program :stream に ‘shell’ が設定されているときは、この値が変数 imap-shell-program に割り当てられます。これは format ふうの文字列 (または文字列のリスト) でなければなりません。例を示しましょう:

```
ssh %s imapd
```

有効な書法仕様指示子は以下の通りです。

‘s’ サーバーの名前。
 ‘l’ imap-default-user で設定された利用者名。
 ‘p’ サーバーのポート番号。

これらの指定に使われる値は、対応するキーワードに与えた値から取ってきます。

:mailbox メールを取得するメールボックスの名前。デフォルトは ‘INBOX’ で、これは普通は入ってくるメールを受け取るメールボックスです。

:predicate
 取得する記事を見つけるために使われる述語。デフォルトの、‘UNSEEN UNDELETED’ はおそらくたいいていの人には最良の選択でしょうが、ときどき IMAP クライアントでメールボックスを覗いて、いくつかの記事に既読 (または SEEN) の印を付けるなら、これを ‘1:*’ に設定する必要があるかもしれません。そうすれば、メールボックスのすべての記事は印の如何に関わらず取得されます。述語の完全な一覧は、RFC2060 の 6.4.4 節を読んで下さい。

:fetchflag
 サーバーで、取得した記事にフラグを付ける方法。デフォルトの ‘\Deleted’ はそれらに消去のフラグを付けますが、単に既読のフラグを付けるための ‘\Seen’ が代案になるでしょう。これらは最もありそうな二つの選択ですが、他のフラグも RFC2060 の 2.3.2 節で定義されています。

:dontexpunge
 nil でなかったら、記事を取得した後で、それらに消去の印が付いていても削除しません。

IMAP メールソースの例:

```
(imap :server "mail.mycorp.com"
      :stream kerberos4
      :fetchflag "\\Seen")
```

webmail <http://www.hotmail.com/>, <http://webmail.netscape.com/>,
<http://www.netaddress.com/>, <http://www.yahoo.com/> などのウェブ
 メールサーバーからメールを取得します。

注: ウェブメールはクッキーに大きく依存します。url "4.0pre.46" を使う場合は "one-line-cookie" パッチを当てる必要があります。

警告: メールが失われるかもしれません。無保証です。

キーワード:

:subtype ウェブメールサーバーの型です。デフォルトは hotmail です。他の候補には netscape, netaddress, my-deja があります。

:user ウェブメールサーバーに渡す利用者名です。デフォルトはログイン名です。

`:password`

ウェブメールサーバーに渡すパスワードです。指定しない場合は、利用者は入力することを促されます。

`:dontexpunge`

`nil` でなかったら、未読の記事だけを取得し、かつ取得後にそれらをごみ箱のフォルダーに移動しません。

ウェブメールのソースの例です:

```
(webmail :subtype 'hotmail
         :user "user-name"
         :password "secret")
```

Common Keywords

共通キーワードはどんな型のメールソースにも使うことができます。

キーワード:

`:plugged` `nil` でなかったら、Gnus が `unplugged` (ネットワークから切り離されている状態) であってもメールを取得します。ディレクトリーをメールソースに使っているのならば、この例のように指定することができます:

```
(setq mail-sources
      '((directory :path "/home/pavel/.Spool/"
                  :suffix ""
                  :plugged t)))
```

こうしておくことによって `unplugged` であっても Gnus はメールを取得します。これはローカルのメールとニュースを使う場合に便利です。

6.3.4.2 関数インターフェース

上記のいくつかのキーワードは、実行するための Lisp 関数を指定します。関数が実行されている間だけ、それぞれのキーワード `:foo` に対して Lisp 変数 `foo` が、そのキーワードの値に束縛されます。例えば、以下のメールソースの設定例について考えてみて下さい。

```
(setq mail-sources '((pop :user "jrl"
                          :server "pophost" :function fetchfunc)))
```

関数 `fetchfunc` が実行されているとき、`user` というシンボルの値は `"jrl"` に束縛され、`server` というシンボルの値は `"pophost"` に束縛されます。port, password, program, prescript, postscript, function および authentication の値もまた (それらのデフォルト値に) 束縛されます。

それぞれの型のメールソースのためのキーワードのリストについては、前述の説明を参照して下さい。

6.3.4.3 メールソースのカスタマイズ

以下はメールの取得方法に影響する変数の一覧です。普通はこれらのどれも設定または変更する必要は無いでしょう。

`mail-source-crash-box`

メールが、それを処理している期間中に格納されている場所です。デフォルトは `'~/ .emacs-mail-crash-box'` です。

mail-source-delete-incoming

nil でなければ、入って来たメールを一時的に格納したファイルを、それを処理した後で消去します。t ではファイルをただちに消去し、nil ではいかなるファイルも消しません。正の数だった場合は、その日数以上に古いファイルを消去します (消去は新着メールを受け取るときだけ行なわれます)。mail-source-delete-incoming を nil にしておいて、mail-source-delete-old-incoming をフックで呼ぶか、または対話的に呼ぶこともできます。

mail-source-delete-old-incoming-confirm

非-nil だったら、古い incoming (メールの到着時に使われた) ファイルを消去するときの確認を求めます。この変数は mail-source-delete-incoming が正の数である場合だけ使われます。

mail-source-ignore-errors

非-nil だったら、メールソースからメールを読むときのエラーを無視します。

mail-source-directory

入ってきたメールソースのファイルが (もしあれば) 格納されるディレクトリーです。ディフォルトは ‘~/Mail/’ です。現時点でこれが使われる唯一のものは、変数 mail-source-delete-incoming が nil または数値であった場合に、入ってきたメールを格納するファイルの置き場所の指定です。

mail-source-incoming-file-prefix

入ってきたメールを一時的に格納するファイルの名前の接頭語です。ディフォルトは ‘Incoming’ で、この場合ファイルは ‘Incoming30630D_’ や ‘Incoming298602ZD’ のようになります。これが本当に関係するのは mail-source-delete-incoming が nil または数値の場合だけです。

mail-source-default-file-modes

すべての新しいメールファイルはこのファイルモードになります。ディフォルトは 384 です。

mail-source-movemail-program

nil でなかったら、新着メールを取り込むためのプログラムの名前であると解釈されます。nil だったら exec-directory にある movemail が使われます。

6.3.4.4 メール取得

新しいメールをどこから取得するかを実際に Gnus に指示する手段は、mail-sources をメールソース指示子のリストに設定することです (see Section 6.3.4.1 [Mail Source Specifiers], page 148)。

この変数 (と旧式の nnmail-spool-file) が nil であれば、メールバックエンドは決して自分自身ではメールを取得しようとしません。

ローカルのスプールと POP メールサーバーの両方からメールを取得したいなら、このようにすることができます:

```
(setq mail-sources
      '((file)
        (pop :server "pop3.mail.server"
              :password "secret")))
```

あるいは、これらのキーワードのどんなディフォルトも使いたくなければ、このようにして下さい:

```
(setq mail-sources
  '((file :path "/var/spool/mail/user-name")
    (pop :server "pop3.mail.server"
        :user "user-name"
        :port "pop3"
        :password "secret"))))
```

あなたがメールバックエンドを使うとき、Gnus はすべてのメールを inbox から吸い上げてホームディレクトリーに放り込みます。あなたがメールバックエンドを使っていないときは、Gnus は一通もメールを移動しません—そういう場合には、最初にたくさんの魔法を唱えなければなりません。まず五芳星を描き、蝋燭を灯し、山羊を生け贄として捧げ終えた後には、Gnus があなたのメールを移動したとしても、あなたは実際にはあまり驚かないはずです。

6.3.5 メールバックエンド変数

これらの変数 (のほとんど) は、すべてのさまざまなメールバックエンドに関連します。

nnmail-read-incoming-hook

すべてのメールバックエンドは、新しいメールを読み込んだ後にこのフックを呼びます。そうしたければ、何かのメール監視プログラムに知らせるためにこのフックを使うことができます。

nnmail-split-hook

それぞれのメッセージがそのヘッダーに基づいて分割がなされる直前に、それが格納されているバッファで実行されるフックです。このフックは、それがふさわしいと考えられるようにするために、どんなやり方でも自由にバッファの内容を編集することができます—バッファは分割が終わった後で捨てられ、バッファで行なわれた変更はどのファイルにも現れません。gnus-article-decode-rfc1522 は、このフックに加えられることがありそうな関数の一つです。

nnmail-pre-get-new-mail-hook

nnmail-post-get-new-mail-hook

これらは、入ってくるメールを処理するときに実行される、有用な二つのフックです—nnmail-pre-get-new-mail-hook は新しいメールを処理する直前に呼ばれ、nnmail-post-get-new-mail-hook はメールの扱う処理が終わったときに呼ばれます。以下はこれらの二つのフックを使って、新しいメールのファイルのファイルモードを変更する例です:

```
(add-hook 'nnmail-pre-get-new-mail-hook
  (lambda () (set-default-file-modes 511)))

(add-hook 'nnmail-post-get-new-mail-hook
  (lambda () (set-default-file-modes 551)))
```

nnmail-use-long-file-names

nil でないなら、メールバックエンドは長いファイル名とディレクトリー名を使います。'mail.misc' のようなグループは 'mail.misc' という長い名前のディレクトリーかファイルに収められます (nnml バックエンドの場合はディレクトリー、nnfolder バックエンドの場合はファイルです)。nil だったら、同じグループは 'mail/misc' に収められます。

nnmail-delete-file-function

ファイルを消去するために呼ばれる関数です。デフォルトは delete-file です。

nnmail-cache-accepted-message-ids

nil でないと、バックエンドに (例えば Gcc によって) 入って来た記事の Message-ID を、メールの重複を発見するためのキャッシュに入れます。デフォルトは nil です。

nnmail-cache-ignore-groups

正規表現が正規表現のリストです。名前がどれかの正規表現に合致するグループの記事は、Message-ID キャッシュに記録されません。

例えば特級分割 (see Section 6.3.6 [Fancy Mail Splitting], page 157) を関数 `nnmail-split-fancy-with-parent` とともに使っている場合に役立つでしょう。

6.3.6 特級メール分割

訳注: *Fancy* という単語は、創造的、空想的、気まぐれな、好きな、派手な、上等な、極上の、変わり種の、等々のさまざまな意味で使われますが、ここでは助動詞無しで使える利便を考えて「特級」という訳語を割り当てました。

比較的単純な、標準のメール分割指定の方法では思い通りにならないならば、`nnmail-split-methods` を `nnmail-split-fancy` に設定すると良いでしょう。そうすると、変数 `nnmail-split-fancy` で遊ぶことができるようになります。

まずこの変数の値の例を見てみましょう:

```
;; メイラーデーモンから送られたメッセージが、普通のグループにクロス
;; ポストされないようにします。警告は本当のエラーとは違ったグループ
;; に入れます。
(| ("from" mail (| ("subject" "warn.*" "mail.warning")
                  "mail.misc"))
;; エラーでないメッセージはすべての関連するグループにクロスポスト
;; しますが、(ding) メーリングリストと他の (ding) 関連のメールの
;; ためのグループにはクロスポストしません。
(& (| (any "ding@ifi\\.uio\\.no" "ding.list")
      ("subject" "ding" "ding.misc"))
;; 他のメーリングリスト...
(any "procmail@informatik\\.rwth-aachen\\.de" "procmail.list")
(any "SmartList@informatik\\.rwth-aachen\\.de" "SmartList.list")
;; 以下のどちらのメーリングリストも同じ接尾語なので、bugs- だ
;; けに投稿されたものが mypkg.list にクロスポストされないよう
;; にしています。しかし本当にクロスポストされた記事をクロスボ
;; ストすることはできるようになっています。
(any "bugs-mypackage@somewhere" "mypkg.bugs")
(any "mypackage@somewhere" - "bugs-mypackage" "mypkg.list")
;; 人々...
(any "larsi@ifi\\.uio\\.no" "people.Lars_Magne_Ingebrigtsen"))
;; 合致しなかったメールはすべてを捕まえるグループへ行きます。
"misc.misc")
```

この変数は「分割」の様式になっています。分割は (ことによると) それぞれの分割が他の分割を含む再帰的構造です。以下は使うことができる分割の構文です:

group 分割が文字列だったら、それはグループ名であるとみなされます。通常の正規表現の展開が行なわれます。後述の例 (訳注: ‘\\&’, ‘\\1’~‘\\9’) を見て下さい。

(*field value* [- *restrict* [...]] *split* [*invert-partial*])

この分割は少なくとも三つの要素を含んでいる必要があります。最初の要素 *field* (正規表現) に合致するヘッダーが *value* (これも正規表現) に合致する文字列を含んでいたならば、*split* で指定されたグループにメッセージを格納します。

field の後にあって、しかも合致した *value* の最後尾より前にある何かの文字列に *restrict* (これもまた正規表現) が合致したら、*split* は無視されます。いくつかの *restrict* のどれもが合致しなければ *split* が実行されます。

最後の要素 *invert-partial* は任意です。これが省略されていなくて、しかも値が *nil* でなければ、語 (word) の境界をまたいで正規表現の合致を行なうかどうかの振る舞い (*nnmail-split-fancy-match-partial-words* 変数によって制御されます; 下記参照) が反転します。(Gnus 5.10.7 の新機能)

(| *split* ...)

分割がリストで、最初の要素が | (垂直棒) だったら、それぞれの *split* をそのうちの一つが合致するまで実行します。ここで言う「合致」とは、ある *split* がメッセージを一つ以上のグループに格納しようとすることです。

(& *split* ...)

分割がリストで、最初の要素が & だったら、そのリストにあるすべての *split* を実行します。

junk もし分割がシンボル *junk* だったら、そのメッセージを保存しません (すなわち、消去してしまいます)。非常に注意して使って下さい。

(: *function arg1 arg2* ...)

もし分割がリストで、最初の要素が : だったら、二番目の要素が *args* を引数として関数として呼ばれます。関数は *split* を返さなければなりません。

例えば以下の関数は、記事のボディーに基づいた分割に使えるでしょう:

```
(defun split-on-body ()
  (save-excursion
    (save-restriction
      (widen)
      (goto-char (point-min))
      (when (re-search-forward "Some.*string" nil t)
        "string.group"))))
```

function が実行されるとき、バッファはメッセージの部分に狭められます。それが上記の例で *save-excursion* と *save-restriction* の後で (*widen*) を呼ぶ必要がある理由です。さらに *nnimap* バックエンドの場合、デフォルトでは記事のボディーがダウンロードされないことに注意して下さい。それをするためには *nnimap-split-download-body* を *t* に設定する必要があります (see Section 6.5.1 [Splitting in IMAP], page 190)。

(! *func split*)

分割がリストで最初の要素が ! だったら、*split* が実行され、*func* は *split* の結果を引数として呼ばれます。*func* は分割を返さなければなりません。

nil 分割が *nil* だったら、それは無視されます。

これらの分割で *field* は完全なフィールド名 (というかヘッダー名) に合致しなければなりません。

通常これらの分割における *value* は、基礎モード (fundamental mode) 構文テーブル (syntax table) に従って、完全に 語 (word) に合致しなければなりません。言い換えれば、すべての *value* は暗に `<...>` 印 (語の区切り記号) で囲まれます。したがって、例えば以下の分割を使うと、

```
(any "joe" "joemail")
```

`'joedavis@foo.org'` からやって来たメッセージは、通常 `'joemail'` には格納されないでしょう。この振る舞いを変更したければ、以下の三つのやり方のどれでも使うことができます:

1. `nnmail-split-fancy-match-partial-words` 変数を `nil` ではない値に設定することによって、語の境界を無視させることができます。すると、合致はより `grep` ふうになります。この変数は、特級分割で語の境界をまたいだ合致を行なうかどうかを制御します。デフォルト値は `nil` です。

分割の規則のすべての *value* に影響することに注意して下さい。

2. `.*` で始まる *value* は、語の前にある語の境界を無視させます。同様に `.*` で終わる *value* は、語の後ろにある語の境界を無視させます。例えば `"@example\\.com"` という *value* は `'foo@example.com'` に合致しませんが、`".*@example\\.com"` ならば合致します。
3. この章の最初の方で述べた `invert-partial` フラグを、`'(field value ...)'` 型の分割規則で使うことができます。このフラグが設定されていると、`nnmail-split-fancy-match-partial-words` が `nil` であっても、語の両側にある語の境界は無視されます。逆に、このフラグが設定されていると、`nnmail-split-fancy-match-partial-words` が `nil` ではない値であっても、語の境界は無視されません。(Gnus 5.10.7 の新機能)

field と *value* は Lisp シンボルであることもできます。その場合それらは `nnmail-split-abbrev-alist` で指定された内容に従って展開されます。これはセルの `CAR` がキーを含んでいて、`CDR` が関連付けられた値を持っているコンセル (cons cell) の連想リストです。以下の項目が、あらかじめ `nnmail-split-abbrev-alist` に定義されています:

`from` `'From'`、`'Sender'` および `'Resent-From'` の各フィールドに合致します。

`to` `'To'`、`'Cc'`、`'Apparently-To'`、`'Resent-To'` および `'Resent-Cc'` の各フィールドに合致します。

`any` `from` と `to` を統合したものです。

`nnmail-split-fancy-syntax-table` は、これらのすべての分割が実行されているときに有効な構文テーブルです。

ヘッダーのいくつかの情報に基づいて、Gnus に動的にグループを作らせたい (例えば、グループ名の一部を `replace-match` のようなやり方で置き換えさせたい) ならば、次のようなことができます。

```
(any "debian-\\b\\(\\w+\\)@lists.debian.org" "mail.debian.\\1")
```

この例では、`'debian-foo@lists.debian.org'` に送られたメールは `'mail.debian.foo'` というグループに入れられます。

文字列が要素 `'\\&'` を含んでいる場合は、直前に合致した文字列で置き換えられます。同様に、要素 `'\\1'` から `'\\9'` までは、合致した文字列の一部で置き換えられます (訳注: 正規表現の中に `'\\('` と `'\\)'` を使ってグループにまとめられたものが一つ以上ある場合に、`'\\n'` はその正規表現の *n* 個目のグループに合致する文字列の一部で置き換えられます)。

その際、合致した文字列を小文字にしたもので置き換えるべきかどうかを `nnmail-split-lowercase-expanded` が決定します。これを非-`nil` にすることによって、アドレスで大文字と小文字が区別せずに使われている (例えば `mailing-list@domain` と `Mailing-List@Domain`) 場合でも、複数のグループが生成されてしまうことを避けることができます。デフォルトは `t` です。

関数 `nnmail-split-fancy-with-parent` は、フォローアップ記事を親記事と同じグループに振り分けるために使います。メールの振り分けを一生懸命設定してみても完璧にはできないことがありますね。例えば、上司から個人宛てのメールが届いたとします。自分が携っているプロジェクトとは別の話です。けれど「他のメールと区別できるようにこれこれこういう言葉を表題に書いて下さい」と上司に向かって指図するわけにはいきませんから、結局自分の手を煩わしてひとつひとつメールを正しいグループに振り分けるはめになります。そんなときにこの関数を使うと、この面倒な作業を一スレッドにつき一回きりで済ますことができます。

この機能を利用するためには、まず変数 `nnmail-treat-duplicates` および `nnmail-cache-accepted-message-ids` の値を `nil` ではない値に設定する必要があります。それができたら `nnmail-split-fancy-with-parent` を使ってみて下さい。コロンを使ってこんな風に書きます:

```
(setq nnmail-treat-duplicates 'warn      ; または delete
      nnmail-cache-accepted-message-ids t
      nnmail-split-fancy
      '(| (: nnmail-split-fancy-with-parent)
          ;; 残りの振り分け方はここに書く
      ))
```

この機能は実際、次の様に働いています: 変数 `nnmail-treat-duplicates` の値が非-`nil` の場合、Gnus は見つけた全記事のメッセージ ID を変数 `nnmail-message-id-cache-file` で指定されたファイルに記録します。このとき、それぞれの記事が格納されたグループの名前を併記します (ただしメールではないメッセージの場合は、グループ名は省略されます)。さて、いよいよメールの振り分けが始まると、関数 `nnmail-split-fancy-with-parent` は、分割される各記事の References (と In-Reply-To) ヘッダーを調べ、`nnmail-message-id-cache-file` で指定されたファイルにそれらのメッセージ ID があるかどうか調べます。親記事が見つかったら、そのグループ名が正規表現 `nnmail-split-fancy-with-parent-ignore-groups` に合致しなければ、この関数は対応するグループ名を返すわけです。ここで、変数 `nnmail-message-id-cache-length` の値をデフォルトよりも幾らか大きな値に設定することを勧めます。そうすると、今調べられたメッセージ ID たちは今しばらくキャッシュの中に存続できます (5000 に設定するとキャッシュファイルの大きさはだいたい 300 キロバイトぐらいになるみたいです)。さらに、変数 `nnmail-cache-accepted-message-ids` の値を非-`nil` に設定すれば、Gnus は移動された記事のメッセージ ID をも記録するので、フォローアップ記事は親記事の移動先と同じグループに入るようになります。

特定のグループをキャッシュに記録したくない場合は、変数 `nnmail-cache-ignore-groups` も参照して下さい。例えば、外に出すすべてのメッセージを“outgoing”グループに保存しているのならば、`nnmail-cache-ignore-groups` をそのグループ名に合致するように設定すれば良いでしょう。さもないとあなたのすべてのメッセージに対する返事が“outgoing”グループに入ってしまいます。

6.3.7 グループメール分割

何ダースものメーリングリストを購読しているけれど、手でメール分割規則を維持したくないというときのために、グループメール分割というものがあります。あなたがしなければならないことは、グループパラメーターかグループカスタマイズで `to-list`, `to-address` の両方もしくはどちらかを設定して `nnmail-split-methods` を `gnus-group-split` に設定するだけです。分割関数はすべてのグループでこれらのパラメーターを走査し、それに従って分割します。すなわち、メールグループのパラメーター `to-list` か `to-address` で指定されたアドレスから投稿されたものか、そのアドレスへ投稿されたメッセージがそのグループに保存されます。

ときには、メーリングリストには複数のアドレスがあり、メール分割にそれらすべてを認識させる必要があるかもしれません: `extra-aliases` グループパラメーターを追加のアドレスのリストに設定するだけで終了です。あえて正規表現を使いたければ、`split-regexp` を設定して下さい。

これらのすべてのグループのパラメーターは、nnmail-split-fancy の分割を作成するために使用されます。その分割の仕様の中身は、*field* の値が 'any' であり、*value* の値が to-list と to-address と extra-aliases のすべてと split-regexp に合致するもののすべてに合致する単一の正規表現、そして *split* がグループの名前になります。restrict も使うことができます: それには split-exclude パラメーターを正規表現のリストに設定して下さい。

これらのすべてのパラメーターを使って正しい分割が生成されないときや、何かもっと凝ったものが必要なときは、split-spec パラメーターを nnmail-split-fancy の分割に設定することができます。この場合は、前もって書かれたすべてのパラメーターは gnus-group-split に無視されます。特に、split-spec は nil (訳注: これも nnmail-split-fancy の分割の一種です) に設定することができ、その場合そのグループは gnus-group-split に無視されます。

それぞれのグループのために、一つの分割を含む単一の & 特級分割を定義することによって、gnus-group-split は合致するすべてのグループにクロスポストをします。どの分割にも合致しないメッセージは、どれかのグループで split-spec が catch-all に設定されていない場合は gnus-group-split-default-catch-all-group で指定された名前のグループに格納されます。その場合、そのグループはすべてを受け取る (catch-all) グループとして使われます。この変数はしばしばグループを指定するためだけに使われますが、任意の複雑な特級分割に設定することもできるので (結局のところグループ名は特級分割なのです)、個人のメールフォルダーにそれらのメールが格納されるどのメーリングリストにも当てはまらないメールを、分割するのに便利でしょう。なおこの特級分割は、1 分割リスト (それは、グループパラメーターから抽出された規則を持った & 分割をも含んでいます) の最後の要素として追加されることに注意して下さい。

そろそろ例を出すべきでしょう。以下のグループパラメーターが定義されていることを仮定します:

```
nnml:mail.bar:
((to-address . "bar@femail.com")
 (split-regexp . ".*@femail\\.com"))
nnml:mail.foo:
((to-list . "foo@nowhere.gov")
 (extra-aliases "foo@localhost" "foo-redirect@home")
 (split-exclude "bugs-foo" "rambling-foo")
 (admin-address . "foo-request@nowhere.gov"))
nnml:mail.others:
((split-spec . catch-all))
```

nnmail-split-methods を gnus-group-split に設定すると、nnmail-split-fancy が選択されていて、かつ変数 nnmail-split-fancy が以下のように設定されているかのように振舞います:

```
(| (& (any "\\(bar@femail\\.com\\|.*@femail\\.com\\)" "mail.bar")
      (any "\\(foo@nowhere\\.gov\\|foo@localhost\\|foo-redirect@home\\)"
            - "bugs-foo" - "rambling-foo" "mail.foo"))
  "mail.others")
```

グループ分割をすべてのメールグループで積極的には使いたくなければ、nnmail-split-fancy の分割を次のように使用することで、いくつかのグループだけで使うことができます。

```
(: gnus-group-split-fancy groups no-crosspost catch-all)
```

groups は、出力の分割を生成するためにパラメーターが走査されるグループ名のリストか、それらのグループ名に合致する正規表現です。no-crosspost はクロスポストを禁止するために使うことができ、その場合は、単一の | 分割が出力されます。catch-all は gnus-group-split-default-catch-all-group のように、最後の手段として使われる特級分割です。catch-all が nil に設定さ

れているか、`split-regexp` がどれかの選択されたグループで空の文字列に合致すると、すべてを受け取る (`catch-all`) 分割は発行されません。そうでない場合、あるグループに `catch-all` に設定されている `split-spec` があると、そのグループは `catch-all` 引数の値よりも優先されます。

不運なことに、すべてのグループとそれらのパラメーターを走査することは、特にすべてのメッセージに対して行なわなければならないことを考慮に入れると、非常に遅くなるでしょう。でも、絶望してはいけません。`gnus-group-split-setup` 関数を、はるかに効率的な方法で `gnus-group-split` を動作させるために使うことができます。それは `nnmail-split-methods` を `nnmail-split-fancy` に設定し、`nnmail-split-fancy` を `gnus-group-split-fancy` で生成される分割に設定します。そうすることによって、どんなに分割するメッセージがたくさんあっても、グループパラメーターは一度だけ走査されるようになります。

しかしながら、グループパラメーターを変更すると、`nnmail-split-fancy` を手で更新しなければならなくなるでしょう。`gnus-group-split-update` を実行することによって、それを行なうことができます。どちらかと言えば、それを自動的に更新したい場合には、`gnus-group-split-setup` にそれを実行するように指示して下さい。例えば、`~/gnus.el` に以下のものを追加すれば良いでしょう:

```
(gnus-group-split-setup auto-update catch-all)
```

`auto-update` が `nil` でなければ `gnus-group-split-update` が `nnmail-pre-get-new-mail-hook` に追加されるので、二度と `nnmail-split-fancy` の更新について心配する必要はありません。`catch-all` を省略しない場合は (それはオプションで `nil` と等価です)、`gnus-group-split-default-catch-all-group` がその値に設定されます。

`gnus-group-split-update` によって設定された `nnmail-split-fancy` を後で変更する必要があるための、この関数 (`gnus-group-split-update`) は終了する直前に `gnus-group-split-update-hook` を実行します。

6.3.8 古いメールを取り込む

たいていの人は色々なファイルフォーマットで保存されたたくさんの古いメールを持っているでしょう。Gnus の粋なメールバックエンドの一つを使うように設定したのであれば、おそらく古いメールをメールグループに取り込みたいと思いますよね。

それをすることはとても簡単です。

例を挙げましょう: `nnml` (see Section 6.3.13.3 [Mail Spool], page 169) を使ってメールを読んでいて、`nnmail-split-methods` を申し分の無い値に設定しているものとしましょう。重要な、しかし古いメールで、古い Unix `mbox` ファイルが満たされています。あなたはそれを `nnml` グループに移動したいと思っています。

方法です:

1. グループバッファーに行って下さい。
2. `G f` をタイプして下さい。 `nndoc` グループを作成するための元になる `mbox` ファイルの名前を求められるので、それを入力して下さい (see Section 2.9 [Foreign Groups], page 21)。
3. `SPACE` をタイプして、新しく作られたグループに入って下さい。
4. `M P b` をタイプして、グループバッファーのすべての記事に実行印を付けて下さい (see Section 3.7.6 [Setting Process Marks], page 60)。
5. `B r` をタイプしてプロセス印の付いたすべての記事を再スプールして下さい。その際に入力を求められるので、`'nnml'` と答えて下さい (see Section 3.25 [Mail Group Commands], page 103)。

今や `mbox` ファイルのすべてのメールメッセージは、あなたの `nnml` グループ群にもばらまかれています。それらに入って、ものごとが変な故障も無く、うまくいったかどうかを調べて下さい。大

丈夫なようであれば、mbox ファイルを消そうと思うかもしれませんが、私はすべてのメールがあるべきところに納まったことを完全に確認するまでは、そうはしません。

再スプールすることは、あるメールバックエンドを別のものに変更するときにも便利なものです。古いメールグループにあるメールは、新しいメールバックエンドを使ってただ再スプールすれば良いのです。

6.3.9 メールの期限切れ消去

伝統的なメールリーダーは、既読の印を付けるとメールの記事を何らかの方法で削除する傾向があります。Gnus はメールを読むことに対して、基本的に違う方法を取ります。

基本的に Gnus は、メールを少々変わった方法で受け取られたニュースであるとみなします。実際にメールを変更したり、メールメッセージを消す権限があるとは考えません。あなたがメールグループに入って記事に「既読」の印を付けたり、何らかの他のやり方で切ったりしても、メールの記事はまだシステムに存在しています。繰り返します: Gnus はあなたの古い既読のメールを消去しません。もちろん、あなたがそうしろと要求しない限りの話ですが。

要らないメールを Gnus に削除させるには、記事に「期限切れ消去可能」(expirable) の印を付けなければなりません。(デフォルトのキー割り当てでは、E をタイプしなければならないということです。) しかしながら、これは記事が即座に消え去るということではありません。一般的にメール記事は、1) 期限切れ消去可能の印が付いていて、かつ 2) 一週間以上経っている、という場合に、システムによって削除されます。記事を期限切れ消去可能にしなければ、それは地獄が凍りつくまでシステムに残り続けます。このことは、もう一度強調付きで繰り返されるに足るものです: 「もし」あなたが記事を「期限切れ消去可能」に「しない」なら、Gnus は「決して」それらの「記事」を消去しません。

手作業で記事に期限切れ消去可能の印を付けなければならないわけではありません。Gnus は “auto-expire” および “total-expire” と呼ばれる二つの機能を提供して、あなたの手助けをします。かいつまんで言えば “auto-expire” はあなたが記事を選択したときに Gnus が E を叩いてくれることを意味します。そして “total-expire” は、すべての既読の記事は期限切れ消去可能であると Gnus が解釈することを意味します。したがって ‘E’ の印が付けられた記事に加えて、‘r’、‘R’、‘O’、‘K’、‘Y’ などの印が付けられた記事も期限切れ消去可能であると解釈されます。

では auto-expire または total-expire は、いつ使用されるべきなのでしょう? メーリングリストを購読しているほとんどの人々は、それぞれのリストがそれ用のグループに分割されるようにして、それらのグループに対して auto-expire または total-expire を有効にしています。(それぞれのリストをそれ用のグループへの分割する件についてのさらなる情報は Section 6.3.3 [Splitting Mail], page 147 を参照して下さい。)

auto-expire と total-expire のどちらが良いのでしょうか? それに答えるのは簡単ではありません。概して言えば、たぶん auto-expire が速いでしょう。auto-expire の別の利点は、より多くの印を後で読み返すつもりの記事に使うことができる、つまり今までどおりに可視 (tick)、保留 (dormant) または既読 (read) の中から選ぶことができるということです。しかし total-expire では、dormant と ticked からしか選べません。total-expire の利点は、適応スコア付け (see Section 7.6 [Adaptive Scoring], page 237) で良好に働くことです。auto-expire は通常のスコア付けでは動作しますが、適応スコア付けではだめです。

正規表現 `gnus-auto-expirable-newsgroups` に合致するグループでは、読んだすべての記事に自動的に期限切れ消去可能の印が付けられます。期限切れ消去可能の印の付いたすべての記事は、概略バッファの最初の桁に ‘E’ が表示されます。

自動期限切れ消去を有効にすると、デフォルトではあなたが読んだすべての記事に、以前に読まれたかどうかに関わらず、Gnus は期限切れ消去可能の印を付けます。既読の印が付いている記事

に、自動的に期限切れ消去可能の印が付けられるのを避けるには、以下のようなものを ‘~/gnus.el’ ファイルに置いておけば良いでしょう:

```
(remove-hook 'gnus-mark-article-hook
             'gnus-summary-mark-read-and-unread-as-read)
(add-hook 'gnus-mark-article-hook 'gnus-summary-mark-unread-as-read)
```

グループを自動期限切れ消去可能にしても、すべての既読の記事が期限切れ消去されるわけではなく、期限切れ消去可能の印が付いている記事だけが期限切れ消去されることに気を付けて下さい。また、`d` 命令が自動的に記事を期限切れ消去可能にするのでは無いことにも気を付けて下さい——自動期限切れ消去可能にしたグループでは、記事に既読の印が半自動で付けられることによるのみ、記事が期限切れ消去可能になるということです。

2~3 のメーリングリストを講読していて、読み終わってしばらく経ったら記事が消えてしまうようにしたいなら、例えばこんな風に設定しましょう:

```
(setq gnus-auto-expirable-newsgroups
      "mail.nonsense-list\\|mail.nice-list")
```

自動期限切れ消去を行なわせるもう一つの方法は、そのグループのグループパラメーターに `auto-expire` という要素を持たせることです。

もし適応スコア付け (see Section 7.6 [Adaptive Scoring], page 237) と自動期限切れ消去を使用していると、問題が起こるでしょう。自動期限切れ消去と適応スコア付けはあまり良く調和しません。

変数 `nnmail-expiry-wait` で、期限切れ消去可能な記事をどれくらいの期間残しておくかのデフォルトの時間を設定します。Gnus はメッセージが送り出されたときではなく、それが 到着 してからの日数を計算します。デフォルトは 7 日間です。

Gnus は記事がどのグループに属しているかに基づいて、それをどのくらい残しておくかをこまめに設定する関数も提供しています。以下の例では ‘mail.private’ グループは一ヶ月、‘mail.junk’ グループは一日、その他全部は六日間に、それぞれ期限を設定します:

```
(setq nnmail-expiry-wait-function
      (lambda (group)
        (cond ((string= group "mail.private")
                31)
              ((string= group "mail.junk")
                1)
              ((string= group "important")
                'never)
              (t
                6))))
```

この関数に与えられるグループ名には「装飾」すなわち ‘nnml:’ のようなものは付きません。

変数 `nnmail-expiry-wait` と関数 `nnmail-expiry-wait-function` は、数値 (整数である必要はありません) かシンボルの `immediate` か `never` のどちらかにすることができます。

期限切れ期間を選択的に変更するために、グループパラメーターの `expiry-wait` を使うこともできます (see Section 2.10 [Group Parameters], page 23)。

記事を期限切れ消去するときに取られる通常の動作は、それらを消去することです。しかし、場合によってはそれらを消去するよりも別のグループに移動した方が有意義かもしれません。変数 `nnmail-expiry-target` (とグループパラメーター `expiry-target`) はこれを制御します。この変数の値はすべてのグループに対するデフォルトになりますが、特定のグループごとにグループパラメーター

を使って指定すれば、そちらを優先させることができます。デフォルトの値は `delete` ですが、文字列 (記事を移動する先のグループ名) または移動先のグループ名か `delete` を返す関数にすることができます (関数の場合は、記事に範囲を狭めたバッファで、その記事が存在しているグループ名が引数として与えられます)。

グループ名を指定する場合の例:

```
(setq nnmail-expiry-target "nnml:expired")
```

Gnus には期限切れのメールをグループに移動させるための関数があります。それは変数 `nnmail-fancy-expiry-targets` に従って動作します。例です:

```
(setq nnmail-expiry-target 'nnmail-fancy-expiry-target
  nnmail-fancy-expiry-targets
  '(to-from "boss" "nnfolder:Work"
    ("subject" "IMPORTANT" "nnfolder:IMPORTANT.%Y.%b")
    ("from" ".*" "nnfolder:Archive-%Y")))
```

この設定を行なうことにより、表題ヘッダーに `IMPORTANT` を持っていて、`YYYY` 年 `MMM` 月に発信されたいかなるメールも、期限になると `nnfolder:IMPORTANT.YYYY.MMM` グループに移動させられます。また、`From` または `To` ヘッダーが文字列 `boss` を含んでいるメールは `nnfolder:Work` に、それ以外のすべてのメールは `nnfolder:Archive-YYYY` に、それぞれ期限になると移動させられます。

`nnmail-keep-last-article` が `nil` でないと、Gnus はメールグループの最後の記事を決して期限切れ消去しません。これは `procmail` の利用者の人生をより楽にするためのものです。

補足: 上記の、Gnus が決して期限切れ消去可能でない記事を期限切れ消去することはない、というのは嘘です。 `total-expire` をグループパラメーターに入れても、記事に期限切れ消去の印が付くことはありませんが、読んだすべての記事は期限切れ消去の処理に通されます。非常に注意して使ってください。さらに危険なのは変数 `gnus-total-expirable-newsgroups` です。この正規表現に合致するすべてのグループでは、読んだすべての記事が期限切れ消去の処理に通されます。これは、当のグループのすべての古いメールの記事は、しばらく後で削除されるということです。非常に注意して使ってください。そして、あなたが使った正規表現が間違ったグループに合致してしまい、すべての重要なメールが消えてしまったと言って、私に泣き付いて来ないでください。しっかりなさい! (直訳: 男になりなさい、あるいは女になりなさい、さもなければもっと気持ちいい何にでもなりなさい!) ほうら、言わんこっちゃない!

たいていの人はほとんどのメールグループを `total-expirable` (全体期限切れ消去可能) にしますが。

`gnus-inhibit-user-auto-expire` が `nil` でなければ、グループで自動期限切れ消去が有効になっても、利用者が印を付ける命令が記事に期限切れ消去可能の印を付けることはありません。

6.3.10 メール洗濯

メイラーやメーリングリストのサーバーは、メールに対して本当に本当に馬鹿げたことをすることで悪名高いです。「わあ、RFC822 はサーバーを通過していくメッセージのすべての行の最後に `wE aRe ElItE!!!!1!!` を加えることを明示的に禁止はしていないぞ。さあ、やってみよう!!!!1!!」ええ、そのとおりですが、RFC822 はおろか者が読むようには書かれていません。当たり前のこと (訳注: 良識から逸脱すること) はそこでは議論されていません。ですから、この章が必要なのです。

適例: ドイツ語版の Microsoft Exchange は返答の表題に `'Re: '` の代わりに `'AW: '` を付け加えます。私はこれに動揺して狼狽しているふりをすることもできましたが、そうする気力がありませんでした。それは笑うべきことです。

Gnus は表示する記事を洗濯するために多すぎるほどの関数を提供していますが、メールをディスクに保存する前にふるいにかけることができた方が良いでしょう。その目的のために、三つのフックとそれらのフックに入れることができる色々な関数を用意しています。

`nnmail-prepare-incoming-hook`

このフックはメールに何かをする前に呼ばれ、総括的に掃除してきれいにする所作のためにあります。それは新しいすべての入ってきたメールを含んでいるバッファーで呼ばれます。使うことのできる関数は:

`nnheader-ms-strip-cr`

それぞれの行から、最後にあるキャリッジリターン (carriage return) を取り除きます。これは MS のマシン上で動作している Emacs のデフォルトです。

`nnmail-prepare-incoming-header-hook`

このフックはそれぞれのメールのヘッダーに範囲を狭められて呼ばれます。ヘッダーをきれいにするときを使うことができます。使うことのできる関数は:

`nnmail-remove-leading-whitespace`

「役に立つ」メーリングリストのサーバーが、見栄えを良くするためだと称して、ヘッダーの前の方に付け加えた空白を無くします (訳注: 例えば ‘Subject:’ などの直後に二つ以上の空白文字があったら、一つを残して消します)。まったくもう。

(この関数はすべてのメッセージのボディーの中にあるヘッダー (ボディーの中にある別のメッセージが持っているヘッダー行のようなもの) に対しても動作するので、使用に際しては潜在的な危険を孕んでいます。したがってバグを修正するよりは、そういう特徴があることを文書で説明するのが、もちろん正しい解決の道です。)

`nnmail-remove-list-identifiers`

いくつかのメーリングリストのサーバーは、そのリストが配信したメールであることを同定するための識別子—例えば ‘(idm)’—をすべての Subject ヘッダーの先頭に付け加えます。石器時代のメールリーダーを使っている人たちには、それは確かに良いことです。この関数は正規表現 `nnmail-list-identifiers` に合致する文字列を取り除きます。それは正規表現のリストでも構いません。ただし正規表現に `\\(\\.\\.\\)` を含めてはいけません。

例えば ‘(idm)’ と ‘nagnagnag’ という識別子を取り除きたいのなら:

```
(setq nnmail-list-identifiers
      '("(idm)" "nagnagnag"))
```

これは `gnus-list-identifiers` で非破壊的に行なうこともできます。See Section 3.17.3 [Article Hiding], page 83.

`nnmail-remove-tabs`

すべての ‘TAB’ 文字を ‘SPACE’ 文字に変換します。

`nnmail-ignore-broken-references`

いくつかの MUA (例えば Eudora と Pegasus) は壊れた References ヘッダーを作成しますが、In-Reply-To ヘッダーにはちゃんとしたものを入れます。この関数は、ヘッダー部に正規表現 `nnmail-broken-references-mailers` に合致する行があったら、References ヘッダーを取り除きます。

nnmail-prepare-incoming-message-hook

このフックはそれぞれのメッセージに範囲を狭められて呼ばれます (訳注: 一度に複数のメールを受信した場合でも、一通ずつ呼ばれるということです)。使うことのできる関数は:

article-de-quoted-unreadable

Quoted Readable エンコードをデコードします (訳注: 実際に行なうのは quoted printable のデコードです)。

6.3.11 重複

いくつかのメーリングリストのメンバーなら、時々同じメールを二つ受け取ることがあるでしょう。これはとても煩わしいので、nnmail はそれが見つけたどんな重複をも、調べて処理します。これをするために、nnmail は古い Message-ID を nnmail-message-id-cache-file (デフォルトでは `~/nnmail-cache`) に保存します。それに保存される Message-ID のおおよその最大数は変数 nnmail-message-id-cache-length で制御され、デフォルトは 1000 です。(ですから千個の Message-ID が溜められます。) これで怖気をふるったなら、nnmail-treat-duplicates を warn (デフォルトではそのようになっていますが) に設定しても良いでしょう。そうすると、nnmail は重複したメールを消去しない代わりに、それが別のメッセージの重複であるという警告をメールのヘッダーに挿入します。

この変数は関数であることもできます。その場合、関数は当のメッセージに範囲を狭められたバッファから Message-ID を引数として呼ばれます。この関数は nil, warn, delete のどれかを返さなければなりません。

変数を nil に設定することによって、この機能を完全に使わないようにすることができます。

もしすべての重複したメールを特別な *duplicates* グループに入れたいのであれば、普通のメール分割方法を使ってそれを行うことができます:

```
(setq nnmail-split-fancy
  '(| ;; 重複したメッセージは分かれたグループへ。
    ("gnus-warning" "duplicat\\(e\\|ion\\) of message" "duplicate")
    ;; デモンやポストマスターなどからのメッセージは他へ。
    (any mail "mail.misc")
    ;; 他の規則。
    [ ... ] ))
```

もしくは次のようなもの:

```
(setq nnmail-split-methods
  '(("duplicates" "^Gnus-Warning:.*duplicate")
    ;; 他の規則。
    [...]))
```

すてきな使い方があるよ: 受け手である彼女がメールを Gnus で読んでいることと、彼女が nnmail-treat-duplicates を delete に設定してあることを知っていれば、彼女がすでに受け取ったことがわかっているメールの Message-ID そのものを使って、考えられる限りたくさんの侮辱を送ることができるんだぜ。その面白さを考えてみてよ! 彼女はそれらを決して見ることはないんだ! わお!

6.3.12 メールを読むのではない

あなたが使い始めたどんなメールバックエンドでも、あなたがそれらでメールを読みたいと思っていると仮定するという、悩ましい癖を持っていることに気が付くでしょう。これは決して不合理ではないかもしれませんが、あなたの望むことではないかもしれません。

`mail-sources` と `nnmail-spool-file` を `nil` に設定すれば、どのバックエンドも入ってくるメールを読もうとしなくなって、それは助けになるはずです。

でも、それは行き過ぎでしょう。あなたが、例えば `nnml` でメールを読むことと、しまいこんである古い `Rmail` ファイルを `nnbabyl` を使ってざっと覗くことだけで、まったく満足しているのなら。すべてのバックエンドには `バックエンド-get-new-mail` という変数があります。もし `nnbabyl` がメールを読み込みをやめさせたいのであれば、そのグループの仮想サーバー編集して、`nnbabyl-get-new-mail` を `nil` に設定しましょう。

すべてのメールバックエンドは、入ってくるメールを読み込むときに、保存されるべき記事に範囲を狭めて `nn*-prepare-save-mail-hook` を呼びます。

6.3.13 メールバックエンドを選ぶ

メールグループを動作するようにすると Gnus はメールスプールを読み込みます。メールのファイルはまずあなたのホームディレクトリーに複写されます。その後で何が起るかは、メールをどの様式で格納したいかによります。

標準の Gnus では六つの違ったメールバックエンドがあり、さらに多くのバックエンドを個別に手に入れることができます。ほとんどの人が使うメールバックエンドは (それがたぶん最速なので) `nnml` です (see Section 6.3.13.3 [Mail Spool], page 169)。

6.3.13.1 Unix メールボックス

`nnmbox` バックエンドはメールを格納するために標準の `Un*x mbox` ファイルを用います。`nnmbox` はそれぞれのメール記事にそれがどのグループに属しているかを示す追加のヘッダーを加えます。

仮想サーバーの設定:

`nnmbox-mbox-file`

利用者のホームディレクトリーのメールボックスの名前。デフォルトは `'~/mbox'` です。

`nnmbox-activate-file`

メールボックスのアクティブファイルの名前。デフォルトは `'~/mbox-active'` です。

`nnmbox-get-new-mail`

`nil` でなければ、`nnmbox` は入って来たメールを読み込んでグループに分割します。デフォルトは `t` です。

6.3.13.2 Rmail Babyl

`nnbabyl` バックエンドはメールを格納するために `Babyl` メールボックス (別名 `Rmail mbox`) を使います。`nnbabyl` はそれぞれの記事にそれがどのグループに属しているかを示す追加のヘッダーを加えます。

仮想サーバーの設定:

`nnbabyl-mbox-file`

`Rmail mbox` ファイルの名前。デフォルトは `'~/RMAIL'` です。

`nnbabyl-active-file`

`Rmail mbox` のためのアクティブファイルの名前。デフォルトは `'~/rmail-active'` です。

`nnbabyl-get-new-mail`

`nil` でなければ、`nnbabyl` は入ってくるメールを読み込みます。デフォルトは `t` です。

6.3.13.3 メールスプール

`nnml` スプールメール様式は他の知られている様式とは互換性がありません。それは少し注意して使われるべきです。

このバックエンドを使うと、`Gnus` は入ってくるメールを、それぞれのメールを 1 ファイルとしてファイルに分割し、記事を変数 `nnml-directory` で指定されたディレクトリーの下に対応するディレクトリーに入れます。デフォルトの値は `'~/Mail/'` です。

前もってディレクトリーを作っておく必要はありません。その面倒は `Gnus` がすべて見てくれます。

あなたのアカウントに保存できるファイルの数に厳密な制限があるなら、このバックエンドを使うべきではありません。それぞれのメールはそれ自身のファイルを伴うので、数週間で数千の `i` ノードを占有する可能性は十分にあります。あなたにとってこれが問題でなく、親切なシステム管理者が気が狂ったように「誰が僕の `i` ノードを食いつぶしているんだ？ 誰だ？ 誰!?!」と叫びながら歩き回ることも問題でないなら、これがおそらく使うことのできる一番速い様式であるということは知っておくべきでしょう。新しいメールを読むためだけに大きな `mbox` ファイルを重い足取りで探す必要はありません。

`nnml` は記事分割に関してはおそらく一番遅いバックエンドでしょう。多くのファイルを作らなければならず、入ってくるメールのための `NOV` データベースも作成しなければなりません。これのために、メールを読むことに関してはたぶん最速のバックエンドになるのです。

印ファイル (訳注: marks file) が使われると (それがデフォルトですが)、`nnml` サーバーは `tar` などを使ってバックアップしたり、後であなたが付けた印がすべて保たれた状態で `Gnus` に戻す (本来の `nnml` サーバーによって追加する) ことができる特質を持つようになります。グループの印はそれぞれの `nnml` グループのディレクトリー内の、通常 `'marks'` ファイル (`nnml-marks-file-name` を参照) に格納されます。また、個々の `nnml` グループについてもバックアップすることが可能で、そうするには (バックアップを `nnml` ディレクトリーに戻した後で) `G m` キーを使ってそのグループを元に戻して下さい。

何らかの理由によって `'marks'` ファイルがおかしくなっていると思ったときは、単にそれら全部を消してしまえば良いでしょう。`Gnus` は次回起動するときに、それらを正しく再作成してくれます。

仮想サーバーの設定:

`nnml-directory`

すべての `nnml` ディレクトリーはこのディレクトリーの下に置かれます。デフォルトは `message-directory` の値 (そのデフォルト値は `'~/Mail'`) です。

`nnml-active-file`

`nnml` サーバーのためのアクティブファイル。デフォルトは `'~/Mail/active'` です。

`nnml-newsgroups-file`

`nnml` グループ記述ファイル。See Section 10.8.9.2 [Newsgroups File Format], page 354. デフォルトは `'~/Mail/newsgroups'` です。

`nnml-get-new-mail`

`nil` でなければ、`nnml` は入って来たメール読み込みます。デフォルトは `t` です。

`nnml-nov-is-evil`

`nil` でなければ、このバックエンドはどの `NOV` ファイルも無視します。デフォルトは `nil` です。

`nnml-nov-file-name`

NOV ファイルの名前。デフォルトは `‘.overview’` です。

`nnml-prepare-save-mail-hook`

保存する前に一つの記事に範囲を狭めて実行するフックです。

`nnml-marks-is-evil`

非-`nil` であると、このバックエンドはいかなる 印 ファイルも無視します。デフォルトは `nil` です。

`nnml-marks-file-name`

「印」ファイルの名前です。デフォルトは `‘.marks’` です。

`nnml-use-compressed-files`

非-`nil` だったら、`nnml` は圧縮されたメッセージファイルを扱うことができるようになります。ただし `auto-compression-mode` が有効になっていなければなりません (see section “Compressed Files” in *The Emacs Editor*)。 `nnml-use-compressed-files` の値が文字列だった場合、それは圧縮プログラムを指定するファイル拡張子として使われます。Emacs がそれをサポートしていれば、それを `‘.bz2’` に設定することができます。値 `t` は `‘.gz’` と等価です。

`nnml-compressed-files-size-threshold`

メッセージファイルを圧縮するかどうかを判断するための、サイズの閾値です。 `nnml-use-compressed-files` が非-`nil` に設定されていて、本文の文字数がこの変数の値より大きかったら、メッセージファイルは圧縮されます。

`nnml` グループと NOV ファイルの調子が完全に狂ってしまったら、`M-x nnml-generate-nov-databases` とタイプすることによって、完全に更新することができます。この命令は、それぞれすべてのファイルを見ることによって `nnml` 階層全体をトロール魚網でさらうので、それが終わるまでには時間がかかるかもしれません。この機能へのより良いインターフェースはサーバーバッファで見つかるでしょう (see Section 6.1.2 [Server Commands], page 134)。

訳注: 単一の `nnml` グループの NOV データベースを再生成させるための `nnml-generate-nov-databases-1` という命令もあります。

6.3.13.4 MH スプール

`nnmh` は、NOV データベースを作らないこととアクティブファイルや印ファイルを保持しないことを除いて、`nnml` と似ています。このことは `nnmh` を `nnml` より かなり 遅いバックエンドにしていますが、`procmail` のスクリプトを書くことはずっとやりやすくなってもいます。

仮想サーバーの設定:

`nnmh-directory`

すべての `nnmh` ディレクトリーはこのディレクトリーの下に置かれます。デフォルトは `message-directory` の値 (そのデフォルトは `‘~/Mail’`) です。

`nnmh-get-new-mail`

`nil` でなければ、`nnmh` は入ってくるメールを読み込みます。デフォルトは `t` です。

`nnmh-be-safe`

`nil` でなければ、`nnmh` はフォルダーにある記事が実際に Gnus が考えているものと同じであるかを調べるという馬鹿げたことをやります。それは日付と目に入るすべての情報を調べるので、これを `t` に設定すると深刻な速度低下が起こります。`nnmh` の記事を

読むのに Gnus 以外のものを使っていないのであれば、この変数を `t` に設定する必要はありません。デフォルトは `nil` です。

6.3.13.5 Maildir

`nnmaildir` は各々の Gnus のグループに対応する `maildir` に、`maildir` フォーマットでメールを格納します。このフォーマットは <http://cr.yp.to/proto/maildir.html> および <http://www.qmail.org/man/man5/maildir.html> で文書化されています。また `nnmaildir` は `maildir` の中の `‘.nnmaildir/’` ディレクトリーに追加の情報を格納します。

`Maildir` フォーマットは、配送と講読を、ロックを必要とせずに同時に行なうことができるようにするために設計されました。他のバックエンドでは、メールを何らかのスプールに渡した後で、そのスプールからグループに分割するために、Gnus を設定しなければならないでしょう。それは今まで通り `nnmaildir` で行なうことができますが、もっと普通のやり方は、Gnus のグループとして現われる `maildir` に配送されたメールを、直接手にすることです。

`nnmaildir` は完全に信頼できることを目指しています: `C-g` はメモリー中のデータを壊さないし、`SIGKILL` がファイルの中のデータを壊すことはありません。

`nnmaildir` は記事の印と `NOV` データを、それぞれの `maildir` に格納します。それによって、ある Gnus の環境から別の場所に `maildir` 全体をコピーすることができ、印は保持されます。

仮想サーバーの設定:

`directory`

それぞれの `nnmaildir` サーバー (一つを越えるサーバーが必要だとはとても思えません) に対してディレクトリーを作り、それを `maildir` または `maildir` へのシンボリックリンクとして実装する必要があります (`maildir` のためだけにです。他の目的のためにすでに使われているディレクトリーを選んではいけません)。それぞれの `maildir` は、そのサーバーのニュースグループとして Gnus に現れ、シンボリックリンクのファイル名がそのグループの名前になります。ディレクトリーにある `‘.’` で始まるどんなファイル名も無視されます。ディレクトリーは最初に Gnus を起動したときとグループバッファで `g` をタイプしたときはいつでも走査され、どれかの `maildir` が削除または追加されていると、`nnmaildir` はそのときにそれを知ります。

`directory` パラメーターの値は Lisp 式でなければなりません。それはこのサーバーのためのディレクトリーのパスを得るために `eval` と `expand-file-name` で処理されます。その式はサーバーが開かれたときだけ `eval` され、その結果得られた文字列が、サーバーが閉じられるまで使われます (もし、式や `eval` を知らなくても心配ご無用; 単なる文字列で動作します)。このパラメーターは任意ではなく、必ず設定しなければなりません。 `"~/Mail"` やそのサブディレクトリーを使うことは推奨しません。なぜかと言うと、Gnus の他の複数の部分がそれをデフォルトでいろんなものに使うので、`nnmaildir` でもそれを使うと混乱するかもしれないからです。 `"~/nnmaildir"` が一般的な値です。

`target-prefix`

これは Lisp 式でなければなりません。それは `eval` と `expand-file-name` で処理されます。その式が `eval` されるのはサーバーが開かれたときだけで、その結果得られた文字列がサーバーが閉じられるまで使われます。

`nnmaildir` サーバーにグループを作ると、その名前の頭に `target-prefix` が付加された `maildir` と、その `maildir` を指し示すシンボリックリンクが素のグループ名の名前で作成されます。したがって、`directory` が `"~/nnmaildir"` で、`target-prefix` が `"../maildirs/"` だった場合に `foo` というグループを作ると、`nnmaildir`

は maildir として ‘~/nnmaildir/./maildirs/foo’ を、‘./maildirs/foo’ へのシンボリックリンクとして ‘~/nnmaildir/foo’ を作成します。

同じ directory に maildirs とシンボリックリンクの両方を作成するために、スラッシュを含まない文字列を target-prefix に設定することができます。この場合は、directory で見つかる名前が target-prefix で始まるどの maildir も、グループとは見なされません (が、それらを指し示すシンボリックリンクがグループになります)。

特別な場合として target-prefix が "" (それがデフォルトです) だったら、グループを作るときに、対応するシンボリックリンクを持たない maildir が directory において作成されます。そのようなグループに対しては、force 引数を与えないと gnus-group-delete-group が使えないことに気をつけて下さい。

directory-files

これは directory-files と同じインターフェースを持っている関数 (または directory-files そのもの) でなければなりません。これは maildir 用のサーバーの directory を走査するために使われます。このパラメーターは任意です。デフォルトは、nnheader-directory-files-is-safe が nil だったら nnheader-directory-files-safe で、それ以外の場合は directory-files です (nnheader-directory-files-is-safe はサーバーが開いたときに一回だけ検査されますが、ディレクトリーが走査されるときに毎回チェックさせたいのならば、それを行なう関数をあなたが自前で用意する必要があります)。

get-new-mail

非-nil にしておく、いつもの通りにグループの maildir 自体において新着メールを走査した後で、このサーバーはさらに mail-sources から、nnmail-split-methods か nnmail-split-fancy の設定に従って、従来の Gnus の方法でメールを取り込みます。デフォルト値は nil です。

mail-sources と nnmaildir グループの両方で同じ maildir を使っては いけません。その結果は運良く有益になるかもしれませんが、そんな意図では設計されていませんし、将来は違う結果をもたらす可能性があります。あなたの分割規則が新しいグループを作るようになっている場合は、create-directory サーバーパラメーターを設定することを忘れないで下さい。

6.3.13.6 グループパラメーター

nnmaildir は複数のグループパラメーターを使います。これらのすべてを無視しても安全です。デフォルトの nnmaildir の動作は、他のメールバックエンドのデフォルト (記事が一週間後に消去される、など) と同じです。期限切れ消去のパラメーターを除いて、この機能はすべて nnmaildir だけにあるものです。したがって、別のバックエンドですで行なっている動作を単に踏襲させようというのであれば、これを無視することができます。

これらのパラメーターのうちのどれでも、その値がベクトルである場合は、オリジナルの値に代わって、第一の要素が Lisp 式として評価された結果が使われます。値がベクトルでない場合は、その値そのものが Lisp 式として評価されます。(それが、これらのパラメーターが他とは違う名前、すなわち他のバックエンドでサポートされているものとは違うけれども似た意味を持っている同様のパラメーターを使っている理由です。)(数値、文字列、nil、および t についても eval の関与を無視することができます。他の値について、そうすることがふさわしい場合には、追加のクォートを使い、かつベクトルで値を包むことを忘れないで下さい。)

expire-age

記事が消去されるまでの寿命の秒数を指定する整数、あるいは記事が期限切れ消去されなければならないことを指定する `never` というシンボルです。このパラメーターが設定されていないと、いつもの `nnmail-expiry-wait` 変数または `nnmail-expiry-wait-function` 変数を最後のよりどころにします (`expiry-wait` グループパラメーターが設定されていると、その値が `nnmail-expiry-wait` より優先して使われ、`nnmail-expiry-wait-function` は無効にされます)。3 日の値が必要ならば、`[(* 3 24 60 60)]` のようなものを使って下さい。nnmaildir は式を評価して、その結果を使います。記事の寿命は記事ファイルの変更時刻を基点に計測されます。通常これは記事が配送された時刻と同じですが、記事の編集はそれを若くします。(期限切れ消去以外の) 記事の移動もまた、記事を若くしてしまうでしょう。

expire-group

これが以下のような完全な Gnus のグループ名の文字列で、

```
"backend+server.address.string:group.name"
```

かつこのパラメーターが設定されているグループの名前と同じではなかったら、期限切れ消去が行なわれる際に、記事は消去される代わりに、これで指定されたグループに移動させられます。これが nnmaildir グループに設定されていると、移動先のグループにおいて、記事は元のグループにあったときとちょうど同じ古さになります。したがって、移動先のグループにおける `expire-age` には注意して下さい。これがパラメーターが設定されているのと同じグループの名前に設定されると、記事はまったく期限切れ消去されません。ベクトルの式を使うと、最初の要素が一回、それぞれの記事について評価されます。したがって記事をどこに置くかを決めるために、その式は `nnmaildir-article-file-name` などに照会することができます。たとえこのパラメーターが設定されていなくても、nnmaildir は `expiry-target` グループパラメーターや `nnmail-expiry-target` 変数を顧みません。

read-only

これが `t` に設定されていると、nnmaildir はその記事をこの maildir では読み出し専用として扱います。この意味は、記事は `'new/'` から `'cur/'` に改名されない、記事は `'cur/'` ではなく `'new/'` でのみ見つかる、記事は消去されない、記事は編集できない、ということです。`'new/'` は他の maildir の `'new/'` ディレクトリーへのシンボリックリンクであると想定されます (そのディレクトリーには、例えばみんなが興味があるメーリングリストを含んでいる、システムで共通のメールボックスがあります)。`'new/'` 以外の maildir にあるすべてのものは、読み出し専用として扱われません。したがって、みんなで共有するメールボックスに対しては、あなた自身の maildir を設置する (または共有のメールボックスに書き込み権限を持つ) 必要が依然としてあります。そうすれば、あなたの maildir は記事の余分なコピーをまったく含まなくて済むでしょう。

directory-files

`directory-files` と同じインターフェースの関数です。記事を見つけるために、このグループに対応する maildir のディレクトリーを走査するために使われます。デフォルトはそのサーバーの `directory-files` パラメーターで設定されている関数です。

distrust-Lines:

非-nil にしておくと、nnmaildir は `Lines:` ヘッダーフィールドを使う代わりにいつも記事の行数を数えます。nil だった場合は、あればそのヘッダーフィールドが使われます。

always-marks

[`'(read expire)`]] のような印シンボルのリストです。Gnus が記事の印を `nnmaildir` に尋ねるときはいつでも、ファイルシステムに格納されている印が何であるかとは無関係に、`nnmaildir` はすべての記事がこれらの印を持っていると答えます。これは機能を検証するためのもので、おそらく結局は削除されるでしょう。それは Gnus 本体で行なわれるか、あるいは有益でなければ放棄されるべきです。

never-marks

[`'(tick expire)`]] のような印シンボルのリストです。Gnus が記事の印を `nnmaildir` に尋ねるときはいつでも、ファイルシステムに格納されている印が何であるかとは無関係に、`nnmaildir` はこれらの印を持っている記事は無いと答えます。`never-marks` は `always-marks` よりも優先されます。これは機能を検証するためのもので、おそらく結局は削除されるでしょう。それは Gnus 本体で行なわれるか、あるいは有益でなければ放棄されるべきです。

nov-cache-size

NOV メモリーキャッシュのサイズを指定する整数です。スピードアップのために、`nnmaildir` はそれぞれのグループの限定された数の記事に対して、メモリー上に NOV データを保持します。(これはたぶん有用ではなく、将来はおそらく削除されるでしょう)。このパラメーターの値は、サーバーが開かれた後で最初にグループが見られたとき、すなわち一般には最初に Gnus を起動したときだけ注目されます。サーバーが閉じられて再び開かれるまでは、NOV キャッシュのサイズは変更されません。デフォルトは概略バッファに表示される記事の数の見積り (`tick` 印がある記事の数か `read` が無い記事の数に、少々余分を加えたもの) です。

6.3.13.7 記事の識別

記事はそれぞれの `maildir` の `'cur/` ディレクトリーに格納されます。各々の記事には `uniq:info` のような名前が付けられます。ここで `uniq` はコロンを含みません。`nnmaildir` は `:info` の部分を保持しますが無視します。(他の `maildir` リーダーは一般に印を格納するためにこの部分を使います。) `uniq` の部分は記事をユニークに識別し、`maildir` の `'.nnmaildir/` サブディレクトリーの色々な場所に、対応する記事の情報を格納するために使われます。記事の完全なパス名は、概略バッファで記事を要求した後で `nnmaildir-article-file-name` 変数から得られます。

6.3.13.8 NOV データ

`uniq` によって識別される記事は、その NOV データ (概略バッファの行を生成するために使われる) を `'.nnmaildir/nov/uniq` に格納します。`nnmaildir-generate-nov-databases` 関数はありません。(その必要はあまりありません。記事の NOV データは記事が `nnmail-extra-headers` が変化したときに自動的に更新されます。) 対応する NOV ファイルを消すことによって、単一の記事だけの NOV データの生成を `nnmaildir` に強制することはできます。しかし ご用心。これは `nnmaildir` にこの記事に新しい記事番号を割り振らせるので、`seen` 印、エージェント、およびキャッシュにとって面倒なことになります。

6.3.13.9 記事の印

`'.nnmaildir/marks/flag/uniq` ファイルがある場合に、`uniq` によって識別される記事は、`flag` 印を持つものと考えられます。Gnus が `nnmaildir` にグループの印を尋ねると、`nnmaildir` はそのようなファイルを探して、見つけた印のセットを報告します。Gnus が `nnmaildir` に印のセットを格納することを要求すると、`nnmaildir` は必要に応じて対応するファイルを生成し、または消

去します。(実際は、それぞれの印のために新しいファイルを作るのではなく、i ノードを節約するために単に ‘.nnmaildir/markfile’ へのハードリンクを張ります。)

‘.nnmaildir/marks/’ に新しいディレクトリーを作ることによって、新しい印を創造することができます。印を保持しつつ maildir を tar でまとめてサーバーからそれを削除し、後で tar をほどくと、印は保持されています。印ファイルを作成または消去することによって、あなた自身が印を追加または削除することができます。Gnus が動作していて nnmaildir サーバーが開いているときにこれを行なう場合は、最初にすべての nnmaildir グループの概略バッファから退出してグループバッファで *s* をタイプし、その後グループバッファで *g* か *M-g* をタイプするのが最良です。そうしないと Gnus は変更を捉えてくれずに、それらを元に戻してしまうかもしれません。

6.3.13.10 メールフォルダー

nnfolder はそれぞれのメールグループを別々ファイルに格納するバックエンドです。それぞれのファイルは標準の Un*x mbox 様式です。nnfolder は記事番号と到着時刻を見失わないようにするための追加のヘッダーを加えます。

印ファイル (訳注: marks file) が使われると (それがデフォルトですが)、nnfolder サーバーは tar などを使ってバックアップしたり、後であなたが付けた印がすべて保たれた状態で Gnus に戻す (本来の nnfolder サーバーによって追加する) ことができる特質を持つようになります。グループの印は nnfolder ディレクトリー内の、mbox ファイルに通常 ‘.mrk’ (nnfolder-marks-file-name を参照) が付加された名前のファイルに格納されます。また、個々の nnfolder グループについてもバックアップすることが可能で、(バックアップを nnfolder ディレクトリーに戻した後で) *G m* キーを使えば、そのグループは元に戻ります。

仮想サーバーの設定:

nnfolder-directory

すべての nnfolder メールボックスはこのディレクトリーの下に置かれます。デフォルトは message-directory の値 (そのデフォルトは ‘~/Mail’) です。

nnfolder-active-file

アクティブファイルの名前。デフォルトは ‘~/Mail/active’ です。

nnfolder-newgroups-file

グループ記述ファイルの名前。See Section 10.8.9.2 [Newsgroups File Format], page 354. デフォルトは ‘~/Mail/newsgroups’ です。

nnfolder-get-new-mail

nil でなければ、nnfolder は入ってくるメールを読み込みます。デフォルトは t です。

nnfolder-save-buffer-hook

フォルダーを保存する前に実行されるフックです。nnfolder バッファに対してさえも、Emacs は通常とおりファイル名を変更してバックアップを行なうことに注意して下さい。この機能を無効にしたいのであれば、‘./.gnus.el’ ファイルで次のようなことをすれば良いでしょう:

```
(defun turn-off-backup ()
  (set (make-local-variable 'backup-inhibited) t))

(add-hook 'nnfolder-save-buffer-hook 'turn-off-backup)
```

nnfolder-delete-mail-hook

これから消去されるメッセージに範囲を狭められて実行されるフックです。この関数は別の場所にメッセージをコピーしたり、消去する前に何らかの情報を取り出すために使うことができます。

nnfolder-nov-is-evil

もし非-nil なら、このバックエンドはどんな NOV ファイルをも無視します。デフォルトは nil です。

nnfolder-nov-file-suffix

NOV ファイルの拡張子です。デフォルトは '.nov' です。

nnfolder-nov-directory

NOV ファイルが格納されるディレクトリーです。nil だったら nnfolder-directory が使われます。

nnfolder-marks-is-evil

非-nil であると、このバックエンドはいかなる 印 ファイルをも無視します。デフォルトは nil です。

nnfolder-marks-file-suffix

印 ファイルの拡張子です。デフォルトは '.mrk' です。

nnfolder-marks-directory

印 ファイルが格納されるディレクトリーです。nil だったら nnfolder-directory が使われます。

nnfolder で読みたいたくさんの nnfolder に似たファイルを持っているのなら、そのようなすべてのファイルが nnfolder-directory にあることを nnfolder に気付かせるために、*M-x nnfolder-generate-active-file* 命令を使って下さい。もっとも、これは長いファイル名を使っているときだけ動作しますが。

6.3.13.11 メールバックエンドの比較

まず用語としての「バックエンド」(back end) は、それによってなにものかが取得される、低次のアクセス手段、あるいはそう言いたければ輸送手段です。それが意図するのはどこからかメールを取ってくることなので、Gnus がすぐに手が届く距離の範囲内でメールを受け取るための、適当なバックエンドを選択する必要があります。

同じ概念が Usenet 自身にも存在します。近ごろでは記事へのアクセスは一般的に NNTP で行なわれますが、淒涼たる暗黒の昔には、世界中の誰もが、記事を置いてあるマシン（今日では NNTP サーバーと呼ぶもの）でリーダーを動作させることによって Usenet に接続したものでした。また、アクセスは記事のディレクトリーのスプールの領域に直接に踏み込むリーダーによって行なわれました。たまたまそういうサーバーにいるのなら（あるいは NFS を介して、とにかくそのスプールのディレクトリーを見ることができるのなら）、今でも nntp か nnsPOOL バックエンドのどちらかを選ぶことができます。

(訳注:「淒涼たる暗黒の昔には」はポーの詩「大鴉」の冒頭部分“Once upon a midnight dreary ”。)

メールバックエンドを選択することの行き着く先は、元の形式を処理し、かつ将来便利に使える形式でメールを残すことを、同時に実現するのに適した方法を選び出すことです。それぞれいくつかの良い点と悪い点があります:

nnmbox 歴史的に UNIX システムは、とても一般的で行き届いた定義のたった一つの形式を持っています。すべてのメッセージは単一の「スプールファイル」に到着し、それらは正規表現 ‘^From_’ に合致する行で区切られています。(‘_’ という記号はスペースを意味し、この例ではこれが RFC で規定されている ‘From:’ ヘッダーではないことをはっきりさせるために使っています。) Emacs それに Gnus も歴史的に Unix 環境から始まっているので、元の mailbox 形式をあまりいじくり回さずに済めば、それが最も単純です。したがってこのバックエンドを選んだ場合に、本当のスプールからメールを取得して Gnus にとって都合がよいディレクトリーにメールを移動するために Gnus が主に行なうのは、処理の過程で何も (目立つような) 変更をせずに、単にそれを複製することです。それは Gnus が処理を行なうことができる環境にメールを移動するための「最も気が利かない」方法です。これは移動させることを速くしますが、Gnus がどこに何があるかを調べるときは、解析が遅くなります。

nnbabyl むかしむかしあるところに DEC-10 と DEC-20 がありました。それらは TOPS というオペレーティングシステムや似たようなものを実行していて、メールを読むための普通の (もしかしたら唯一の?) 環境は Babyl というものでした。そのシステムに届いたメールでどんな形式が使われていたかはわかりませんが、Babyl にはメールを変換するための、それ用の内部形式がありました。その変換とは、Babyl 特有のヘッダーと状態ビットを、ファイルにあるそれぞれのメッセージの先頭に挿入するための仕組みによって、スプールファイル風の実体を作ることでした。Rmail は Emacs の最初のメールリーダーで、Richard Stallman によって書かれました。Stallman はその TOPS/Babyl の環境の出身だったので、すでに存在していたメールファイルの一族を理解するように Rmail を書きました。Gnus は (この件に関しては VM も) この形式をサポートし続けています。それは、そのメーラー特有のヘッダー/ステータス・ビットというものが、かなり良質だと認められているからです。Rmail 自身もちろんまだ存在していて、今でも Stallman によって維持されています。

上記の両方の形式は、メールをファイルシステムにおける単一のファイルに置いたままにするので、メールを見るたびにファイル全体を解析しなければなりません。

nnml nnml は、あたかも nnsPOOL でアクセスされる Usenet システムで実際に操作しているかのような感じのするバックエンドです。(実際のところ、nnml はすごく以前に nnsPOOL から枝分かれしたものだと思います。) メールは元のスプールファイルから取り出された後で、個々のファイルに 1:1 で切り分けられます。Usenet 様式のアクティブファイル (INN や CNews に基づいたニュースシステムの ‘/var/lib/news/active’ ファイル (例えば) や、‘NNTP LIST’ 命令で返されるものに類似したもの) を維持し、今ではかなりの年数にわたって NNTP サーバーのために定義されている overview ファイルも、グループへ入るときの効率を良くするために作成します。たくさんのファイルを作成し、nnml アクティブファイルを更新し、さらにメッセージ毎に overview への追加を行なうので、メール分割では遅くなりますが、アクセスするときには、アクティブファイルと overview によって提供される索引機能に支援されて、とてつもなく速くなります。

nnml は inode を非常にたくさん消費します。すなわち、新しいファイルを置くことができる場所をファイルシステム上に定めるための資源を、たくさん占有します。ぎっしりつまった共有ファイルシステムで大量の inode を占有することを、システム管理者は快く思いません。もっとも、そのファイルシステムが自分自身のもので、容量が希少ではない個人のマシンにいるのなら、nnml には非常に大きな利点があるのですが。

FAT16 の Windows の世界にいる場合にも、たくさんの小さなファイルで多くの場所を取られてしまう点で問題があります。

nnmh Rand MH メール閲覧システムは UNIX システムにかなり長い間存在しています。それはメッセージのスプールファイルを個々のファイルに分割することによって動作しますが、索引機能は少ししか、あるいはまったくありません—nnmh は、意味的には「アクティブファイルまたは overview の無い nnml」と等価です。これはおそらく最悪の選択でしょう。なぜならば、個々のファイルを作ることの遅さが、何がグループで新しいかを知るときに解析するために行なうアクセスの遅さに結び付くからです。

nnfolder 基本的に nnfolder が実現することは、グループ毎の nnmbox (上で説明されている最初の方法) です。すなわち nnmbox 自体は すべての のメールを一つのファイルに入れます。でも nnfolder はメールグループのそれぞれが Unix mail box ファイルを持つように、ほんの少し最適化をします。それぞれのグループは別々に解析されるので nnmbox よりも速く、しかもなお、メールを移動させるのに最小限の労力しか要求しない、単純な Unix mail box 形式を提供します。加えて「アクティブ」ファイルを維持し、Gnus がそれぞれの別のグループにどのくらいのメッセージがあるかを調べることをとても速くします。

もしたくさんの量のメッセージを受け取ることが予想されるグループがあるなら、nnfolder は最善の選択ではありませんが、ほどほどの量のメールしか受け取らないなら、おそらく nnfolder はすべての中で最も都合の良いバックエンドでしょう。

nnmaildir

期限切れ消去その他もろもろを設定するのに、nnmaildir は他のメールバックエンドとは少々異なった、互換性の無いグループパラメーターを使います。

nnmaildir は大方 nnml と似たものですが、いくらか顕著な違いがあります。それぞれのメッセージは別々のファイルに格納されますが、ファイル名は Gnus の記事番号と関係がありません。また nnmaildir は nnml の overview に相当するファイルを記事ごとに一つ格納するので、nnml の約二倍の量の i ノードを使います。(df -i を使って i ノードの割り当てがどれほどたくさんあるかを調べて下さい。) そのために遅くなったり多くの場所を取ってしまうようならば、ReiserFS (<http://www.namesys.com/>) や他の非ブロック構造のファイルシステムへの転換を検討して下さい。

maildir は受信配送のためのロックを必要としないので、あなたがグループとして使っている maildir は、配送されてきたメールを直接受け取るための maildir にすることもできます。これは、メールが配送されてくる過程で異なるメールボックスに仕分けられるようになっているのならば、Gnus のメール分割を省略できることを意味します。mail-sources における directory の項には (訳注: maildir を使わなくても) 似た効果がありますが、配送されてくるメールをスプールするためのメールボックスの一揃い (mbox 形式ではそのためにメッセージの本文が壊れる) と、他の (何であれあなたの好みの形式の) グループとして使われる組が必要です。一方 maildir は、new/ サブディレクトリーに置かれる組み込みスプールを持ちます。メール分割による代わりに new/ から cur/ に移動されたメールは、ダブっているかどうかをチェックするような処理を今のところは受けないことに注意してください。

nnmaildir はグループの記事の印を、それに対応する maildir に格納します。Gnus の外からそれらを簡単に操作できるようにするために、そのように作られているのです。maildir を tar でまとめてから別のどこかで展開しても、印はそのままです。nnml も印を格納しますが、nnmaildir で Gnus の外からそれらを使うように簡単ではありません。

nnmaildir は速度を上げるためにかなりの量のメモリを使います。(nnml の場合はファイルに格納し、nnmh では何度もメッセージファイルを解析して得るものごとを、それはメモリ上に保持します。)これがあなたにとって問題ならば、`nov-cache-size` グループパラメーターを何か小さな値 (0 はおそらくだめですが 1 だったらたぶん働きます) に設定することによって、少ないメモリで済むようにすることができます。このキャッシュ機構は、おそらく将来は削除されるでしょう。

起動は他のバックエンドよりも nnmaildir の方が遅いでしょう。ファイルシステムに依存していないすべての部分では速いでしょう。

nnmaildir は nnnoo を使わないので、nnmaildir から派生したバックエンドを書くのに nnnoo は使えません。

6.4 Browsing the Web

ウェブに基づいた議論の場はどんどん広まっています。多くの分野で、ウェブの掲示板は最も重要な場になり、メーリングリストやニュースグループの重要性を翳らせています。理由は簡単です—新しい利用者が使い易いからです。ただ場所をクリックするだけで、議論の場があります。メーリングリストでは、面倒な購読手続きをしなければならず、ほとんどの人はニュースグループというものが何であるかすら知りません。

この筋書きから浮かび上がる問題は、ウェブブラウザはニュースリーダーとしてはあまり良くないということです。どんな記事を読んだかを記録しません。興味のある表題にスコアを付けることができません。オフラインで読むことができません。何度もクリックすることを要求し、最後にはあなたを怒らせます。

ならば—ウェブブラウザが掲示板を読むのに適していないのなら、代わりに Gnus を使いませんか？

Gnus はこれらのソースへのインターフェースを提供するバックエンド群を少し備えています。

すべてのウェブソースは、動作させるために Emacs/W3 と url ライブラリー、またはそれらの代替が必要です。

これらのウェブソースの一番の問題は、長期間は動作しない可能性が高いことです。HTML のデータから情報を拾い集めるのはせいぜい推測で、その構造が変化したときには、Gnus バックエンドは動作しません。でも、ある程度新しいバージョンのバックエンドを使っていれば大丈夫のはずです。

これらのウェブの手段に共通することは、ウェブソースはしばしば落ちていたり、使用可能でなかったり、はっきり言って楽しむには遅すぎる、ということです。そういう場合に、Gnus Agent (see Section 6.9 [Gnus Unplugged], page 210) に記事のダウンロードを任せて、ローカルディスクから好きなときに読むようにすることは、大いに意義があります。これで World Wide Wait とはおさらばです。

6.4.1 メールの保存

いくつかのバックエンド、特に nnml, nnfolder および nnmaildir は、今ではそれぞれのグループの記事の印を本当に保持するようになりました (訳注: そうなったのはだいぶ前ですが)。これらのサーバーで、グループの印を保ちつつ保存したり元に戻すのはかなり簡単です。

(でも、グループレベルとグループパラメーターをも保持するには、今までとおり ‘.newsrsrc.eld’ の神に、舞いと生贄を捧げなければなりません。)

nnml, nnfolder または nnmaildir サーバーにまるごと保存するには、サーバーのディレクトリーを再帰的にコピーして下さい。Gnus を終了する必要は無いので、保存は cron やそれに類するものを行なうことができます。データを復歸させるにはディレクトリー木 (tree) を元に戻すこと

によって行ない、そのディレクトリーを指し示すように Gnus のサーバーの定義に追加しましょう。Section 3.14 [Article Backlog], page 73, Section 3.11 [Asynchronous Fetching], page 70 およびその他のものはデータを上書きして邪魔をするかもしれないので、データを復帰させる前に Gnus を終了する必要があるかもしれません。

さらに、個々の `nnml`, `nnfolder` または `nnmaildir` のグループを、印を保持しつつ保存することもできます。`nnml` または `nnmaildir` では、そのグループのディレクトリーにあるすべてのファイルをコピーして下さい。`nnfolder` では、基本のフォルダーファイルそのもの（例えば `'F00'`）と印ファイル（`'F00.mrk'`）の両方をコピーする必要があります。グループを元に戻すには、グループバッファで `G m` キーを使いましょう。その最後の手順が、Gnus に新しいディレクトリーができたことを知らせます。`nnmaildir` は自動的に新しいディレクトリーを知るので、その場合 `G m` は不要です。

6.4.2 ウェブ検索

ううむ、まあ、調べたい文字が書いてある記事を、その、Usenet で探すということは、ですね、もちろん素晴らしいことこの上ないのではありませんが、しかし、何と申しましょうか、ウェブブラウザというものは、そういうものを使ってことを行なうのは、何ともその、はばかりながら、ぶざまと言いますか、そうすると、コマーシャルを見ないわけにはいかないのでありまして、しかるに、Gnus を使えばブラウザ無しで検索することができます。

`nnweb` バックエンドは、強力な検索エンジンへの簡単なインターフェースを提供します。`nnweb` グループを作成し、検索パターンを入力してから、そのグループに入って他の普通のグループのように記事を読んで下さい。グループバッファ (see Section 2.9 [Foreign Groups], page 21) の `G w` 命令によって、手軽にこれができます。

`nnweb` グループは、固定グループになるうとはしません—このグループでは記事番号はごく一時的なものとして扱われます。実際、`nnweb` グループに入るたびに（たとえ検索パターンを変更していても）、記事の順序が違っているかもしれません。また、重複抑制 (see Section 3.29 [Duplicate Suppression], page 110) を使っても役に立たないでしょう。というのは、検索エンジン（例えば Google）を使って記事を読み込む前の段階では、`nnweb` はそれらの Message-ID を知らないからです。あなたが読んだ記事を覚えておくための唯一の方法は、Date ヘッダーを元にスコアを付けることだけです—つまり、そのグループを最後に読んだ日付より前に投稿された記事を、すべて既読にするということです。

もし検索エンジンの出力形式が実質的に変更されると、`nnweb` はそれをうまく解釈できなくて、処理に失敗するでしょう。ウェブの提供者たちがそんなことをしても、彼らを責めることはできないでしょう—それは広告で金を稼ぐのが彼らの レーゾン・デートル（存在理由）であり、社会にサービスを提供することではないからです。`nnweb` はすべての記事から広告を洗い流してしまうので、提供者たちがムカついていると思われるかもしれません。まあ見て下さい。

`nnweb` を使うには、`url` と `W3` パッケージ、またはそれらの代替（`'mm-url'` 変数グループに対し `customize-group` を試してみてください）をインストールしておかなくてはなりません。

以下は仮想サーバー変数です。

`nnweb-type`

どの検索エンジンを使うかを指定します。現在サポートされている種類は、`google`, `dejanews` そして `gmene` です。`dejanews` は `google` の別名になっていることに注意して下さい。

`nnweb-search`

検索エンジンに与える検索文字列です。

`nnweb-max-hits`

一つの検索で表示する最大のヒット数の希望値で、デフォルトは 999 です。

nnweb-type-definition

種類と定義の連想リストです。この連想リストは、さまざまな検索エンジンの種類に対して、nnweb がどうすべきかを指定します。以下に示す要素を与えなくてはなりません。

article	記事をデコードし、Gnus が理解できる何かを提供する関数です。
map	メッセージヘッダーと URL を、記事番号を元にして得るための連想リストを作成する関数です。
search	検索エンジンに検索文字列を送るための関数です。
address	前述の関数が検索文字列を送るべきアドレスです。
id	Message-ID を元にして記事を取得するための、URL フォーマットの文字列です。

6.4.3 Slashdot

Slashdot (<http://slashdot.org/>) は人気のあるニュースサイトで、ニュース記事に関して活発な議論がなされています。nnslashdot では便利な方法でこの会議室を読むことができます。

ここから読むための一番簡単な方法は、以下のようなものを ‘~/gnus.el’ ファイルに入れることです。

```
(setq gnus-secondary-select-methods
      '((nnslashdot "")))
```

これは Gnus に nnslashdot バックエンドに対して新しいコメントとグループを尋ねさせます。F 命令はそれぞれの新しいニュース記事を新しい Gnus のグループとして作るので、これらのグループに入ることによってコメントを読むことができます。(デフォルトの講読方法では、新しいグループをゾンビとして作ることに注意して下さい。他の方法を使うこともできます (see Section 1.5.2 [Subscription Methods], page 6)。

古い nnslashdot グループを削除したい場合には、G DEL が最も手軽な道具です (see Section 2.9 [Foreign Groups], page 21)。

nnslashdot のコメントにフォローアップ (または新しいコメントの投稿) をするときは、いくつかの軽い HTML 変換が行なわれます。特に ‘>’ で引用されたテキストは代わりに ‘blockquote’ で引用され、署名にはそれぞれの行の最後に ‘br’ が追加されます。それ以外は、メッセージバッファーに直接 HTML を書くことができます。Slashdot はいくつかの HTML 様式をふるい落とすことに気を付けて下さい。

以下の変数で振る舞いを変えることができます:

nnslashdot-threaded

nnslashdot がグループをスレッドで表示するかどうかを指定します。デフォルトは t です。スレッドを表示できるようにするためには、nnslashdot はグループのすべてのコメントを完全に取得する必要があります。スレッド表示が要求されていないと、nnslashdot は実際に利用者が欲しいコメントだけを取得します。スレッドの方が快適ですが、スレッド無しの場合より、ずっとずっと遅くなります。

nnslashdot-login-name

投稿時に使うログイン名です。

nnslashdot-password

投稿時に使うパスワードです。

nnslashdot-directory

nnslashdot がファイルを保存する場所です。デフォルトは ‘~/News/slashdot/’ です。

nnslashdot-active-url

ニュース記事の情報とコメントを取得するために使われる URL のフォーマット文字列です。デフォルトは ‘http://slashdot.org/search.pl?section=&min=%d’ です。

nnslashdot-comments-url

コメントを取得するために使われる URL のフォーマット文字列です。

nnslashdot-article-url

ニュース記事を取得するために使われる URL のフォーマット文字列です。デフォルトは ‘http://slashdot.org/article.pl?sid=%s&mode=nocomment’ です。

nnslashdot-threshold

スコアのしきい値です。デフォルトは -1 です。

nnslashdot-group-number

最新の 10 個に加えて更新を続ける古いグループの数です。デフォルトは 0 です。

6.4.4 Ultimate

The Ultimate Bulletin Board (<http://www.ultimatebb.com/>) はおそらく一番良く使われているウェブ掲示板システムでしょう。とても規則的で良質なインターフェースを持っていて、Gnus がグループを最新の状態に保っておくために必要な情報を得ることが可能です。

nnultimate を始めるための一番簡単な方法は、グループバッファで以下のようなものをすることです: *B nnultimate RET http://www.tcj.com/messboard.ubbcgi/ RET*。(興味のある会議室の URL (‘Ultimate.cgi’ 等を最後に含んでいないもの) に変えて下さい。Ultimate ウェブサイトにはたくさん挙げられています。) それからサーバーバッファで興味のあるグループを購読し、グループバッファからそれらを読んで下さい。

以下の nnultimate 変数が変更可能です:

nnultimate-directory

nnultimate がファイルを保存するディレクトリーです。デフォルトは ‘~/News/ultimate/’ です。

6.4.5 Web Archive

いくつかのメーリングリストは、<http://www.egroups.com/> や <http://www.mail-archive.com/> のようなウェブサーバーでだけ、そのアーカイブを持っています。それはとても規則的で良質なインターフェースを持っているので、Gnus がグループを最新の状態に保っておくために必要な情報を得ることが可能です。

nnwarchive を始めるための一番簡単な方法は、グループバッファで以下のようなものをすることです: *M-x gnus-group-make-warchive-group RET an_egroup RET egroups RET www.egroups.com RET your@email.address RET*。(an_egroup を購読しているメーリングリストに、your@emailaddress を電子メールアドレスに置き換えて下さい。) または *B nnwarchive RET mail-archive RET* でバックエンドをブラウズして下さい。

以下の nnwarchive 変数が変更可能です:

nnwarchive-directory

nnwarchive がファイルを保存するディレクトリーです。デフォルトは ‘~/News/warchive’ です。

`nnwarchive-login`

ウェブサーバーでのアカウント名です。

`nnwarchive-passwd`

ウェブサーバーでのアカウントのパスワードです。

6.4.6 RSS

いくつかのウェブサイトは RDF site summary (RSS) を持っています。RSS は、ニュース関連のサイト (BBC や CNN のような) の見出しを要約するためのフォーマットです。しかし、基本的にはリストのようなものなら何でも、RSS feed として提供することができます: weblogs, changelogs あるいは wiki (例えば <http://cliki.net/recent-changes.rdf>) の最新の変更などが対象になります。

RSS はとても規則的で良質なインターフェースを持っているので、Gnus がグループを最新の状態に保っておくために必要な情報を得ることが可能です。

注: utf-8 coding system をサポートする Emacs を使うのが良いでしょう。RSS は非-ASCII テキストをエンコードするために、デフォルトで UTF-8 を使うからです。それはまた、非-ASCII グループ名にもデフォルトで使われます。

Feed を講読するには、グループバッファから `G R` を使って下さい—feed の所在、タイトルおよび説明の入力を求められるでしょう。タイトルはどんな文字でもよく、それはグループ名とグループのデータ・ファイルの名前に使われます。説明は省略できます。

簡単に `nnrss` を始める方法は、グループバッファで `B nnrss RET RET y` のようなことを唱え、そしてグループを講読することです。

`nnrss` バックエンドは、それぞれの `nnrss` グループのためのデータ・ファイルを `nnrss-directory` (下記参照) に保存します。非-ASCII 文字を含んでいるファイル名は、`nnmail-pathname-coding-system` 変数で指定された coding system でエンコードされます。それが `nil` であると、Emacs では coding system はデフォルトで `default-file-name-coding-system` の値になります。あなたが XEmacs を使っていて、非-ASCII グループ名を使いたければ、`nnmail-pathname-coding-system` 変数の値を適切に設定しなければなりません。

`nnrss` バックエンドは、それぞれが 'text/plain' パートと 'text/html' パートを含んでいる 'multipart/alternative' 型の MIME 記事を作ります。

あなたの講読目録を OPML フォーマット (Outline Processor Markup Language) でロード/セーブするために、以下のコマンドを使うこともできます。

`nnrss-opml-import file` [Function]

OPML ファイルの入力を促し、そのファイルにあるそれぞれの feed を講読します。

`nnrss-opml-export` [Function]

現在の RSS 講読目録を OPML フォーマットでバッファに書き出します。

以下の `nnrss` 変数が変更可能です:

`nnrss-directory`

`nnrss` がファイルを書き込むディレクトリーで、デフォルトは `'~/News/rss/'` です。

`nnrss-file-coding-system`

`nnrss` グループのデータ・ファイルを読み書きするときに使われる coding system です。デフォルトは `mm-universal-coding-system` の値 (そのデフォルトは Emacs では `emacs-mule`、XEmacs では `escape-quoted`) です。

`nnrss-use-local`

`nnrss-use-local` を `t` に設定すると、`nnrss` は `nnrss-directory` にあるローカルファイルから `feed` を読みます。`nnrss-generate-download-script` コマンドを使うことによって、`wget` を使ったダウンロード・スクリプトを作ることができます。

`nnrss-wash-html-in-text-plain-parts`

`nil` ではない値にすると `nnrss` は `'text/plain'` パートにあるテキストを HTML として描画します。テキストの描画には `mm-text-html-renderer` 変数 (see section “表示のカスタマイズ” in *The Emacs MIME Manual*) で定義された関数が使われます。`nil` だったら (それがデフォルトです)、テキストは単に折り返されるだけです。もし `'text/html'` パートを見るのが好みならば、`nil` のままにしておいて下さい。

概略バッファに説明を表示させたいならば、以下のコードが役に立つでしょう。

```
(add-to-list 'nnmail-extra-headers nnrss-description-field)
(setq gnus-summary-line-format "%U%R%Z%I%([%4L: %-15,15f%]) %s%uX\n")
```

```
(defun gnus-user-format-function-X (header)
  (let ((descr
        (assq nnrss-description-field (mail-header-extra header))))
    (if descr (concat "\n\t" (cdr descr)) "")))
```

以下のコードは、概略バッファから直接 `nnrss` の url をオープンするのに便利かもしれません。

```
(require 'browse-url)

(defun browse-nnrss-url( arg )
  (interactive "p")
  (let ((url (assq nnrss-url-field
                  (mail-header-extra
                   (gnus-data-header
                    (assq (gnus-summary-article-number)
                          gnus-newsgroup-data))))))
    (if url
        (progn
          (browse-url (cdr url))
          (gnus-summary-mark-as-read-forward 1))
        (gnus-summary-scroll-up arg))))

(eval-after-load "gnus"
  #'(define-key gnus-summary-mode-map
    (kbd "<RET>") 'browse-nnrss-url))
(add-to-list 'nnmail-extra-headers nnrss-url-field)
```

あなたが HTML パートを見たくないために `"text/html"` を `mm-discouraged-alternatives` 変数 (see section “表示のカスタマイズ” in *The Emacs MIME Manual*) に加えていたとしても、特に `nnrss` グループでは `'text/html'` を表示の方が便利かもしれません。以下は `nnrss` グループでだけは `'text/html'` パートを表示するために、グループパラメーターとして `mm-discouraged-alternatives` を設定する例です:

```
; ; mm-discouraged-alternatives のデフォルト値を設定。
(eval-after-load "gnus-sum"
```

```

' (add-to-list
  'gnus-newsgroup-variables
  ' (mm-discouraged-alternatives
    . ' ("text/html" "image/*.*)"))))

;; nnrss グループでは 'text/html' パートを表示。
(add-to-list
  'gnus-parameters
  ' ("\\'nnrss:" (mm-discouraged-alternatives nil)))

```

6.4.7 W3 のカスタマイズ

Gnus はウェブページを取得するために url ライブラリーを、ウェブページを表示するために Emacs/W3 を (またはそれらの代替を) 使います。Emacs/W3 のことはそのマニュアルに記載されていますが、ここでは Gnus の利用者にとってより適切な、いくつかの事柄を述べることにします。

例えばよくある質問は、Emacs/W3 に browse-url の関数 (Netscape のような外部ブラウザーを呼びます) を使ってリンクを参照させるにはどうしたらよいか、というものです。以下は一つの方法です:

```

(eval-after-load "w3"
  '(progn
    (fset 'w3-fetch-orig (symbol-function 'w3-fetch))
    (defun w3-fetch (&optional url target)
      (interactive (list (w3-read-url-with-default)))
      (if (eq major-mode 'gnus-article-mode)
          (browse-url url)
          (w3-fetch-orig url target)))))

```

これをあなたの .emacs ファイルに書き込んで下さい。そうすれば、Gnus の記事バッファで W3 が描画した HTML リンクを叩くと、browse-url を使ってそのリンクを参照してくれるでしょう。

6.5 IMAP

IMAP はメール (もしくは、ニュース、もしくは ...) を読むためのネットワークプロトコルです。現代風の NNTP と考えて下さい。IMAP サーバーへの接続はニュースサーバーへの接続と非常に似ていて、そのサーバーのネットワークアドレスを指定するだけになっています。

IMAP には二つの特質があります。一つは、IMAP は POP でできることは全部できる、それゆえ POP++ のようにも見えるということ。二つ目は、IMAP は NNTP がニュースを貯蔵するプロトコルであるように、メールを貯蔵するプロトコルであるということです。しかしながら IMAP は NNTP より多くの機能を提供します。メールは読み書きができるのに対して、ニュースはだいたいにおいて読むだけですから。

IMAP を POP++ のように使いたいときは、mail-sources に imap の項目を登録して下さい。これによって Gnus は IMAP サーバーからメールを取り込んで、ローカルディスクに格納します。ここではその使い方の説明はしませんから、Section 6.3.4 [Mail Sources], page 148 を参照して下さい。

IMAP をメールを貯蔵するプロトコルとして使いたいときは、gnus-secondary-select-methods に nnimap の項目を登録して下さい。これによって Gnus は IMAP サーバーに格納されているメールを操作するようになります。これがここで説明する種類の使い方です。

いくつかの IMAP サーバーを ‘~/gnus.el’ で設定すると、たぶん以下のようなものになるでしょう。(注: TLS/SSL では外部プログラムとライブラリーが必要です。以下を参照して下さい。)

```
(setq gnus-secondary-select-methods
      '( (nnimap "simpleserver") ; 特殊ではない設定
          ; もしかしたら SSH ポートをフォワードしたサーバー:
          (nnimap "dolk"
                  (nnimap-address "localhost")
                  (nnimap-server-port 1430))
          ; ローカルホストで稼働している UW サーバー:
          (nnimap "barbar"
                  (nnimap-server-port 143)
                  (nnimap-address "localhost")
                  (nnimap-list-pattern ("INBOX" "mail/*")))
          ; 匿名で使える cyrus の公衆サーバー:
          (nnimap "cyrus.andrew.cmu.edu"
                  (nnimap-authenticator anonymous)
                  (nnimap-list-pattern "archive.*")
                  (nnimap-stream network))
          ; 標準ではないポートの SSL サーバー:
          (nnimap "vic20"
                  (nnimap-address "vic20.somewhere.com")
                  (nnimap-server-port 9930)
                  (nnimap-stream ssl))))
```

新しいサーバーを定義した後で、*U* のような Gnus の通常のコマンドをグループバッファで使う (see Section 2.4 [Subscription Commands], page 18) か、またはサーバーバッファ (see Section 6.1 [Server Buffer], page 133) を介して、そのサーバーのグループを講読することができます。

以下の変数は仮想 nnimap サーバーを作成するために使うことができます。

nnimap-address

遠隔 IMAP サーバーのアドレスです。指定されていない場合は仮想サーバーの名前になります。

nnimap-server-port

接続するサーバーのポートです。デフォルトはポート 143、または TLS/SSL では 993 です。

これは整数でなければならないことに注意して下さい。以下はサーバー指定の例です:

```
(nnimap "mail.server.com"
      (nnimap-server-port 4711))
```

nnimap-list-pattern

使うことができるグループを制限するための、メールボックスの文字列または文字列のリストです。これは、サーバーに非常に多くのメールボックスがあるけれど、興味のあるものは少しだけであるときに使用します。いくつかのサーバーはホームディレクトリーを IMAP 経由でアクセスできるようにするので、その場合はおそらくメールボックスを ‘~/Mail/*’ に制限したいでしょう。

文字列は REFERENCE と上記の文字列との cons であることもできます。どの REFERENCE が使用されるかはサーバーによりますが、ワシントン大学のサーバーでは、メールボックスと連結されるディレクトリーです。

以下はサーバー指定の例です:

```
(nnimap "mail.server.com"
  (nnimap-list-pattern ("INBOX" "Mail/*" "alt.sex.*"
    ("~friend/Mail/" . "list/*"))))
```

nnimap-stream

サーバーに接続するときに使われるストリームの型です。デフォルトでは nnimap は TLS/SSL を除く以下のすべてを自動的に検知してそれを使います。(TLS/SSL を使う IMAP は STARTTLS で置き換えられています。これは自動検出できますが、まだ広範囲に配備されていません。)

以下はサーバー指定の例です:

```
(nnimap "mail.server.com"
  (nnimap-stream ssl))
```

nnimap-stream の値はシンボルであることに注意して下さい!

- *gssapi*: GSSAPI (普通は Kerberos 5) で接続します。‘gsasl’ または ‘imtest’ プログラムが必要です。
- *kerberos4*: Kerberos 4 で接続します。‘imtest’ プログラムが必要です。
- *starttls*: STARTTLS 拡張 (TLS/SSL に類似) を介して接続します。‘starttls.el’ 外部ライブラリーと ‘starttls’ プログラムが必要です。
- *tls*: TLS を通して接続します。GNUTLS (‘gnutls-cli’ プログラム) が必要です。
- *ssl*: SSL を通して接続します。OpenSSL (‘openssl’ プログラム) か SSLeay (‘s_client’) が必要です。
- *shell*: シェル命令を使って IMAP 接続を開始します。
- *network*: 生の TCP/IP のネットワーク接続です。

‘imtest’ プログラムは Cyrus IMAPD に含まれています。もし 2.0.14 未満の Cyrus IMAPD に含まれている ‘imtest’ (バージョン 1.5.x および 1.6.x) を使っているなら、imap-process-connection-type をいじって、‘imap.el’ が ‘imtest’ と通信するときに、パイプではなくて pty を使うようにさせる必要があります。そうするとあなたは IMAP コマンドの行の長さの制限に悩まされ、メールボックスにたくさんの記事がある場合には、Gnus が無期限にハングするよう見えるかもしれません。変数 `imap-kerberos4-program` は imtest プログラムに渡すパタメータを含みます。

TLS 接続では GNUTLS 由来の `gnutls-cli` プログラムが必要です。<http://www.gnu.org/software/gnutls/> から手に入れることができます。

このパラメーターは、副シェルで GSSAPI 認証による IMAP 接続を起動するためのコマンド行のリストを指定します。これらは接続が確立するまで、またはリストが使い尽くされるまで、順ぐりに試されます。デフォルトでは、<http://www.gnu.org/software/gsas1/> にある GNU SASL による ‘gsasl’ と、Cyrus IMAPD による ‘imtest’ プログラム (`imap-kerberos4-program` 参照) が試されます。

SSL 接続のための OpenSSL プログラムは <http://www.openssl.org/> から入手できます。OpenSSL は以前は SSLeay として知られていたもので、nnimap はそれもサポートします。しかし SSLeay の最新版である 0.9.x には、それを役に立たなくしてしまう重大なバグがあることが知られています。以前の版、特に SSLeay 0.8.x は使えることがわかっています。変数 `imap-ssl-program` は OpenSSL/SSLeay に渡すパタメータを含みます。

shell ストリームを使う IMAP 接続では、何のプログラムを呼ぶかを変数 `imap-shell-program` で指定します。

nnimap-authenticator

サーバーに接続するために使われる認証手段です。デフォルトでは nnimap はサーバーにできる最も安全な認証手段を使います。

以下はサーバー指定の例です:

```
(nnimap "mail.server.com"
      (nnimap-authenticator anonymous))
```

nnimap-authenticator の値はシンボルであることに注意して下さい!

- *gssapi*: GSSAPI (普通は Kerberos 5) 認証です。外部プログラム `gsasl` または `imtest` が必要です。
- *kerberos4*: Kerberos 4 による認証です。外部プログラム `imtest` が必要です。
- *digest-md5*: DIGEST-MD5 で暗号化された 利用者名/パスワード です。外部ライブラリー '`digest-md5.el`' が必要です。
- *cram-md5*: CRAM-MD5 で暗号化された 利用者名/パスワード です。
- *login*: LOGIN 経由での生の 利用者名/パスワード です。
- *anonymous*: 電子メールアドレスをパスワードとして与え、“ anonymous ”としてログインします。

nnimap-expunge-on-close

パルメニデスと違って、IMAP の設計者達は、存在していないものが実際には存在していると決めました。もっと詳しく説明すると、IMAP には実際には記事を消去せずに Deleted という印を記事に付けるという概念があります。これ (すなわち Deleted の印を付けること) が Gnus で記事を消去するときに nnimap が行なうことです (*B DEL* など)。(訳注: パルメニデスは紀元前 5 世紀のギリシアの Elea 派の哲学者。)

記事を Deleted フラグで印を付けたときには本当は消去されていないので、実際に消去するための方法が必要となります。まだどうどうめぐりをしているように感じますか?

伝統的に nnimap はメールボックスを閉じるときに Deleted という印の付いたすべての記事を消去してきましたが、今ではこのサーバー変数によって設定することが可能になりました。

可能な選択肢は:

- | | |
|---------------------|--|
| <code>always</code> | これがデフォルトの振舞いで、メールボックスを閉じるときに“ Deleted ”として印が付けられている記事を消去します。 |
| <code>never</code> | 決して記事を消去しません。現在は消去の印が付いた記事を nnimap で表示する方法はありませんが、他の IMAP クライアントではできるかもしれません。いつも手で EXPUNGE コマンドを発行したいならば、Section 6.5.4 [Expunging mailboxes], page 193 を参照して下さい。 |

`ask` メールボックスを閉じるときに、`nnimap` が消去された記事を削除するかどうかを尋ねます。

`nnimap-importantize-dormant`

非-nil (デフォルト) だったら、他の IMAP クライアントのために保留記事に可視記事として (も) 印を付けます (訳注: 保留==`dormant`、可視==`ticked`)。Gnus の内部では、当然ながら保留記事には保留記事としてのみ印が付けられます。一方これは、保留記事を、他の IMAP クライアントにおいて、あたかも可視記事のようにきわ立たせます。(別の言い方をすると、Gnus には二つの「可視」印があり、IMAP にははたった一つだということです。)

おそらくこれをいじる唯一の理由は、あなたが利用者ごとに永続的な保留フラグを付けようとしているかどうかということでしょう。こんな感じで:

```
(setcdr (assq 'dormant nnimap-mark-to-flag-alist)
         (format "gnus-dormant-%s" (user-login-name)))
(setcdr (assq 'dormant nnimap-mark-to-predicate-alist)
         (format "KEYWORD gnus-dormant-%s" (user-login-name)))
```

この場合、あなたは利用者ごとの保留フラグが付いている記事が、他の利用者には可視記事として見えるようにしたくないのでしょうね。

`nnimap-expunge-search-string`

この変数には、期限切れ消去するのが望ましい記事を探すときにサーバーに送った IMAP の検索コマンドが入っています。デフォルトは `"UID %s NOT SINCE %s"` で、ここで最初の `%s` は UID の一揃いで置き換えられ、二番目の `%s` は日付で置き換えられます。この代わりに使いものになる値はたぶん `"UID %s NOT SENT SINCE %s"` だけで、それは `nnimap` に記事の内部的な日付の代わりに `Date:` を使うようにさせます。使うことができる文字列に関するさらなる情報は、RFC 2060 の第 6.4.4 章を見て下さい。

しかしながら `nnimap-search-uids-not-since-is-evil` が真になっていると、後述のように検索論理が反転されるので、この変数は無効になります。

`nnimap-authinfo-file`

サーバーにログインするために使う認証情報 (credentials) を含むファイルです。その形式は `ftp` の `~/netrc` ファイルと (ほとんど) 同じです。厳密な様式については、変数 `nnntp-authinfo-file` を見て下さい。そして Section 6.2.1 [NNTP], page 138 も見て下さい。IMAP サーバー用の `.authinfo` 行の例です:

```
machine students.uio.no login larsi password geheimnis port imap
```

安全な IMAP で使われる実際のポート番号は `port 993` ですが、`nnimap-stream` として `tls` または `ssl` を使う場合、それは `port imap` または `port 143` でなければならぬことに注意して下さい。便宜上 Gnus は `port imap` の同義語として `port imaps` を受け入れます。

`nnimap-need-unselect-to-notice-new-mail`

メールボックス群で新着メールを探す前に、それらを未選択にします。いくつかのサーバーが、何らかの状況の元でこれを必要とするようです。Courier 1.7.1 はそうだという報告がありました。

`nnimap-nov-is-evil`

NOV データベースを作らないか、またはローカルのものを使います。デフォルトは `gnus-agent` の値です。

普通 NOV データベースを使うとヘッダーの取得がとて速くなりますが、ある種のサーバー（特にいくつかの Courier の版）では非常に遅い UID SEARCH UID コマンドを使います。Gnus エージェントは、その遅いコマンドを使わずに NOV データベースに情報をキャッシュするので、この変数のデフォルト値は、エージェントが使われる場合は真に、そうでない場合は偽になります。

nnimap-search-uids-not-since-is-evil

UID SEARCH UID *message numbers* NOT SINCE *date* コマンドを使わないようにします。それは、ある種の IMAP サーバー（特にいくつかの Courier の版）では非常に遅くなります。代わりに UID SEARCH SINCE *date* を使って、Gnus で期限切れ消去する記事のリストから余分なものを取り除きます。

Gnus がメールの期限切れ消去 (see Section 6.3.9 [Expiring Mail], page 163) を行うときは、期限切れ消去してもよい記事のリストが始めにあり、IMAP サーバーに「これらの記事のうち、一週間より古いものはどれ?」のような問いを發します。これは完全に合理的な質問に見えますが、見たところいくつかの IMAP サーバーは、すべての古い記事について期限切れ消去の対象かどうかを調べるので、それに答えるために長い時間がかかります。不思議なことに「すべての記事のうち、一週間より古いものはどれ?」という質問の方がめっぽう速く答えが返ってくるので、この変数を設定して Gnus にこの質問を出させることによって、本来の質問そのものへの答を得ることができるのです。この問題は実際にあなたに忍び寄ってくるでしょう。最初のころは、Gnus を設定して、すべてがうまくいったとしても、いったん二～三千通のメッセージが溜まったならば、あなたは Gnus の遅いことをのしりはじめましょう。一方、あなたが大量の電子メールをたった一週間で受け取るのであれば、この変数を設定すると Gnus と IMAP サーバー間の通信量の増大を招くでしょう。

6.5.1 IMAP での分割

分割は Gnus の利用者が何年も愛用してきたもので、今や残りの世界も追いつこうとしています。ええ、彼らには勝手に夢を見てもらいましょう。サーバー側で分割できる IMAP サーバーはあまり多くなく、しかもそれらは標準ではないプロトコルを使っているようです。つまり Gnus の IMAP サポートは、自分自身で分割をしなければならないということです。

そして実際にします。

(ついでに言えば、みんなが夢見ていたのでしょうね。その結果 Sieve は市場占有率を増して、いくつもの IMAP サーバーによってサポートされるようになりました。幸いに Gnus もそれをサポートします。See Section 2.17.5 [Sieve Commands], page 42.)

関連する変数は三つです:

nnimap-split-crosspost

nil でなければ、複数の分割規則がそのメールと合致したときにクロスポストをします。

nil ならば、nnimap-split-rule で最初に見つかったものが使われます。

Nnmail で対応するもの: nnmail-crosspost.

nnimap-split-inbox

分割の元となる IMAP のメールボックスの名前を指定する文字列か文字列のリストです。デフォルトは nil で、分割は使用しないようになっています!

```
(setq nnimap-split-inbox
      '("INBOX" ("~/friend/Mail" . "lists/*") "lists.imap"))
```

Nnmail に対応するものではありません。

`nnimap-split-rule`

`nnimap-split-inbox` で見つかった新しいメールは、この変数に従って分割されます。この変数はリストのリストから成ります。副リストの最初の要素は IMAP のメールボックスで、二つめの要素の正規表現に合致した記事の移動先を指定します。わかりましたか? いいえ、私もわかりません。例が必要です。

```
(setq nnimap-split-rule
  '(("INBOX.nnimap"
     "^Sender: owner-nnimap@vic20.globalcom.se")
    ("INBOX.junk"    "^Subject:.*MAKE MONEY")
    ("INBOX.private" "")))
```

これは `nnimap` メーリングリストからのすべての記事をメールボックス `INBOX.nnimap` に入れ、`Subject:` 行に `MAKE MONEY` のあるすべての記事を `INBOX.junk` に入れ、その他すべてのものを `INBOX.private` に入れます。

最初の文字列は、`replace-match` で合致したテキストから副表現を挿入するときに使用されるのと同じような、`'\1'` 形式を含むことができます。例えば:

```
("INBOX.lists.\1"      "^Sender: owner-\\([a-z-]+\1\)@")
```

最初の要素をシンボル `junk` にして、合致するメッセージを単に消すべきであることを表すこともできます。気を付けて使って下さい。

二つ目の要素は関数であることもできます。その場合は、その規則の最初の要素を引数として、記事のヘッダーがあるバッファーと呼ばれます。メールがそのグループに属すると考える場合は、`nil` でない値を返す必要があります。

`Nnmail` の利用者は、最後の正規表現はすべての記事に合致するように、空でなくてはならないことを覚えているかもしれません (上の例のように)。これは `nnimap` では必要ではありません。正規表現のどれにも合致しない記事は `inbox` から移動されません。(inbox に未読記事を大量に置いておくと、分割のコードは新しいメールを取得するときにそれらすべてを調べるので、実行速度に影響するかもしれません。)

これらの規則は連想リストの最初から終りに向かって実行されます。クロスポストを有効にしていない限り、最初に合致した規則が「勝ち」ます。有効にしている場合は、すべての合致した規則が「勝ち」ます。

この変数はその値として関数を持つこともできます。その関数は記事のヘッダーの部分に範囲が狭められた状態で呼ばれ、記事の移動先だと思ふグループを返すものでなければなりません。`nnimap-split-fancy` を参照して下さい。

分割コードは必要ならメールボックスを作成しようとしします。

異なる仮想サーバー毎に違う分割の規則を使ったり、それどころか同じサーバーの異なる `inbox` 毎に違う分割の規則を使うことができるようにするために、この変数の構文は以下のやり方で拡張されています。

```
(setq nnimap-split-rule
  '(("my1server"      (".*"      (("ding"      "ding@gnus.org")
                                   ("junk"      "From:.*Simon"))))
    ("my2server"      ("INBOX"   nnimap-split-fancy))
    ("my[34]server"   (".*"      (("private"   "To:.*Simon")
                                   ("junk"      my-junk-func)))))
```

仮想サーバー名は、同じ規則を複数のサーバーに適用できるように、実際には正規表現になっています。この例ではサーバー `my3server` と `my4server` の両方が同じ規則を

使います。同様に `inbox` 文字列も正規表現です。実際の分割の規則は、前に説明したように、関数か、グループ/正規表現またはグループ/関数を要素群とするリスト、の両方です。

Nnmail で対応するもの: `nnmail-split-methods`.

`nnimap-split-predicate`

この述語に合致する `nnimap-split-inbox` にあるメールは分割されます。これは文字列で、デフォルトは `'UNSEEN UNDELETED'` です。

`inbox` にあるメールを読むために別の IMAP クライアントを使っているが、購読度に関わらずにすべての記事を Gnus に分割させたいならば、これは役に立つかもしれません。その場合は `'UNDELETED'` に変えれば良いでしょう。

`nnimap-split-fancy`

特級分割を使いたいならば、`nnimap-split-rule` の値を `nnmail-split-fancy` に設定することができます。See Section 6.3.6 [Fancy Mail Splitting], page 157.

しかし `nnmail` と `nnimap` とで異なる特級分割方式を持つようにするには、`nnimap-split-rule` の値を `nnimap-split-fancy` に設定して、`nnimap` 特有の特級分割方式を `nnimap-split-fancy` に定義して下さい。

例:

```
(setq nnimap-split-rule 'nnimap-split-fancy
      nnimap-split-fancy ...)
```

Nnmail で対応するもの: `nnmail-split-fancy`.

`nnimap-split-download-body`

分割している最中にすべての記事をダウンロードするには、非-`nil` に設定して下さい。これは普通は必要としないし、ものごとを相当に遅くしてしまうでしょう。記事を分割するためにその本文を分析する高度な関数を使いたい場合には、必要かもしれません。

6.5.2 IMAP での期限切れ消去

`nnimap` は完全な `nnmail` 由来のバックエンドではありませんが、標準的なたいていの期限切れ消去 (see Section 6.3.9 [Expiring Mail], page 163) の機能をサポートします。IMAP の分割 (see Section 6.5.1 [Splitting in IMAP], page 190) では `nnmail` の変数を複製しない (例えば `nnimap-expiry-wait` を生成しない) 点が違うのですが、`nnmail` の変数を流用します。以下は `nnimap` の期限切れ消去の処理で使われる変数です。

また、期限切れ消去の印がどのように IMAP サーバーに記録されるかについても、ここで言及しておくのが適切でしょう。期限切れ消去の印は `imap` クライアント特有の印である `gnus-expire` に変換され、メッセージに記録されます。そうするのは、おそらく Gnus だけが適切に `gnus-expire` の印を理解して扱うからです。もっとも他のクライアントは、メッセージのクライアント特有のフラグを見させてくれるでしょうが。このことは、クライアント特有のフラグを恒久的にメッセージに保存することを、サーバーがサポートしなければならないことも意味します。たいていはサポートします。幸いにも。

もし IMAP メールの期限切れ消去がとても遅く感じられるのならば、サーバー変数 `nnimap-search-uids-not-since-is-evil` を設定することを試してみてください。

`nnmail-expiry-wait`

`nnmail-expiry-wait-function`

これらの変数は完全にサポートされています。期限切れ消去の値は、数、シンボルの `immediate` または `never` です。

nnmail-expiry-target

この変数はサポートされていて、内部的にはこれを扱う nnmail 関数を呼ぶことによって実装されています。それには、行き先が同じサーバーの IMAP グループだったら、記事を追加する代わりにコピーする（アップロードし直す）という最適化も含まれます。

6.5.3 IMAP の ACL を編集する

ACL は Access Control List (使用制限一覧) の略です。IMAP では、ACL は他の利用者によるあなたのメールボックスの使用を制限 (もしくは許可) するために使われています。すべての IMAP サーバーにこの機能があるわけではないので、無いサーバーでこれを使うとエラーが発生します。

あるメールボックスのための ACL を編集するには、`G l` (`gnus-group-edit-nnimap-acl`) をタイプして下さい。そうすると、詳しい説明をともなった ACL 編集ウィンドウが現れます。

使うことがありそうな例:

- あなたのメーリングリストのメールボックスで “anyone” に “lrs” 権 (lookup, read, seen/unseen フラグの保持) を与えることによって、そのリストに購読登録しなくても、同じサーバーの他の利用者が読むことができます。
- 少なくとも Cyrus のサーバーでは、誰もが “plussing” を使うことができるようにするためには、利用者 “anyone” に投稿 (“p”) の許可を与える必要があります (“plussing” とはすなわち、`user+mailbox@domain`宛てに送られたメールが `INBOX.mailbox` という IMAP のメールボックスに届くようにすることです)。

6.5.4 メールボックスの削除

`nnimap-expunge-on-close` に `never` を設定している場合には、メールボックスのすべての消去 (deleted) された記事を手動で削除 (expunge) する必要があるでしょう。まさにこれが `G x` が行なうことです。

今のところ消去された記事を表示する方法はありません。ただ消去できるだけです。

6.5.5 名前空間に関する注意

IMAP プロトコルには名前空間 (namespaces) と呼ばれる概念があり、以下の RFC2060 の文書で記述されています:

5.1.2. メールボックス名前空間命名規則

習慣により “#” で始まっているすべてのメールボックス名の最初の階層要素は、残りの名前の “名前空間” を示している。これは、それぞれ独自の名前空間を持つ異なるメールボックス保管の間での、曖昧さを取り除くことを可能にする。

例えば USENET ニュースグループへのアクセスを提供する実装は、USENET ニュースグループ名前空間を他のメールボックスから分離するために、“#news” 名前空間を用いてもよい。こうして `comp.mail.misc` ニュースグループは “#news.comp.mail.misc” というメールボックス名を持ち、名前 “comp.mail.misc” は別のオブジェクト (例えば、利用者の個人メールボックス) を指すことはありうる。

(訳註: <http://kame.zit.to/~obata/imap/rfc/rfc2060ja.txt> より転載)

Gnus における IMAP の実装を保証する記述がこの文書には無い一方で、いくつかのサーバーは Gnus のメールボックスの名前の使い方では動作しないやり方で名前空間接頭語 (namespace prefix) を使っています。

具体的には、ワシントン大学の IMAP サーバーは `#driver.mbx/read-mail` のようなメールボックス名を使っていて、それは `CREATE` と `APPEND` コマンドでだけ有効です。メールボックスが作られた後 (またはメッセージがメールボックスに追加された後) では、それは名前空間接頭語を付けずにアクセス、すなわち `read-mail` されなければなりません。Gnus は利用者が `CREATE` と `APPEND` コマンドだけで入力したメールボックス名を保証できないので、絶対に名前空間接頭語が付いたメールボックス名を Gnus で使ってはいけません。

`#driver.* /` について、どのように接頭語を使えば良いかについてのさらなる情報は、UoW IMAPD の文書を見て下さい。それらは強力な工具なので、どんな効果があるかが確かな場合だけ使して下さい。

6.5.6 IMAP のデバッグ

IMAP は NNTP や POP3 よりもっと複雑なプロトコルです。実装上のバグが無いとは言い切れないので、私たちは、すぐにそれらを直すために最善を尽くします。あなたが奇妙な振る舞いに出会ったとしたら、サーバーか Gnus のどちらかにバグがある可能性があります。

あなたが一般的なネットワーク・プロトコルに精通しているならば、Gnus とサーバーの間でやりとりされるプロトコル・ダンプを読むことによって、おそらくいくつかの手掛かりを抽出することができるでしょう。精通していなくても、プロトコル・ダンプを IMAP に関するバグ報告に含めれば、その問題の解決にとって重要なデータで私たちを助けることになります。したがって Gnus の IMAP バグを報告するときに、プロトコル・ダンプを含めることを強く奨励します。

プロトコル・ダンプは、それを有効にしているとたくさんのデータを生成するので、デフォルトでは無効になっています。有効にするには、以下のように `imap-log` を設定して下さい:

```
(setq imap-log t)
```

これはサーバーとのやりとりを何でも `imap.el` パッケージに記録させます。その記録は `*imap-log*` というバッファに格納されます。ときとして、BAD という札が付けられるエラーメッセージを探して下さい。でも、バグを提出するときは、すべてのデータを含めるようにして下さい。

6.6 その他のグループ源

Gnus はただ単にニュースやメールを読む以上のことができます。以下に示す方法によって、Gnus でディレクトリーやファイルを、あたかもニュースグループであるかのように閲覧することができるようになります。

6.6.1 ディレクトリーグループ

たくさんの記事が個別のファイルとして入っているディレクトリーがあれば、それをニュースグループとして扱うことができます。もちろん、ファイルは数字のファイル名をもっていなければなりません。

素晴らしい Emacs のパッケージの中でも最も素晴らしい `ange-ftp` (とその後継の `efs`) について触れるのに、ここは良い機会でしょう。私が `nndir` を書いたときは、これ (ディレクトリーを読むバックエンド) についてはあまり考えていませんでした。とんでもないことだね。

`ange-ftp` はこの状況を劇的に変化させました。例えばディレクトリー名として `ange-ftp` の様式で `/ftp.hpc.uh.edu:/pub/emacs/ding-list/` というファイル名をディレクトリー名として

入力したとすると、ange-ftp あるいは efs は実に「シナ」の向こうのディレクトリーをニュースグループとして読めるようになるのです。おーい、分散ニュースだぞーっ!

(訳注:「シナ」(原典 'sina') は China のことか?)

nndir は NOV ファイル群が存在すればそれらを利用します。

nndir は「読み出し専用」のバックエンドです—この選択方法では、記事の削除や期限切れ消去を行なうことはできません。nndir が使えるものなら何でも、nnmh あるいは nnml でも使うことができるので、もし読み出し専用ではない nndir が必要だと思ったら、これらのどちらかの方法に切り替えることもできます。

6.6.2 なんでもグループ

nneething は nndir バックエンド (単一のスプール風ディレクトリーを読むバックエンド) のほんの少し先にあるもので、それはどんなディレクトリーでもニュースグループに見せかけてしまいます。不思議ですが真実です。

nneething にディレクトリーを与えると、そのディレクトリーを走査して各ファイルに記事番号を割り当てます。そのようなグループに入ったら、nneething は Gnus が使える「ヘッダー」を作らなくてはなりません。つまるところ Gnus はニュースリーダーなんです。忘れていたかもしれないので念のため、nneething はこれを二段階で処理します。最初に、対象となるそれぞれのファイルを覗いてまわります。もしそのファイルが記事のように見えたなら (すなわち最初の数行がヘッダーのように見えたなら) それをヘッダーとして使います。もしそれがヘッダーの無いただの適当なファイル (例えば C のソースファイル) だったら、nneething はヘッダーを虚空からでっち上げます。これはファイルの所有者、名前および日付を使い、それらの要素を元にできることを何でもやります。

これはあなたにとってはすべて自動的に起こることで、あなたはニュースグループにとっても良く似た何かを見せられることになるでしょう。本当に寸分違わない、ニュースグループのようなものを。記事を選択すると、それはいつものように記事バッファーに表示されるでしょう。

ディレクトリーを表わしている行を選択すると、Gnus はいきなりあなたをこの nneething グループのための新しい概略バッファーに連れて行くでしょう。以下同様に、あなたがそうしたければ、この方法で全ディスクを駆け巡ることができます。ですが、Gnus は本当は dired ではないし、そのように意図されたものでもないことは覚えておいて下さい。

ここでの動作には二つの全体的なモードがあります—一時モードと固定モードです。一時的な操作を行なっているときは (すなわちグループバッファーで *G D*)、Gnus はどのファイルを読んだか、どのファイルが新しいか、などの情報を憶えておきません。普通に *G m* で固定 nneething グループを作れば、Gnus は記事番号とファイル名の対応表を憶えておくので、このグループを他のグループと同様に扱うことができるようになります。固定 nneething グループを活かすと、それが未読記事をいくつか含んでいるかを知らせてもらえる、等々の利便があります。

いくつかの変数があります:

nneething-map-file-directory

すべての固定 nneething グループの対応表が、このディレクトリーに格納されます。このデフォルトは '~/.nneething/' です。

nneething-exclude-files

この正規表現に合致するファイルはすべて無視されます。自動保存ファイルなどを除外するのに便利に使えます。そしてそれがまさにデフォルトで行なわれる動作です。

nneething-include-files

どのファイルをグループに含めるかを示す正規表現です。この変数が nil でなければ、この正規表現に合致するファイルだけが含まれます。

nneething-map-file

対応表ファイルの名前です。

6.6.3 文書グループ

nndoc は単一のファイルをニュースグループとして読むことをできるようにする、ちょっと気の利いたやつです。複数のファイルの種別がサポートされています:

babyl Babyl (Rmail) 形式のメールボックス。

mbox 標準 Unix mbox ファイル。

mmdf MMDF 形式のメールボックス。

news 一つのファイルにまとめられた複数のニュース記事。

rnews rnews のバッチ転送形式。

nsmail Netscape のメールボックス。

mime-parts

MIME のマルチパートのメッセージ。

standard-digest

標準 (RFC1153) のまとめ送り形式。

mime-digest

MIME のまとめ送りメッセージ。

lanl-gov-announce

ロスアラモス国立研究所 (LANL) Gov Announce からの発表メッセージ。

rfc822-forward

RFC822 で転送されたメッセージ。

outlook Outlook のメールボックス。

oe-dbx Outlook Express の dbx メールボックス。

exim-bounce

Exim MTA から跳ね返されたメッセージ。

forward 非公式の規則で転送されたメッセージ。

rfc934 RFC934 形式で転送されたメッセージ。

mailman mailman のまとめ送り。

clari-briefs

Clarinet のニュース項目を要約したまとめ送り。

slack-digest

非標準まとめ送り形式—だいたいものを扱えるが、下手。

mail-in-mail

最後の手段。

特別な「ファイル種別」である guess を使うこともできます。これを使うと、見ているファイルの種別が何かを nndoc が推測しようとします。また、digest というファイル種別は、そのファイルがどのまとめ送り形式かを nndoc に推測させます。

nndoc はファイルを書き換えようとしたり、余分なヘッダーを挿入しようとしたりはしません—単に、ファイルをそのグループを作る元として使えるようにする、というようなことです。それだけのことです。

保存された古い記事を持っていて、それを新しくてカッコいい Gnus のメールバックエンドに追加したいなら、おそらく nndoc が助けになるはずです。例えば新しい nnml グループに振り分けた古いメールが、今は古い 'RMAIL' ファイルにメールがあるとしましょう。そういう場合は、そのファイルを nndoc を使って開き (グループバッファで *G f* 命令 (see Section 2.9 [Foreign Groups], page 21) を使いましょう)、バッファ内の全記事にプロセス印を (例えば *M P b* で) 付けてから、それらが nnml グループ群に振り分けられるように (*B r* 命令を使って) 再スプールして下さい。すべてがうまくいけば、'RMAIL' ファイル内のすべてのメールは、たくさんの nnml ディレクトリーの中にも格納されます。そうしたら、あの厄介な 'RMAIL' を削除してしまっても良いでしょう。あなたにガッツがあれば!

仮想サーバー変数:

nndoc-article-type

これは mbox, babyl, digest, news, rnews, mmdf, forward, rfc934, rfc822-forward, mime-parts, standard-digest, slack-digest, clari-briefs, nsmail, outlook, oe-dbx, mailman および mail-in-mail または guess のいずれかでなくてはなりません。

nndoc-post-type

この変数は、そのグループをニュースグループとみなすかメールグループとみなすかを Gnus に伝えます。二つの有効な値は mail (デフォルト) および news です。

6.6.3.1 文書サーバーの内部

nndoc で認識される新しい文書の種別を追加することは難しくありません。その文書がどのように見えるかの定義を仕上げ、その文書種別を認識するための述語関数を書いて、nndoc を手なずけるだけで良いのです。

まず、これが文書の種別の定義の例です:

```
(mmdf
 (article-begin . "\^A\^A\^A\^A\n")
 (body-end . "\^A\^A\^A\^A\n"))
```

この定義は種別を示すためのユニークな名前 (*name*) と、それに続く仮想的な変数名およびその設定値の単純な連なりからなります。以下が使うことができる変数です—変数の数に圧倒されないで下さい。ほとんどの文書の種別は、ごくわずかな設定で定義することができます:

first-article

これが設定されていると、nndoc はこの正規表現に合致する何かが見つかるまで、すべてのテキストを読み飛ばします。それより前のすべてのテキストは完全に無視されます。

article-begin

この設定は、すべての文書の種別の定義に必ず存在しなければなりません。それぞれの記事の始まりがどのように見えるかを指定する正規表現です。単純な正規表現では対処できないもっと複雑なことをしたい場合は、これの代わりに article-begin-function を使うことができます。

article-begin-function

これを設定する場合は、それぞれの記事の開始位置にポイントを移動させる関数を指定して下さい。これは article-begin より優先されます。

head-begin

これを設定する場合は、記事のヘッダーの始まりに合致する正規表現を指定して下さい。単純な正規表現では対処できないもっと複雑なことをしたい場合は、この代わりに `head-begin-function` を使うことができます。

head-begin-function

これを設定する場合は、記事のヘッダーの開始位置にポイントを移動させる関数を指定して下さい。これは `head-begin` より優先されます。

head-end これを設定する場合は、記事のヘッダーの最後に合致する正規表現を指定して下さい。デフォルトは `‘^$’`、つまり空行です。

body-begin

これを設定する場合は、記事のボディーの始まりに合致する正規表現を指定して下さい。デフォルトは `‘^\n’` です。単純な正規表現では対処できないもっと複雑なことをしたい場合は、この代わりに `body-begin-function` を使うことができます。

body-begin-function

これを設定する場合は、記事のボディーの開始位置にポイントを移動させる関数を指定して下さい。これは `body-begin` より優先されます。

body-end これを設定する場合は、記事のボディーの最後に合致する正規表現を指定して下さい。単純な正規表現では対処できないもっと複雑なことをしたい場合は、この代わりに `body-end-function` を使うことができます。

body-end-function

これを設定する場合は、記事のボディーの最後の位置にポイントを移動させる関数を指定して下さい。これは `body-end` より優先されます。

file-begin

これを設定する場合は、ファイルの始まりに合致する正規表現を指定して下さい。それより前のすべてのテキストは完全に無視されます。

file-end これを設定する場合は、ファイルの最後に合致する正規表現を指定して下さい。それより後ろのすべてのテキストは完全に無視されます。

このように `nndoc` はこれらの変数を使って、文書ファイルをそれぞれヘッダーとボディーを持った記事の連なりとして切り分けることができます。しかし、すべての文書の種別がこのようなニュース風になっているわけではないので、さらにヘッダーやボディーを Gnus の趣味に合うように変形させる変数が、いくらか必要になります。

prepare-body-function

これに関数を設定しておく、記事が要求されたときに呼び出されます。これはボディーの開始位置のポイントを引数として呼び出され、文書にいくつかのエンコードされた内容物のパートがある場合に有用です。

article-transform-function

これに関数を設定しておく、記事が要求されたときに呼び出されます。これは記事のヘッダーとボディーの両方に、より広範囲な変形を行なうために使われるものです。

generate-head-function

これに関数を設定しておく、Gnus が理解できるヘッダーを生成するために呼び出されます。これは記事番号をパラメーターとして呼び出され、その記事のための良質なヘッダーを生成することを求められます。すべての記事のヘッダーが要求されるときに呼び出されます。

generate-article-function

これに関数を設定しておく、Gnus が理解できる完全な記事を生成するために呼び出されます。これはすべての記事のヘッダーが要求されるときに、記事番号をパラメーターとして呼び出されます。

dissection-function

これに関数を設定しておく、それだけを使って文書ファイルを記事に切り分けるために呼び出されます。これは first-article, article-begin, article-begin-function, head-begin, head-begin-function, head-end, body-begin, body-begin-function, body-end, body-end-function, file-begin および file-end より優先されます。

私が出会った中で最も複雑な例を見て下さい。標準まとめ送り形式のためのものです:

```
(standard-digest
 (first-article . ,(concat "^" (make-string 70 ?-) "\n\n+"))
 (article-begin . ,(concat "\n\n" (make-string 30 ?-) "\n\n+"))
 (prepare-body-function . nndoc-unquote-dashes)
 (body-end-function . nndoc-digest-body-end)
 (head-end . "^ ?$")
 (body-begin . "^ ?\n")
 (file-end . "^End of .*digest.*[0-9].*\n\\*\\*\\*\\|^End of.*Digest *$")
 (subtype digest guess))
```

70 文字のダッシュ (‘-’) の行より前はすべて無視されるというのが分かりますね。また ‘^End of’ で始まる行より後ろもすべて無視されます。各記事は 30 文字のダッシュの行で始まり、ヘッダーとボディーの区切りの行は一個のスペースを含むことがあり、そしてボディーはそれが渡される前に nndoc-unquote-dashes を通されます。

あなた独自の文書のための定義を nndoc で使えるようにするには、nndoc-add-type 関数を使って下さい。これは二つのパラメーターをとります——一つ目は定義そのもので、二つ目の (省略可能な) パラメーターは、この定義を文書の種別を定義する連想リストのどこに置くかを指定します。この連想リストは順番に走査され、与えられた種別 *type* に対して nndoc-type-type-p が呼び出されます。したがって、例えば mmdf という種別であるかどうかを調べるためには nndoc-mmdf-type-p が呼び出され、他の種別の場合も同様です。これらの種別述語関数は、文書がその種別でない場合は nil を返し、その種別である場合は t を返し、その種別かもしれないときは数値を返さなくてはなりません。高い数値は高い可能性を意味し、低い数値は低い可能性を意味します。‘0’ は正しい値の中でもっとも低い数値です。

6.6.4 SOUP

PC の世界の人々はよく、「オフライン」ニュースリーダーについて話をしています。それらはリーダーとニュース配送を合体させた、奇怪なものどもです。内蔵モデムプログラム付きでね。うげーっ!

もちろん、我らが Unix キチガイの人間ども (原典: Unix Weenie types of human beans) は uucp だとか nntpd のようなものを使い、神の領域たるメール、ニュースの配送を正しく設定するのです。そして...、僕たちはただ普通のニュースリーダーを使います。

しかし、あなたがとても遅いモデムを持っていてこれらを正しく設定することに興味が無ければ、ほんの少し脳みそに楽をさせる方をするのも時には便利でしょう。

SOUP というファイル形式は、ニュースとメールをサーバーから家のマシンへと転送し、それをまた戻すために開発されました。ちょっと面倒くさいかもしれませんがね。

はじめにいくつか用語を:

- server* これは外の世界に繋がっていて、あなたがニュースやメールを送受信するマシンです。
- home machine* これはあなたが実際に読んだり返事を書いたりしたいマシンです。これは普通、他の世界といかなる形でも接続されていません。
- packet* メッセージや命令を含んでいる何か。パケットには二種類あります。
- message packets*
 これはサーバーで作られるパケットで、普通はあなたが読むためのたくさんのメッセージを含んでいます。これらはデフォルトでは ‘SoupoutX.tgz’ という名前です。ここで *x* は数字です。
- response packets*
 このパケットは自宅マシンで作られるパケットで、普通はあなたが書いた返事を含んでいます。これらはデフォルトでは ‘SoupinX.tgz’ という名前です。ここで *x* は数字です。

1. まずサーバーにログインして SOUP パケットを作りましょう。SOUP 専用に供されたもの (awk プログラムのようなもの) を使っても良いし、Gnus の SOUP 命令を使ってパケットを作っても良いです (*Os* や *Gsb* に続いて *Gsp*) (see Section 6.6.4.1 [SOUP Commands], page 200)。
2. パケットを自宅に転送しましょう。電車で、船に、自動車に、あるいはモデムに、でも結構です。
3. パケットをあなたのホームディレクトリーに置きましょう。
4. 基本サーバーか二次サーバーとして nnsoup バックエンドを使うようにして、あなたの自宅のマシンで Gnus に火を入れましょう。
5. ニュース記事やメールを読んで、必要なものに返事やフォロー記事を書きましょう (see Section 6.6.4.3 [SOUP Replies], page 202)。
6. これらの返事を SOUP パケットにまとめるために、*Gsr* 命令を実行しましょう。
7. このパケットをサーバーに転送しましょう。
8. このパケットを Gnus の *Gss* 命令で送信しましょう。
9. あとはこれを死ぬまで繰り返しましょう。

というわけで、読むために nnsoup を使い、それらの SOUP パケットをまとめて送り出すためには Gnus を使うという分業システムが手に入りました。

6.6.4.1 SOUP 命令

これらは SOUP パケットを作成して操作するための命令です。

- Gsb* 現在のグループのすべての未読記事をパケットにまとめます (gnus-group-brew-soup)。このコマンドはプロセス/接頭引数の習慣に従います。
- Gsw* すべての SOUP データファイルを保存します (gnus-soup-save-areas)。
- Gss* 返信パケットからすべての返信を送出します (gnus-soup-send-replies)。
- Gsp* すべてのファイルを SOUP パケットにまとめます (gnus-soup-pack-packet)。
- Gsr* すべての返信を返信パケットにまとめます (nnsoup-pack-replies)。
- Os* この概略モード命令は、現在の記事を SOUP パケットに追加します (gnus-soup-add-article)。これはプロセス/接頭引数の習慣に従います (see Section 8.1 [Process/Prefix], page 249)。

Gnus がこれらすべてのものをどこに置くかをカスタマイズするための、いくつかの変数があります。

gnus-soup-directory

SOUP パケットの作成中に Gnus が中間ファイルを保存するディレクトリーです。ディフォルトは ‘~/SoupBrew/’ です。

gnus-soup-replies-directory

返信パケットの送信中に Gnus が使用する一時ディレクトリーです。‘~/SoupBrew/SoupReplies/’ がディフォルトです。

gnus-soup-prefix-file

Gnus が最後に使った接頭語を保存するファイル名です。ディフォルトは ‘gnus-prefix’ です。

gnus-soup-packer

SOUP パケットをまとめる命令を作るためのフォーマット文字列です。ディフォルトは ‘tar cf - %s | gzip > \$HOME/Soupout%d.tgz’ です。

gnus-soup-unpacker

SOUP パケットを取り出す命令を作るためのフォーマット文字列です。ディフォルトは ‘gunzip -c %s | tar xvf -’ です。

gnus-soup-packet-directory

Gnus が返信パケットを探す場所です。ディフォルトは ‘~/’ です。

gnus-soup-packet-regexp

gnus-soup-packet-directory にある SOUP パケットに合致する正規表現です。

6.6.4.2 SOUP グループ

nnsoup は SOUP パケットを読むためのバックエンドです。これは入ってきたパケットを読み込み、それを取り出し、あなたの都合が良いときにそこで読むディレクトリーに展開します。

これらはこの振る舞いをカスタマイズできる変数です:

nnsoup-tmp-directory

nnsoup が SOUP パケットを展開するとき、このディレクトリー内で行ないます (ディフォルトは ‘/tmp/’)。

nnsoup-directory

次に nnsoup は、それぞれのメッセージと索引ファイルをこのディレクトリーに移動させます。ディフォルトは ‘~/SOUP/’ です。

nnsoup-replies-directory

すべての返信は、返信パケットにまとめられる前にこのディレクトリーに格納されます。ディフォルトは ‘~/SOUP/replies/’ です。

nnsoup-replies-format-type

返信パケットの SOUP 形式です。ディフォルトは ‘?n’ (rnews) で、この変数には触るべきではないと私は思います。たぶん文書に明記すべきでさえなかったね。しまった! もう手遅れか。

nnsoup-replies-index-type

返信パケットの索引の種類です。ディフォルトは ‘?n’ で、意味は「なし」です。これもいじるんじゃないよ!

nnsoup-active-file

nnsoup がたくさんの情報を格納する場所です。これは nntp で言うところの「アクティブファイル」ではなく、Emacs Lisp のファイルです。このファイルを無くしてしまったり、何か壊してしまったら、あなたは死にます。デフォルトは ‘~/SOUP/active’ です。

nnsoup-packer

返信 SOUP パケットをまとめる命令を作るためのフォーマット文字列です。デフォルトは ‘tar cf - %s | gzip > \$HOME/Soupin%d.tgz’ です。

nnsoup-unpacker

入ってくる SOUP パケットを展開するための命令文字列の形式です。デフォルトは ‘gunzip -c %s | tar xvf -’ です。

nnsoup-packet-directory

入ってきたパケットを nnsoup が探す場所です。デフォルトは ‘~/’ です。

nnsoup-packet-regexp

入ってきた SOUP パケットに合致する正規表現です。デフォルトは ‘Soupout’ です。

nnsoup-always-save

nil 以外であれば、メッセージを一通投稿する度にその返信バッファを保存します。

6.6.4.3 SOUP 返信

単に nnsoup を使うだけで、自動的に送信したニュース記事やメールが SOUP 返信パケットに納まるわけではありません。そうするためには、もうちょっと働かなくてはなりません。

nnsoup-set-variables 命令は、あなたのすべてのフォローアップ記事と返信が SOUP システムの処理に渡るように、適切な変数を設定します。

具体的には、これが実際に行なわれることです。

```
(setq message-send-news-function 'nnsoup-request-post)
```

```
(setq message-send-mail-function 'nnsoup-request-mail)
```

本当にそれだけです。ニュース記事だけを SOUP システムに処理させたいければ、最初の行だけを使って下さい。メールだけを SOUP させたいなら、二番目を使って下さい。

6.6.5 メールからニュースへのゲートウェイ

あなたのローカルの nntp サーバーが何らかの理由で投稿を許可していなくても、数ある mail-to-news ゲートウェイを使って投稿することができます。nngateway バックエンドはこのインターフェースを提供します。

このバックエンドからは何も読み出せないことに注意して下さい—これは投稿するためだけに使われます。

以下はサーバー変数です。

nngateway-address

これが mail-to-news ゲートウェイのアドレスです。

nngateway-header-transformation

ニュースヘッダーは、mail-to-news ゲートウェイが受け付けられるように、何か奇妙なやり方で変形しておかなければならないことがしばしばです。この変数はどんな変形処理が呼び出されるべきかを指示するもので、デフォルトでは nngateway-simple-header-transformation になります。その関数は変形しようとするヘッダーの領域

だけに狭められたバッファで、ゲートウェイのアドレスを一つの引数として呼び出されます。

デフォルトの関数は、単に Newsgroups ヘッダーとゲートウェイのアドレスに基づいた新しい To ヘッダーを挿入します。例えば、以下のような Newsgroups ヘッダーを持つ記事には、

```
Newsgroups: alt.religion.emacs
```

次のような To ヘッダーが挿入されます。

```
To: alt-religion-emacs@GATEWAY
```

以下の関数が用意されています:

```
nngateway-simple-header-transformation
    newsgroup@nngateway-address のような To ヘッダーを作ります。

nngateway-mail2news-header-transformation
    nngateway-address のような To ヘッダーを作ります。
```

例です:

```
(setq gnus-post-method
      '(nngateway
        "mail2news@replay.com"
        (nngateway-header-transformation
         nngateway-mail2news-header-transformation)))
```

したがってこれを使うには、単にこんな風にすれば良いでしょう:

```
(setq gnus-post-method '(nngateway "GATEWAY.ADDRESS"))
```

6.7 合併グループ

Gnus は、すべてのグループの種類を混合して、大きなグループに合併させることができます。

6.7.1 仮想グループ

nnvirtual グループは、実は複数のグループを寄せ集めたものに過ぎません。

例えば、小さなグループをたくさん読むのが嫌になってきたら、それらを一つの大きなグループに入れて、嫌になるくらい巨大で手に負えないグループを読むことができます。これはコンピューティングの醍醐味だね!

選択方法として *nnvirtual* を指定して下さい。アドレスは、それを構成するグループに合致する正規表現です。

仮想グループで付けられたすべての印は、その構成要素のグループの記事にくっつけられます。つまり、仮想グループで記事に可視記事の印を付けると、その記事はもともとの構成要素のグループでも可視記事になります。(そして逆も成り立ちます—構成要素のグループで付けた印は、仮想グループでも表示されます。) 空の仮想グループを作るには、グループバッファで *G V* (*gnus-group-make-empty-virtual*) を実行し、*M-e* (*gnus-group-edit-group-method*) で選択方法の正規表現を編集して下さい。

これが、Andrea Dworkin に関するすべてのニュースグループを、一つの巨大でシアワセなニュースグループにまとめる *nnvirtual* 選択方法の例です:

```
(nnvirtual "^alt\\.fan\\.andrea-dworkin$\\|^rec\\.dworkin.*")
```

構成要素のグループは基本グループでも外部グループでも構いません。すべて問題無く動くはずですが、もしあなたのコンピューターが爆発でもしてしまったら、それはたぶん私が悪いんでしょうね。

利用者が (訳注: 記事を投稿する人たちが) Distribution ヘッダーを使って配布範囲を制限している場合に、同じグループを複数のサーバーから寄せ集めることは、本当にうまい考えかもしれません。‘soc.motss’ を日本のサーバーとノルウェーのサーバーの両方から読みたければ、グループの正規表現として以下のものを使うことができますでしょう:

```
"^nnntp\\+server\\.jp:soc\\.motss$\\|^nnntp\\+server\\.no:soc\\.motss$"
```

(でもちょっと注意。G m でグループを作成するときは、バックスラッシュを二重に付けてはいけません。そして文字列の最初と最後の引用記号 ("...") も取り払って下さい。)

これはまあ、すらすらと動作するはずですが—両方のグループのすべての記事は一つのグループに入り、重複も無いはずですが。スレッド表示 (とその他) も通常通り動作するでしょうけれど、記事の並び順序には問題があるかもしれません。日付による並べ替えが、ここでは一つの選択肢になるかもしれません (see Section 2.3 [Selecting a Group], page 17)。

なお、ここで一つだけ制限があります—仮想グループに含まれるグループはすべて生きている (すなわち購読または非購読の) 状態でなくてはなりません。削除された (killed) グループあるいはゾンビのグループは nnvirtual グループを構成するグループになることはできません。

nnvirtual-always-rescan 変数が nil でなければ (それ、つまり非-nil がデフォルト)、nnvirtual は仮想グループに入ったときに常に未読記事を走査します。この変数が nil になっていて、仮想グループを作った後に構成要素のグループで記事を読んだ場合は、その構成要素のグループで読まれた記事は、仮想グループに現れてしまうでしょう。共通な構成要素のグループを持つ二つの仮想グループがある場合にも、この影響があります。そういう場合には、この変数を t にするべきです。さもなければ、仮想グループに入る度に、毎回その仮想グループの上で M-g を叩いても良いでしょう—これにはほぼ同様の効果があります。

nnvirtual はメールとニュースの両方のグループを構成要素のグループにすることができます。nnvirtual グループの記事に返答するときは、nnvirtual は記事の出所の構成要素のグループのバックエンドに、それがニュースのバックエンドであるかメールのバックエンドであるかを尋ねなければなりません。しかし ^ をしたときには、普通は構成要素のバックエンドがこれを知るための確実な方法が無いので、その場合 nnvirtual は、Gnus に記事はニュースではないバックエンドからやって来たと告げます。(単にそれが安全な側なので。)

これらの場合にメッセージバッファで C-c C-n を行なうと、応答しようとしている記事から Newsgroups 行を抜き出して挿入します。

nnvirtual グループは、記事と印以外は構成要素のグループから継承しません—例えばグループパラメーターもそうなのですが、それらは継承されません。

6.7.2 Kiboze グループ

OED (オックスフォード英語大辞典) によれば、Kiboze する、とは、「ニュースプール全体 (あるいはその一部) を grep すること」と定義されています。nnkiboze はこれをあなたのために行なってくれるバックエンドです。嬉しいなあ! これでどんな NNTP サーバーでも、要りもしない検索で止まってしまうまで酷使することができるぞ。ああ、なんて幸せなんだ!

kiboze グループを作るには、グループバッファで G k 命令を使って下さい。

nnkiboze 方法におけるアドレス欄は、nnvirtual と同様に、nnkiboze に「含めたい」グループに合致する正規表現です。ここが nnkiboze と nnvirtual バックエンドの最も類似している点です。

構成要素のグループを列挙するこの正規表現に加えて、nnkiboze グループには、グループ内のどの記事を含めるかを定めるスコアファイルがなくてはなりません (see Chapter 7 [Scoring], page 227)。

あなたが欲しい nnkiboze グループを作成した後で、`M-x nnkiboze-generate-groups` を実行しなければなりません。この命令は時間がかかります。とってもかかります。すごく、すごーくかかります。Gnus はその nnkiboze グループの一部となるべき記事があるかどうかを調べるために、全部の構成要素のグループの全部の記事からヘッダーを取得し、それらすべてに対してスコア処理を実行しなくてはならないのです。

限定した正規表現を使って、構成要素のグループの数を抑えて下さい。さもないと、システム管理者はあなたに閉口してしまい、NNTP サーバーからあなたを追い出して二度と入れないようにしてしまうかもしれません。もっと変なことだって起こりました。

nnkiboze を構成するグループは生きている必要はありません—死んでいても良いし、外部グループでも構いません。無制限です。

nnkiboze グループを生成すると、二つのファイルが nnkiboze-directory に書き込まれます。そのデフォルトは `~/News/kiboze/` です。一方はそのグループのすべての記事の NOV ヘッダー行を含み、もう一方は構成要素の記事を見つけるためにどのグループが検索されたかの情報を格納する、`newsrsrc` の補助ファイルです。

既読になった nnkiboze グループの記事は、それらの記事の NOV 行が NOV ファイルから削除されます。

6.8 電子メールによる日程管理

この章では `nn diary` という特別なメール・バックエンドと、その仲間の `gnus-diary` ライブラリーについて説明します。それが「特別」なのは、Gnus でメールを読むための標準の選択肢の一つであるつもりは無いからです。それ (標準の選択肢) については Section 6.3.13 [Choosing a Mail Back End], page 168 を参照して下さい。代わりに、特別な方法であなたのメールのいくつかを扱う、すなわちこれはリマインダー (予定を思い出させるもの) として使われます。

典型的な筋書きは、こうです。

- あなたはアンディ・マクドウェルかブルース・ウィリス (あなたの好みに合わせて、どちらかを選んで下さい) と、一ヶ月後にデートの約束をしました。それを忘れるわけにはいきません。
- そこで、自分宛てにリマインダーのメッセージを (本当に毎日一通) 送ることにしました。
- あなたはそのことをすっかり忘れて、いつもどおりに新しいメールを取り込んで読み続けます。
- デートの日が近付いてくると、グループバッファで `g` をタイプしたときに、ときどきあなたの予定を思い出させるために、あたかも新着で未読のように、メッセージが再びポップアップするでしょう。
- これが含まれている「新しい」メッセージたちを読んで下さい、そして、再びあなたが過ごす夜を夢見て下さい。
- いったんデートが終わると (実際にはディナーのすぐ後で寝入ってしまったとしても)、期限切れ消去の印が付いていれば、メッセージは自動的に消去されます。

Gnus Diary バックエンドは、(常に取り消されることが無い) 定期的な予定を、几帳面な人たちと同じように扱う能力を持っていて、本当のメール・バックエンドのように動作し、いろんなやり方で設定することができます。このすべてが、以下の各章で説明されています。

6.8.1 NNDiary バックエンド

nndiary は nml (see Section 6.3.13.3 [Mail Spool], page 169) にとてもよく似ているバックエンドです。現にそれは nml と nndraft を合わせたものに見えるでしょう。nml をご存知ならば、あなたはすでに nndiary がメッセージを格納する仕組み（一通あたり一つのファイル、グループあたり一つのディレクトリー）に精通しています。

何はさておき、nndiary をちゃんと動作させるには、一つの要件があります：Gnus のグループの日付の機能を使わなければなりません。それがどういうふうに行なわれるかは Section 2.17.3 [Group Timestamp], page 41 を見て下さい。

6.8.1.1 日程メッセージ

七つの特別なヘッダーが必須であること以外、nndiary のメッセージはまったく普通のものです。これらのヘッダーは X-Diary-<something> の様式で表され、<something> の部分は Minute, Hour, Dom, Month, Year, Time-Zone および Dow のうちの一つです。Dom は「日 (Day of Month)」を、Dow は「曜日 (Day of Week)」を意味します。これらのヘッダーは crontab の設定のように働いて、予定日を定義します。

- Time-Zone のもの以外のすべてのヘッダーについて、ヘッダーの値は星印（可能なすべての値を意味します）かコンマで区切られたフィールドのリストです。
- フィールドは整数が範囲のどちらかです。
- 範囲とは、ダッシュ (-) で区切られた二つの整数です。
- 可能な値は、それぞれ Minute には 0-59、Hour には 0-23、Dom には 1-31、Month には 1-12、Year には 1971 より大きい値、そして Dow には 0-6 (0 が日曜日) です。
- 特別な場合として、Dom または Dow のどちらか一方における星印は「可能なすべての値」ではなく、「もう一方のフィールドだけを使う」意味になります。両方とも星印にした場合は、どちらを使っても同じ結果になることに注意して下さい。
- Time-Zone ヘッダーは、値を一つしか持てない（例えば GMT）点で特別です。星印は「可能なすべての値」ではなく（それは意味をなさないので）、「現在のローカルなタイムゾーン」を意味します。ここではたいてい星印を使うでしょう。しかし、利用できるタイムゾーンの値については、変数 nndiary-headers を見て下さい。

1999 年から 2010 年までの毎週月曜日と毎月の一日の 12:00, 20:00, 21:00, 22:00, 23:00 および 24:00 を設定するために、メッセージに加える日程ヘッダーの具体例です（その時何をしたら良いかは、自分で考えて下さい）:

```
X-Diary-Minute: 0
X-Diary-Hour: 12, 20-24
X-Diary-Dom: 1
X-Diary-Month: *
X-Diary-Year: 1999-2010
X-Diary-Dow: 1
X-Diary-Time-Zone: *
```

6.8.1.2 NNDiary を動かす

nndiary には二つの動作モードがあります。一つはデフォルトの「伝統型 (traditional)」も、もう一つは「自律型 (autonomous)」です。伝統型のモードでは、nndiary はそれ自身が新着メールを取得することはありません。日程メッセージとして扱うために、あなたはメールを基本のメール・バックエンドから nndiary グループに、移動 (B m) またはコピー (B c) しなければなりません。自

律型のモードでは、nndiary はそれ自身のメールを取ってきて、基本のメール・バックエンドとは独立してそれを扱います。

本質的に Gnus は、同時に複数の「マスター」メール・バックエンドを許容するようには設計されていなことに注意すべきです。しかし nndiary では、これは意味をなします。あなたは本当に、日程メッセージを日程グループに直接送って、それらを受け取りたいのです。そこで nndiary は、まさに「二番目の第一メール・バックエンド」をサポートします（私が知っている限り、それはこの機能を提供する唯一のバックエンドです）。しかしながら制約があって（いつの日にか解消することを願いますが）、自律型のモードでは再スプールができません。

自律型のモードで nndiary を使うためには、いくつかのことをやらなければならない:

- 新着メールを nndiary が自分で取り込めるようにします。以下の行を ‘~/gnus.el’ ファイルに記入して下さい:

```
(setq nndiary-get-new-mail t)
```

- 日程メッセージ (X-Diary-* ヘッダーを含んでいる) が、Gnus がそれら进行处理する 前 に専用のフォルダーに分配されるように、準備を行わなければならない。繰り返しますが、Gnus が複数の第一メール・バックエンドを適切に扱うことが (まだ ?) できないので、これが必要です。別々のソースからそれらのメッセージを取り込むことによって、この欠点はある程度補われます。

日程ファイルを ‘~/nndiary’ (これがデフォルトの nndiary のメールソース・ファイルです) に格納するための procmailrc の項目の例です:

```
:0 HD :
* ^X-Diary
.nndiary
```

いったんこれを実施したら、日程メールの取り込みと分割の処理に影響する、以下の二つのオプションをカスタマイズする必要があるでしょう:

nndiary-mail-sources [Variable]

標準の mail-sources 変数の、日程用に特化した代替品です。同じ構文 (syntax) を使い、デフォルトは (file :path "~/nndiary") です。

nndiary-split-methods [Variable]

標準の nnmail-split-methods 変数の、日程用に特化した代替品です。同じ構文 (syntax) を使います。

最終的には gnus-secondary-select-methods に、恒久的な nndiary 仮想サーバー ((nndiary "diary")) が行なうべきであるようなものを追加しても良いでしょう。

うまくいけば、Gnus を再起動すると、ほとんどすべて (‘nndiary.el’ の TODO の項を参照) が期待通りに動作するでしょう。自律型のモードでは、*g* や *M-g* をグループバッファでタイプすれば新しい日程メールをも取り込んで、日程用に特化した規則に従ってそれらを分割するし、*F* は新しい日程グループを見つけてくれる、など。

6.8.1.3 NNDiary のカスタマイズ

さあ nndiary が立ち上がって動作しています。それをカスタマイズする 때가 来ました。カスタマイズするためのグループは nndiary です (へえー)。どのオプションをカスタマイズし倒したいかを見つけるために、それに目を通して下さい。あなたが変更したいのは、おそらく以下のたった二つの変数でしょう:

`nndiary-reminders` [Variable]
 予定を思い出させてもらいたい時刻のリスト (例えば三週間前、それから二日前、それから一時間前、そしてそのとき) です。「思い出させてもらう」の意味は、新着メールを取り込んだときに、日程メッセージが真新しく未読になって、ポップアップすることであることを思い出して下さい。

`nndiary-week-starts-on-monday` [Variable]
 読んで字の如し。さもなくば日曜日が仮定されます (それがデフォルトです)。

6.8.2 Gnus Diary ライブラリー

`nndiary` を手作業で使うこと (ヘッダーを手で書くことなど) は、いささかうんざりします。幸い `nndiary` の上位階層に書かれた `gnus-diary` というライブラリーがあって、たくさんの便利なことをやってくれます。

それを使うためには、以下の行を ‘`~/.gnus.el`’ ファイルに加えて下さい:

```
(require 'gnus-diary)
```

さらに、どんな `gnus-user-format-function-[d|D]` (see Section 3.1.1 [Summary Buffer Lines], page 43) も、使ってははいけません。`gnus-diary` はそれらの両方を提供します (あなたがそれらを使っていたら、すみません)。

6.8.2.1 日程の概略行仕様

標準の概略行仕様 (通常 ‘From Joe: Subject’ のようなもの) で日程メッセージを表示するのは、まったく役に立ちません。たいていはあなたがメッセージを書いた人で、おおかた予定の日付を見たいと思っているでしょう。

`gnus-diary` は、概略行仕様で使う二つの追加の利用者定義の書法仕様を提供します。D は次の予定が生じるための整形された時刻表示 (例えば “Sat, Sep 22 01, 12:00”) を表すのに対して、d は次の予定が生じるまでのおおよその残り時間 (例えば “in 6 months, 1 week”) を表します。

ジョーの誕生日が、概略行にどう表示されるかの例です (定期的な予定を指定すると消されないことを除いて、メッセージが期限切れ消去可能であることに気を付けて下さい):

```
E Sat, Sep 22 01, 12:00: Joe's birthday (in 6 months, 1 week)
```

上記のようなものを得るために、普段だったら、あなたは以下の行を日程グループのパラメーターに加えようとするでしょう:

```
(gnus-summary-line-format "%U%R%z %uD: %(%s%) (%ud)\n")
```

しかし `gnus-diary` はそれを自動で行ないます (see Section 6.8.2.4 [Diary Group Parameters], page 209)。それでもあなたは、以下のユーザー・オプション群で提供される概略行仕様を、カスタマイズすることができます:

`gnus-diary-summary-line-format` [Variable]
 日程グループのために使われる概略行仕様を定義します (see Section 3.1.1 [Summary Buffer Lines], page 43)。`gnus-diary` はそれを、日程グループのパラメーターを自動で更新するために使います。

`gnus-diary-time-format` [Variable]
 日程の概略バッファに日付を表示するための書法仕様を定義します。これは利用者定義の書法仕様 D で使われます。詳細は変数の説明文を見て下さい。

`gnus-diary-delay-format-function` [Variable]

日程の概略バッファに遅延 (残り時間) を表示するための整形関数を定義します。これは利用者定義の書法仕様 `d` で使われます。現在は英語とフランス語のための組み込み関数があり、自分で定義することもできます。詳細は変数の説明文を見て下さい。

6.8.2.2 日程記事の並べ替え

`gnus-diary` は並べ替え (see Section 3.10 [Sorting the Summary Buffer], page 69) のために `gnus-summary-sort-by-schedule`、`gnus-thread-sort-by-schedule` および `gnus-article-sort-by-schedule` という新しい関数を提供します。これらの関数によって、最も近い予定から最も遠い方まで、日程の概略バッファを整理することができます。

`gnus-diary` は自動的に概略バッファの「並べ替え (sort)」メニューに `gnus-summary-sort-by-schedule` を組み込み、他の二つを第一次の (ゆえにデフォルトの) 並べ替え関数として、グループパラメーター (see Section 6.8.2.4 [Diary Group Parameters], page 209) に登録します。

6.8.2.3 日程ヘッダーの生成

`gnus-diary` は、`X-Diary-*` ヘッダーの取り扱いを補佐するために、`gnus-diary-check-message` という関数を提供します。この関数は、現在のメッセージがすべての必要な日程ヘッダーを確実に含むようにして、必要ならば値を入力するか修正することを要求します。

記事を日程グループに移動またはコピーすることによって自動的にそれが発動されるようにするために、この関数は `nndiary` バックエンドのフックとして組み入れられています。それはさらに、通常のメールを日程用のものに変換する操作を簡単にするために、`message-mode` と `article-edit-mode` において `C-c D c` キーとして設定もされています。

接頭引数を伴ってこの関数を呼ぶと、それらがあるか、正しいかどうかとは無関係に、日程ヘッダーの入力を強制します。そうやって、例えばすでに正しく設定されたメッセージの日程を、とても簡単に変更することができます。

6.8.2.4 日程グループのパラメーター

新しい日程グループを作るか、またはそれを開くと、`gnus-diary` は自動的にグループパラメーターを検査し、必要なら概略行仕様を日程用に特化した値に設定し、日程用の並べ替え関数を組み込み、さらにそのグループの投稿様式 (`posting-style`) に種々の `X-Diary-*` ヘッダーを加えます。そして、日程メッセージを送るのは、もっと簡単です。メッセージを用意するために、日程グループで `C-u a` か `C-u m` を使うことによって、これらのヘッダーが自動的に挿入されるので (まだ適切な値で満たされていませんが)。

6.8.3 送信するべきか、しないべきか

さて、以上の説明を読んできたものとして、以下は `nndiary` でメールを送信することに関する、二つの最後の注意事項です:

- `nndiary` は 本当の メール・バックエンドです。本当にあなたは本当の日程メッセージを本当に送ります。これは、日程メッセージを送ることによって、誰にでも (彼らが `Gnus` と `nndiary` を使っているのなら) 予定を伝えることができることをも意味します。
- しかしながら `nndiary` は `request-post` メソッドを持ってもいるので、日程グループで `C-u m` の代わりに `C-u a` を使うことによって、メッセージを実際に送信するのではなく、そのグループにローカルに格納することもできます。これは個人的な予定のためには、とても役に立ちます。

6.9 Gnus の切り離し

いにしえの時代 (およそ 1988 年 2 月頃)、人々はニュースリーダーをネットワークに常時接続した大きなマシンで走らせていました。ニュースの配送はニュースサーバーによって取り扱われ、すべてのニュースリーダーがすべきことはニュースを読むことであつたのです。信じられないかもしれませんが。

今日では多くの人々は自宅でニュースやメールを読み、ネットワークに接続するためにモデムの類を使います。電話代の請求書が莫大なものに上らないように、すべてのニュースとメールをすすり込んで電話を切り、数時間かけて読んでから送りたい返信をすべて送信する、という手段を持つことは良いことでしょう。あとはこの手順を繰り返すのです。(訳注: この章の前身は 1997 年頃に書かれました。)

もちろん、これを行なうためにニュースサーバーを使うこともできます。私は inn を slurp, pop, sendmail と一緒にここ数年使ってきましたが、しかしこれは退屈な仕事です。もしあるマシン上でニュースを読む人があなたしかいなければ、ニュースサーバーの機能をニュースリーダーに任せるようにすることは理にかなっています。

Gnus を「オフライン」のニュースリーダーとして仕立てるのは極めて簡単です。実際、エージェントは今やデフォルトで有効になっている (see Section 6.9.11 [Agent Variables], page 221) ので、あなたは何も設定する必要が無いのです。

もちろん、これをそんなふうには使うには、いくつか新しい命令を覚えなくてはなりません。

6.9.1 エージェントの基礎

まず、いくつかの用語を片付けておきましょう。

ネットワークとの接続を切っているとき (かつエージェントにそのことを知らせてあるとき)、Gnus エージェントは *unplugged* です、と言います。ネットワークとの接続が復活したら (かつ Gnus がそのことを知っていれば)、エージェントは *plugged* です。

「ローカル」マシンとは、あなたがそこで作業しているもので、継続的にネットワークに接続されているわけではありません。

「ダウンロード」とは、あなたのローカルマシンに、何かをネットワークから取ってくることを意味します。「アップロード」はその逆をすることです。

ご存知のように Gnus はあなたがドジを踏むすべての機会を提供します。それを柔軟性と言う人もいます。さらに Gnus は大いにカスタマイズ可能で、それは利用者が、Gnus がどのように動作するかについて発言権を持っていることを意味します。他のニュースリーダーは有無を言わずあなたにドジを踏ませるかもしれませんが、Gnus ではあなたに選択権があります!

Gnus は実際には *plugged* または *unplugged* のどちらの状態にもありません。もっと正確に言えば、サーバーごとにそれぞれの状態を持ちます。これは、いくつかのサーバーが *unplugged* でも、他のサーバーは *plugged* になることができるということです。さらに、エージェントがいくつかのサーバーをまとめて無視する (それらを常に *plugged* になっているように見せかける) ようにもできます。

さて、エージェントを *unplugged* にしたのに Gnus がネットに接続しているのを疑問に思ったら、行なうべき次のステップはサーバーがすべてエージェント化されているかどうかを確かめることです。エージェント化されていないサーバーがあったら、あなたは犯人を見つけたのです。

もう一つは「オフライン」という状態です。サーバーはときどき接続できなくなります。Gnus がこのことに気付くと、そのサーバーをオフラインの状態に切り換えても良いかどうかを尋ねます。Yes と答えたならば (オンラインに戻して良いかと Gnus が尋ねた場合以外は)、サーバーはいくらも *unplugged* だったときのように振る舞います。

エージェントを使った典型的な Gnus の対話操作を見てみましょう:

- Gnus を `gnus-unplugged` で起動します。これは `unplugged` で Gnus エージェントを立ち上げます。このモードでは、すでに取得しているニュース記事はすべて読むことができます。
- 次に、新しいニュースが到着しているかどうかを調べることにします。マシンをネットワークに (PPP か何かを使って) 接続してから Gnus を *plugged* にするために `Jj` を叩き、いつものように新着メールを検査するために `g` を使います。Gnus エージェントが `unplugged` になっているときに新着メールを検査するには、Section 6.3.4.1 [Mail Source Specifiers], page 148 を参照して下さい。
- そうすれば、直ちに新しいニュースを読むこともできるし、ニュースをローカルマシンにダウンロードすることもできます。後者を実行したいときは、`g` を押して新しいニュースがあるかどうかを検査し、次に `Js` ですべてのグループのすべての適格な (訳注: あなたが指定した条件に合致する) 記事を取得します。(どの記事をダウンロードしたいかを Gnus に指示するには Section 6.9.2 [Agent Categories], page 211 を参照して下さい。)
- 記事を取得した後で `Jj` を押し、Gnus を再び `unplugged` にして、PPP の接続 (か何か) を閉じます。その後でニュースをオフラインで読みます。
- そして第二ステップに戻ります。

エージェントを初めて使うときは (またはそのくらいの時期に)、以下のいくつかの作業をしなければなりません。

- どのサーバーをエージェントで面倒を見るかを決めます。メールのバックエンドをエージェントに面倒を見させるのはおそらく無意味でしょう。サーバーバッファーに移動し (グループバッファーで ^)、エージェントに扱って欲しいサーバー (複数可) で `Ja` を押す (see Section 6.9.3.3 [Server Agent Commands], page 218) か、またはエージェントに扱って欲しくないのに自動的に追加されたサーバーで `Jr` を押します。デフォルトでは `gnus-select-method` と `gnus-secondary-select-methods` にあるすべての `nntp` と `nnimap` サーバーがエージェント化されます。
- ダウンロード方針を決定します。あなたの方針を実装するために、エージェント分類、トピックパラメーター、グループパラメーターのどれを使うかをいったん決めてしまえば、これはかなり簡単です。あなたが Gnus の初心者ならば、たぶん分類で始めるのが最良でしょう、See Section 6.9.2 [Agent Categories], page 211.
トピックパラメーター (see Section 2.16.5 [Topic Parameters], page 37) とエージェント分類 (see Section 6.9.2 [Agent Categories], page 211) の両方とも、多数のグループに適用する方針の設定を用意しています。どれを使うかは完全にあなたの責任です。両方を混ぜて使う場合は、トピックパラメーターは分類を無効にすることを考慮に入れなければならないでしょう。あなたの方針にそぐわない少数のグループがあるのなら、それらの設定を変更するためにグループパラメーター (see Section 2.10 [Group Parameters], page 23) を使うことができます。
- ええと...、以上です。

6.9.2 エージェント分類

ニュースを配送する機構をニュースリーダーに統合する主要な理由の一つは、どの記事をダウンロードするかについて、もっと強力に制御できるようにすることです。莫大な量の記事をダウンロードすることにあまり意味はなく、それらを読んでもあまり面白くないことが分かるだけです。何をダウンロードするかを選択ではもう少し保守的になって、その記事がやっぱり面白そうだとわかったら、主動でダウンロードするための印を付ける方がすぐれています。

何をダウンロードするかを制御するためのより有効な方法の一つは、分類 (*category*) を作成して、その分類にいくつか (または全部) のグループを割り当てることです。どんな分類にも属さないグ

グループは「デフォルト」の分類に属します。Gnus は分類の作成と管理のための独自のバッファを持っています。

もしそうしたければ、グループパラメーター (see Section 2.10 [Group Parameters], page 23) とトピックパラメーター (see Section 2.16.5 [Topic Parameters], page 37) を、エージェントを制御する代替手段に使うことができます。実際に違うのは、グループとトピックパラメーターが何でもかんでも (kitchen sink) 含むのに対して、分類はエージェントに特化している (したがってあまり学ばなくても良い) という点だけです。

エージェントパラメーターは複数の違う場所で設定することができるので、どのソースが信用できるかを定めるための規則を設けました。この規則は、パラメーターのソースが次の順序で調べられることを定めます: グループパラメーター、トピックパラメーター、エージェント分類、そして最後はカスタマイズできる変数群です。したがって、広い範囲で動作を起こさせるためにこれらのソースをすべて混合することができます。どこに設定を置いたのかを忘れてしまったからといって、私を責めないで下さいよ。

6.9.2.1 分類の文法

分類は、名前、その分類に属するグループのリスト、およびカスタマイズ可能な変数よりも優先される多くの任意なパラメーターから成ります。エージェントパラメーターの完全なリストを以下に示します。

`agent-groups`

この分類にあるグループのリスト。

`agent-predicate`

(通常) どの記事をダウンロードするのが適当かという大まかな輪郭を与える述語。そして

`agent-score`

(通常) どの記事をダウンロードするかを決めるときのよりきめの細かいスコア規則。(このダウンロードスコア (*download score*) は通常のスコアとは必ずしも関係が無いことに注意して下さい。)

`agent-enable-expiration`

このグループの古い記事をエージェントが期限切れ消去すべきかどうかを示すブール変数。大抵のグループはディスク空間を浪費しないために期限切れ消去されるべきです。いや、実際には `gnus.*` 階層は期限切れ消去されるべきではないグループだけを含んでいると言っても、たぶん差し支えありません。

`agent-days-until-old`

既読の記事を期限切れ消去しても差し支えないことを判断する前に、エージェントが待っているべき日数を示す整数。

`agent-low-score`

`gnus-agent-low-score` よりも優先される整数。

`agent-high-score`

`gnus-agent-high-score` よりも優先される整数。

`agent-short-article`

`gnus-agent-short-article` よりも優先される整数。

`agent-long-article`

`gnus-agent-long-article` よりも優先される整数。

agent-enable-undownloaded-faces

ダウンロードされていない記事を gnus-summary-*-undownloaded-face のフェイス群を使って概略バッファーに表示すべきかどうかを示すシンボル。nil 以外ならどんなシンボルでも、ダウンロードされていない記事用のフェイスを使うようになります。

いったん分類が作られたら、分類の名前を変えることはできません。

それぞれの分類は、その分類の排他的な（他の分類には無い）メンバーであるグループのリストを維持します。排他規則は自動的に執行され、新しい分類にグループを追加すると、それは古い分類から自動的に取り除かれます。

述語の一番単純な形式は true や false のような単独の述語からなります。これらの二つはそれぞれ、すべての可能な記事をダウンロードするか、まったく何もしないか、です。これらの二つの特別な述語の場合は、追加のスコア規則は不要です。

high や low という述語は下で説明されているように、gnus-agent-high-score と gnus-agent-low-score との記事のスコアとの関係により記事をダウンロードします。

何をもってダウンロードすることが適格だと見なされるかについて、さらに細かい制御を得るために、述語は論理演算子が間に散りばめられた述語の組み合わせからなることができます。

おそらくいくつかの例が必要でしょう。

以下は簡単な述語です。（これはデフォルトの述語です。実際に他のどの分類にも含まれないすべてのグループに対して使用されます。）

short

とっても簡単でしょ？この述語は、記事が短い（「短い」ことを意味する何らかの価値がある）場合に限り真になります。

これはもっと複雑な述語です：

```
(or high
  (and
    (not low)
    (not long)))
```

この意味は、高いスコアを持っているか、あるいはスコアが低くなくてかつ長くない、という記事をダウンロードする、ということです。様子はわかりましたね。

使ってもよい論理演算子は or, and および not です。（もし使いたければ、より“C”風の演算子 ‘|’, ‘&’, ‘!’ を代りに使うことができます。）

以下の述語があらかじめ定義されていますが、これらのどれもあなたのやりたいことに適さなければ、自分で独自のものを書くこともできます。

それぞれのこれらの述語を評価するとき、名前が付けられた定数は、適切なパラメーターを与えて gnus-agent-find-parameter を呼ぶことによって決定される値に束縛されます。例えば gnus-agent-short-article は (gnus-agent-find-parameter group 'agent-short-article) に束縛されます。これは、あなたの分類で述語を指定してから、その述語を個々のグループについて調整できることを意味します。

short 記事が gnus-agent-short-article の行数より短い場合だけ真です。デフォルトは 100 です。

long 記事が gnus-agent-long-article の行数より長い場合だけ真です。デフォルトは 200 です。

low	記事のダウンロードスコアが <code>gnus-agent-low-score</code> の値より小さい場合だけ真です。デフォルトは 0 です。
high	記事のダウンロードスコアが <code>gnus-agent-high-score</code> の値より大きい場合だけ真です。デフォルトは 0 です。
spam	Gnus エージェントがその記事を spam だと推測した場合だけ真です。この検出法は今後変更されるかもしれませんが、現時点では、これはチェックサムを計算し、記事が一致するかどうかを調べているだけです。
true	常に真です。
false	常に偽です。

独自の述語関数を作成したければ、このことを覚えておかねばなりません: 関数は引数無しで呼び出されますが、`gnus-headers` と `gnus-score` という動的な変数が有意な値に束縛されるということを。

例えば、一定の日数以上前に投稿された記事 (例えば `gnus-agent-expire-days` の日数以上前に投稿されたもの) をダウンロードしないと決断することもできます。その場合、以下のような関数を書いて、

```
(defun my-article-old-p ()
  "Say whether an article is old."
  (< (time-to-days (date-to-time (mail-header-date gnus-headers)))
    (- (time-to-days (current-time)) gnus-agent-expire-days)))
```

そして述語はこのように定義すれば良いでしょう:

```
(not my-article-old-p)
```

もしくは `~/gnus.el` が何かで、あらかじめ定義されている `gnus-category-predicate-list` の値に、自分の述語を追加することもできます。

```
(require 'gnus-agent)
(setq gnus-category-predicate-alist
      (append gnus-category-predicate-alist
              '((old . my-article-old-p))))
```

この場合は、次のように述語を指定するだけです:

```
(not old)
```

上のようなものを使うときは、世の中には正しく設定されていないシステム/メーラーがあり、記事の日付はいつ投稿されたかを常に確実に示すわけではないことを知っていて下さい。困ったことに、それを少しも気にかけない人もいます。

上記の述語はその分類に属する すべて のグループに適用されます。しかし、分類中の個々のグループのための特定の述語を設定したかったり、単に不精を決め込んで新しい分類を設定したくないのならば、グループの個々の述語を次のようにグループパラメーターに入れることができます:

```
(agent-predicate . short)
```

これは `agent` 分類のデフォルトと等価なグループ/トピックパラメーターです。このように単一の語で述語を指定するときは、`agent-predicate` の設定値はドット対で表記しなければならないことに注意して下さい。

上のものと等価な長い方の例はこうなるでしょう:


```
(agent-predicate or high (and (not low) (not long)))
```

述語の値がドット対で表記されていないくて、その値はリストだと仮定されるので、分類の設定で要求される外側の括弧が、ここでは入れられません。

さて、ダウンロードスコアの文法は通常のスコアファイルの文法と同じですが、例外があります。記事そのものを実際に調べる必要がある要素は厳禁です。つまり、以下のヘッダーだけがスコア付けできるということです: Subject, From, Date, Message-ID, References, Chars, Lines および Xref。

述語の場合のように、ダウンロードスコア規則 の設定は、それをグループに関して使う限りは、そのすべてのグループに適用できるものならば分類の定義、グループに特有ならばグループパラメーター、のどちらかにできます。

これら両方の場所で、ダウンロードスコア規則 は以下の三つの形式の一つを取ることができます:

1. スコア規則

上で書かれているように、スコア付けキーワードの一部分しか使えないことを除けば、これは普通の Gnus スコアファイルの構文と同じです。

例:

- 分類指定

```
((("from"
    ("Lars Ingebrigtsen" 1000000 nil s))
  ("lines"
    (500 -100 nil <))))
```

- グループ/トピックパラメーター指定

```
(agent-score ("from"
              ("Lars Ingebrigtsen" 1000000 nil s))
              ("lines"
                (500 -100 nil <))))
```

ここでも一番外側の括弧が省略されていることに注意して下さい。

2. エージェントスコアファイル

これらのスコアファイルは、上で述べられている使用可能なスコア付けキーワード だけ を含んでいなければなりません。

例:

- 分類指定

```
("~/News/agent.SCORE")
```

または、もしかすると

```
("~/News/agent.SCORE" "~/News/agent.group.SCORE")
```

- グループパラメーター指定

```
(agent-score "~/News/agent.SCORE")
```

ここでも前述のように、追加のスコアファイルを指定することができます。括弧について言わなければいけませんか？

3. 普通 のスコアファイルの使用

あるグループのためにあなたが望んだ「ダウンロード」の基準が、「読む」基準と同じならば、一つのグループのために二つのスコア規則を維持管理したいとは思わないでしょう。そういう場合

は、何をダウンロードするかを決める際に、エージェントに 普通 のスコアファイルを参照させることができます。

分類の定義やグループパラメーターでこれらの指示を行なうと、エージェントはあるグループに適用することができるすべてのスコアファイルを読み込んで、使うことが許されているスコア付けキーワードの副セットではない項目を 選別して取り除きます。

- 分類指定
file
- グループパラメーター指定
(agent-score . file)

6.9.2.2 分類バッファ

通常すべての分類は分類バッファから管理します。これに (グループバッファで `J c` 命令を使って) 初めて入ると、デフォルトの分類だけが表示されます。

このバッファでは以下の命令を使うことができます:

- `q` グループバッファに戻ります (`gnus-category-exit`)。
- `e` 選択された分類のパラメーターを一括して設定するために、カスタマイズバッファを使います (`gnus-category-customize-category`)。
- `k` 現在の分類を消去します (`gnus-category-kill`)。
- `c` 現在の分類を複製します (`gnus-category-copy`)。
- `a` 新しい分類を追加します (`gnus-category-add`)。
- `p` 現在の分類の述語を編集します (`gnus-category-edit-predicate`)。
- `g` 現在の分類に属するグループのリストを編集します (`gnus-category-edit-groups`)。
- `s` 現在の分類のダウンロードスコア規則を編集します (`gnus-category-edit-score`)。
- `l` すべての分類を表示します (`gnus-category-list`)。

6.9.2.3 分類変数

`gnus-category-mode-hook`
分類バッファで実行するフックです。

`gnus-category-line-format`
分類バッファの行様式です (see Section 8.4 [Formatting Variables], page 250)。
有効な要素は:

- ‘c’ 分類の名前です。
- ‘g’ その分類に属するグループの数です。

`gnus-category-mode-line-format`
分類モード行の様式です (see Section 8.4.2 [Mode Line Formatting], page 251)。

`gnus-agent-short-article`
この値より少ない行数の記事は短いと見なします。デフォルトは 100 です。

`gnus-agent-long-article`
この値より多い行数の記事は長くと見なします。デフォルトは 200 です。

gnus-agent-low-score

この値より小さいスコアを持つ記事は低スコアだと見なします。デフォルトは 0 です。

gnus-agent-high-score

この値より大きいスコアを持つ記事は高スコアだと見なします。デフォルトは 0 です。

gnus-agent-expire-days

期限切れ消去する前に、既読記事をエージェントのローカルディスクに留めておかなければならない日数 (「期限切れ消去」という名前は同じですが、サーバーで期限切れ消去することではありません。単に記事のローカルな複製を消すことを意味します)。さらに理解すべき大事なことは、記事が読まれた時ではなくローカルディスクに記事が書かれた時から計数が始まるということです。デフォルトは 7 日です。

gnus-agent-enable-expiration

グループの記事が、デフォルトで期限切れ消去されるか、無期限に保持されるかを決定します。デフォルトは ENABLE で、あなたが望むならば期限切れ消去をさせないようにしなければならないことを意味します。一方、これを DISABLE に設定することができます。その場合、選択されたグループでの期限切れ消去を有効にしなければなりません。

6.9.3 エージェント命令

すべての Gnus エージェント命令は *J* サブマップにあります。*J j* (`gnus-agent-toggle-plugged`) 命令はすべてのモードで動作し、Gnus エージェントの `plugged/unplugged` 状態を切り替えます。

6.9.3.1 グループエージェント命令

- J u* 現在のグループの適格な (訳注: あなたが指定した条件に合致する) 記事をすべて取得します (`gnus-agent-fetch-groups`)。
- J c* エージェント分類バッファに入ります (`gnus-enter-category-buffer`)。
- J s* 全グループの適格な (訳注: あなたが指定した条件に合致する) 記事をすべて取得します (`gnus-agent-fetch-session`)。
- J S* 順番待ち (queue) グループにある送信可能なメッセージをすべて送信します (`gnus-group-send-queue`)。See Section 5.7 [Drafts], page 130.
- J a* 現在のグループをエージェント分類に追加します (`gnus-agent-add-group`)。この命令はプロセス/接頭引数の習慣を理解します (see Section 8.1 [Process/Prefix], page 249)。
- J r* 現在のグループを、もし存在していれば、その分類から消去します (`gnus-agent-remove-group`)。この命令はプロセス/接頭引数の習慣を理解します。(see Section 8.1 [Process/Prefix], page 249)。
- J Y* リモートサーバーが `unplugged` のときに変更されたフラグがあれば同期させます。

6.9.3.2 概略エージェント命令

- J #* 記事にダウンロード印を付けます (`gnus-agent-mark-article`)。
- J M-#* 記事からダウンロード印を消去します (`gnus-agent-unmark-article`)。

- @** 記事をダウンロードするかどうかを切り替えます (`gnus-agent-toggle-mark`)。デフォルトではダウンロードの印は ‘%’ です。
- J c** キャッシュされていない、ダウンロードされていない、またはダウンロードできないすべての記事を既読にします (`gnus-agent-catchup`)。
- J S** このグループのすべての望ましい (訳注: あなたが指定した条件に合致する) 記事 (see Section 6.9.2 [Agent Categories], page 211) をダウンロードします。(`gnus-agent-fetch-group`)。
- J s** このグループのすべてのプロセス印が付いた記事をダウンロードします。(`gnus-agent-summary-fetch-series`)。
- J u** 現在のグループのダウンロード可能な記事を、すべてダウンロードします (`gnus-agent-summary-fetch-group`)。

6.9.3.3 サーバーエージェント命令

- J a** 現在のサーバーを Gnus エージェントで扱われるサーバーのリストに追加します (`gnus-agent-add-server`)。
- J r** 現在のサーバーを Gnus エージェントで扱われるサーバーのリストから削除します (`gnus-agent-remove-server`)。

6.9.4 エージェントの視覚効果

Unplugged のときに概略を開くと、現在エージェントに格納されているヘッダーよりも多くの記事があることを Gnus がそのグループの active (訳注: 何番から何番までの記事があるかを示す管理情報) の範囲から知っている場合には、表題が ‘[Undownloaded article #####]’ になっているいくつかの記事を見るかもしれません。それらは見当たらないヘッダーのための穴埋め (placeholders) です。印を設定することは別として、それらの穴埋めの一つでできることは多くはありません。最終的に Gnus がグループのヘッダーを取って来る機会を得たときに、それらの穴埋めは実際のヘッダーで自動的に置き換えられるでしょう。気になるならば、それらの穴埋めを読み飛ばすために、概略バッファの動作を操作することができます (`gnus-auto-goto-ignores` 参照)。

すべての人にとって明白かもしれませんが、オフラインのときに利用できるのは、plugged だった期間にエージェントに取り込まれたヘッダーと記事だけです。言い換えると「plugged だった期間に取り込むことを忘れると、オフラインのセッションを満足できるものにするには足りない」ということです。この理由のために、エージェントは概略バッファに二つの視覚効果を加えます。これらの効果は、オフラインのときにどの記事が利用できるかをいつも知らせるために、それぞれの記事のダウンロードの状態を表示します。

第一の視覚効果は ‘%0’ 仕様です。gnus-summary-line-format をカスタマイズしてこの指示子を含めると、記事のダウンロードの状態を示すために単一の文字を表示する場所が加わります。エージェントがキャッシュのどちらかに取り込まれた記事は、gnus-downloaded-mark (デフォルトは ‘+’) を表示します。それら以外のすべての記事は gnus-undownloaded-mark (デフォルトは ‘-’) を表示します。エージェント化されていないグループを開くと、空白 (‘ ’) が表示されます。

第二の視覚効果はダウンロードされていないことを示すフェースです。多くの Gnus の利用者に好感と嫌悪をもたらすであろう、記事のスコアを三段階 (low, normal, high) で表示するフェースがあります。問題は、フェースの選択が条件検査とフェース名のリスト (gnus-summary-highlight 参照) によって制御されることです。それぞれの条件は、それがリストの中に現れる順に検査されるので、後の条件よりも前の条件が優先されます。これが意味するすべては、ダウンロードされていない

い記事に可視記事 (ticked) の印を付けても、その記事は可視記事のフェースではなくて、ダウンロードされていない記事のフェースで表示し続けられるということです。

(記事を読むたびに同じ記事をダウンロードしないようにするため、または接続時間を最小にするために) エージェントをキャッシュとして使う場合は、ダウンロードされていない記事のフェースはおそらく良い考えのように思えるでしょう。ダウンロードされた記事に対してすべての仕事 (印を付ける、読む、削除する) を行なえば、いつも通常のフェースが現れるからです。しかし、NOV をキャッシュすることによってオンライン性能を改善するためにエージェントを使っている利用者にとっては (Gnus 5.10.2 以降のほとんどの利用者にとっては)、ダウンロードされていない記事のフェースが見えるかもしれないことは、まったくひどいものでしょう。これは、それらのどの記事もエージェントに取り込まれていないので、ダウンロードされていない記事のフェースのために、すべての普通のフェースが目立たなくなってしまうだろうという問題です。

ダウンロードされていない記事のフェースを使いたい場合は、`agent-enable-undownloaded-faces` グループパラメーターを `t` に設定して、ダウンロードされていない記事のフェースを有効にしなければなりません。このパラメーターは他のすべてのエージェントパラメーターと同様に、エージェント分類 (see Section 6.9.2 [Agent Categories], page 211)、グループトピック (see Section 2.16.5 [Topic Parameters], page 37)、あるいは個々のグループ (see Section 2.10 [Group Parameters], page 23) に対して設定することができます。

エージェントを使うすべての利用者に共通した一つの問題は、それがディスクの容量をいかに速く使い尽くすことができるかということです。あなたが多くのグループでエージェントを使っている場合、事実上ディスク容量を回復することはさらにもっと困難です。一つの解決手段は `gnus-group-line-format` で用意されている `'%F'` 形式です。この形式は、エージェントとキャッシュの両方で取得した記事によって占められる実際のディスク容量を表示します。どのグループが最も多い容量を使うかを知ることによって、利用者は記事を「エージェント期限切れ消去」する場合に、どこに努力を集中するべきかがわかります。

6.9.5 キャッシュとしてのエージェント

Gnus が `plugged` であるときに、すでにヘッダーや記事がエージェントに格納されているのならば、それらを再びダウンロードするのは効率的ではありません。そのため Gnus は通常ヘッダーを一回だけダウンロードしてエージェントに格納します。それらのヘッダーは後に概略バッファを生成するときに、`plugged` か `unplugged` にかかわらずに使われます。デフォルトでは記事は (それはたくさんのディスク容量を浪費するかもしれないので) エージェントにキャッシュされませんが、すでにエージェントにダウンロードした記事があるならば、Gnus はサーバーから再び記事をダウンロードせずに、手元に格納されたコピーを使います。

あなたがそう望むのであれば、`plugged` な期間は常にヘッダーと記事をダウンロードするように、エージェント (`gnus-agent-cache` 参照 Section 6.9.11 [Agent Variables], page 221) を設定することができます。Gnus はほとんど確かにもっと遅くなりますが、サーバーとの同期は保たれます。`nntp` か `nnimap` バックエンドを使っている場合は、たぶんこの最後の点は意味をなさないでしょう。

6.9.6 エージェント期限切れ消去

エージェントバックエンド `nnagent` は期限切れ消去を扱いません。えーと、少なくとも他のバックエンドのようにそれを扱いません。その代わりに、`gnus-agent-expire-days` の日数よりも古い既読記事をすべて消去する、特別な `gnus-agent-expire` と `gnus-agent-expire-group` 命令があります。これらはあなたがディスク容量を使い切りそうだと思ったときに、いつでも実行することができます。どちらも特に速くも効率的でもなく、それらの一つをいったん始めてしまったら (`C-g` やその他で) 中断することもあまり良いことではありません。

例えば `gnus-request-expire-articles` のような他の関数は、エージェントをグループに同期させるために `gnus-agent-expire` を実行するかもしれないことに注意して下さい。

`agent-enable-expiration` というエージェントのパラメーターを、選択したグループでの期限切れ消去を抑制するために使うことができます。

`gnus-agent-expire-all` が `nil` でなければ、エージェントの期限切れ消去コマンド群はすべての記事—未読、既読、可視、保留記事を消去します。もし `nil` (これがデフォルト) であれば、既読記事のみが消去の対象となり、未読、可視、さらに保留記事は無期限に保持されます。

期限切れ消去されるはずなのに残っている記事を見つけたならば、もしかするといくつかの Gnus エージェントファイルが壊れています。起こりうる問題を修復するために、`gnus-agent-regenerate` と `gnus-agent-regenerate-group` という特別なコマンドがあります。

6.9.7 エージェントを作り直す

`nnagent` によって使われるローカルのデータ構造は、ある例外的な条件によっておかしくなってしまうかもしれません。これが起こると `nnagent` の機能性が下がるかもしれないし、失敗しさえするかもしれません。この問題の解決策は、内部の矛盾をすべて削除することによって、ローカルのデータ構造を修復することです。

例えば、記事をエージェントにダウンロードしている間にサーバーへの接続が切れてしまう場合、ローカルのデータ構造は接続が切れる前に記事が首尾良くダウンロードされたかどうかを知りません。`gnus-agent-regenerate` または `gnus-agent-regenerate-group` を実行すると、そのような記事を二回ダウンロードしなくても済むようにデータ構造を更新します。

`gnus-agent-regenerate` コマンドは、すべてのエージェント化されたグループで `gnus-agent-regenerate-group` を実行します。どのバッファ上でも `gnus-agent-regenerate` を実行することができますが、最初にすべての概略バッファを閉じることを強く勧めます。

`gnus-agent-regenerate-group` コマンドは、ローカルの NOV (ヘッダー) データベースを修復するために、個々の記事のローカルなコピーを使います。その後それは、どの記事がローカルに格納されるかを記録しておくための内部データ構造を更新します。引数を与えると、エージェントの中の記事に未読の印を付けます。

6.9.8 エージェントとフラグ

エージェントは Gnus のどんなバックエンドでも、例えばサーバーにフラグ (既読 (read)、可視 (ticked) など) を格納する `nnimap` のようなものでも動作します。しかし悲しいかな、エージェントはどのバックエンドがそれらのフラグを `‘.newsrc’` ではなく、そのバックエンドのサーバーで維持するかを、実際には知りません。そのためエージェントは、`unplugged` または接続されていない間に行なったすべてのフラグへの変更を、常に自身のファイルに記録します。

再び接続すると、Gnus は変更されたすべてのフラグを検査して、それらをサーバーと同期させるかどうかを尋ねます。この挙動は `gnus-agent-synchronize-flags` でカスタマイズすることができます。

`gnus-agent-synchronize-flags` が `nil` だったら、エージェントは自動的にフラグを同期させることはしません。それがデフォルトの `ask` だったら、エージェントはあなたが再接続したときにあなたが何らかの変更を行っていたかどうかを調べて、もしそうだったら、それらを同期させたいかどうかを尋ねます。それら以外の値だった場合は、すべてのフラグは自動的に同期させられます。

再接続したときに自動でフラグを同期させたくないなら、手動でそれを行なうこともできます。これにはグループバッファの `J Y` キーに割り当てられた `gnus-agent-synchronize-flags` コマンドを使って下さい。

技術的注釈: すべてのローカルなフラグをサーバーに「押し込む」同期のアルゴリズムは動作しませんが、利用者によって変更されたフラグだけを変更して、サーバー側で見えるフラグを一つずつ更新することは可能です。したがって、あなたが記事の一つのフラグをセットして、そのグループを抜け出してから再度そのグループを選択してそのフラグを消せば、あなたが「同期」の操作を行なったときに、そのフラグはセットされてサーバーからは削除されます。順番待ち (queue) に入れられたフラグに関する動作は、エージェントディレクトリーにあるサーバー毎の flags ファイルの中で見つかるでしょう。それらはあなたがフラグを同期させたときに空になります。

6.9.9 エージェントを IMAP で使う方法

エージェントは `nnimap` を含む Gnus のどんなバックエンドでも動作します。しかし NNTP と IMAP にはいくつかの概念の違いがあるので、この章ではサーバーとの接続が絶たれたモードでの IMAP のクライアントとして、Gnus エージェントをより円滑に使えるようにするための、いくつかの情報を提供します。

サーバーとの接続が絶たれているときの IMAP クライアントにあなたが期待するであろういくつかの機能は、現在のエージェントには盛り込まれていません。それらは以下の通りです:

- Unplugged のときの `nnimap` グループへのコピーと移動。
- Unplugged のときの `nnimap` グループの作成と削除。

6.9.10 差出用メッセージ

Gnus が unplugged のとき、デフォルトではすべての差出用メッセージ (メールとニュースの両方) は下書きグループ “queue” (see Section 5.7 [Drafts], page 130) に格納されます。投稿した後でも、そこでそのメッセージを見たり編集するのは意のままです。

送出するメールが queue される (順番待ちになる) 状況を制御することは可能です (`gnus-agent-queue-mail`, Section 6.9.11 [Agent Variables], page 221 参照)。Gnus が unplugged のとき、外に送り出すニュースは常に queue されるだけです。

下書きグループから、そこで使える特別な命令を使ってメッセージを送信することもできるし、グループバッファ内で `JS` を使って、下書きグループ内のすべての送信可能なメッセージを送信することもできます。ニュースの投稿は Gnus が plugged のときだけできますが、メールはいつでも送信することができます。

Unplugged のときにメールの送信ができなくて、かつ unplugged のときにうっかり `JS` を叩いてしまうことが心配ならば、Gnus にあなたの行動を確認させることができます (`gnus-agent-prompt-send-queue`, Section 6.9.11 [Agent Variables], page 221 参照)。

6.9.11 エージェント変数

`gnus-agent`

エージェントが有効になっているかどうか。デフォルトは `t` です。最初に有効にされると、いくつかのバックエンドを自動的にエージェント化するために、エージェントは `gnus-agent-auto-agentize-methods` を使います。サーバーバッファでエージェントのコマンドを使うことによって、どのバックエンドをエージェント化するかを変更することができます。

サーバーバッファに入るには、グループバッファで `^` (`gnus-group-enter-server-mode`) を使って下さい。

`gnus-agent-directory`

Gnus エージェントがファイルを格納する場所です。デフォルトは `~/News/agent/` です。

gnus-agent-handle-level

この変数の値より高いレベル (see Section 2.6 [Group Levels], page 19) のグループは、エージェントからは無視されます。デフォルトは `gnus-level-subscribed` で、これはデフォルトでは、購読しているグループのみがエージェントの処理の対象となるということです。

gnus-agent-plugged-hook

ネットワークに接続されたときに実行されるフックです。

gnus-agent-unplugged-hook

ネットワークから切断されたときに実行されるフックです。

gnus-agent-fetched-hook

記事を取り込み終わったときに実行されるフックです。

gnus-agent-cache

Plugged のときに、ローカルに格納されている NOV と記事を使うかどうかを制御する変数で、例えばエージェントをキャッシュとして使うには必須です。デフォルトでは非-nil で、エージェントをキャッシュとして使います。

gnus-agent-go-online

`gnus-agent-go-online` が nil だったら、エージェントはオフライン状態のサーバーをオンライン状態にしません。ask だったら、それがデフォルトですが、エージェントは再接続するときにオフライン状態のサーバーをオンライン状態にしたいかどうかを尋ねます。それ以外の値だったら、オフライン状態のサーバーは自動的にオンライン状態になります。

gnus-agent-mark-unread-after-downloaded

`gnus-agent-mark-unread-after-downloaded` が 非-nil だったら、ダウンロードした後で記事に未読の印を付けます。これは通常、新しくダウンロードされた記事を明確に未読にするための安全な行為です。デフォルトは t です。

gnus-agent-synchronize-flags

`gnus-agent-synchronize-flags` が nil だったら、エージェントは決して自動的にフラグを同期させません。それが ask だったら (それがデフォルトです)、エージェントはすべての変更を検査して、再び接続したときにそれらを同期させるかどうかを尋ねます。nil でも ask でもなかったら、すべてのフラグが自動的に同期させられます。

gnus-agent-consider-all-articles

`gnus-agent-consider-all-articles` が非-nil だったら、エージェントはすべての記事について、それらをダウンロードする必要があるかどうかをエージェントの述語に決定させます。nil だった場合、それがデフォルトですが、エージェントは未読の記事をダウンロードするかどうかだけを述語に決定させます。これを有効にするのならば、後でエージェントが期限切れ消去する記事を何度も繰り返しダウンロードしないように、エージェントの期限切れ消去の設定 (see Section 6.9.2.3 [Category Variables], page 216) を見直す必要があるでしょう。

gnus-agent-max-fetch-size

エージェントは、取得した記事を個々のファイルに入れるための解析を行なう前に、それらを一時的なバッファへ取り込みます。最大のバッファサイズを超過しないようにするために、記事がすべて取得されるまで、エージェントは取得と解析を交互に行ないます。`gnus-agent-max-fetch-size` は、繰り返しがどれくらい頻繁に起きるかを

制御するための、サイズの限界を規定します。大きな値は性能を向上させます。小さな値は、万が一取得している間に接続が切れた場合に、遅れ時間を最小にします (グループの状態を更新するために `gnus-agent-regenerate-group` を実行する必要があるかもしれません。でも、接続が切れる前に解析されたすべての記事は、`unplugged` の期間に利用することができるでしょう。)。繰り返しに遭遇することは珍しいので、デフォルトは 10M です

`gnus-server-unopen-status`

エージェント変数ではないかもしれないけれどエージェントに密接に関連するこの変数は、Gnus がサーバーに接続できないときに何をするかを指示します。エージェントが活性化されると、デフォルトの `nil` では、サーバーとの接続を絶つかエージェントを `unplugged` にするかを利用者に尋ねます。エージェントが不活性化されると、Gnus はいつも単にサーバーとの接続を絶ちます。この変数の他の選択肢には `denied` と `offline` があり、後者はエージェントを使う場合だけ有効です。

`gnus-auto-goto-ignores`

おおかたの人は、エージェント変数ではないけれども密接に関連するもう一つの変数をここで探すでしょう。この変数は、ダウンロードされていない (ヘッダーだけがエージェントに格納された)、そして取り込まれていない (記事もヘッダーも格納されていない) 記事の周りでどう移動するかを概略バッファに伝えます。

有効な値は `nil` (どの記事にも移動する)、`undownloaded` (`unplugged` のときは取り込まれていない記事を無視する)、`always-undownloaded` (取り込まれていない記事を常に無視する)、`unfetched` (ヘッダーが取り込まれていない記事を無視する) です。

`gnus-agent-queue-mail`

`gnus-agent-queue-mail` を `always` にすると、Gnus はメールをいきなり送信してしまうのではなく、常に `queue` (順番待ち) に入れます。t だったら Gnus は `unplugged` のときだけメールを `queue` に入れます。nil だったら `queue` に入れません。デフォルトは `t` です。

`gnus-agent-prompt-send-queue`

`gnus-agent-prompt-send-queue` が非-nil だったら、`unplugged` であるのにもかかわらず `J S` を叩いた場合に、Gnus は本当にそれを行なっても良いかどうかを確認します。デフォルトは `nil` です。

`gnus-agent-auto-agentize-methods`

あなたが以前にエージェントを使ったことが無い (もっと技術的には `~/News/agent/lib/servers` が無い場合)、Gnus はほんの少数のサーバーを自動的にエージェント化します。この変数はどのバックエンドを自動でエージェント化すべきかを制御します。一般に、エージェント化することが有用なのは遠隔バックエンドに対してだけです。自動的にエージェント化することは、サーバーに対して `J a` を実行するのと同じ効果があります (see Section 6.9.3.3 [Server Agent Commands], page 218)。もしファイルが存在するならば、それらを追加したり削除するためにサーバーを手動で操作しなければなりません。この変数は最初に Gnus を起動したときだけ適用されます。デフォルトは `(nntp nnimap)` です。

6.9.12 設定例

あなたがこのマニュアルを読みたくなくて、ごく標準的な設定を行なっているのならば、`~/gnus.el` ファイルとして何か以下のようなものを使って始めても良いでしょう。

```
;; Gnus がどのようにニュースを取得するかを定義します。ここ
;; では ISP のサーバーから NNTP で取ってくることにします。
(setq gnus-select-method '(nntp "news.your-isp.com"))

;; Gnus がどのようにメールを読むかを定義します。
;; ISP の POP サーバーからメールを読むことにします。
(setq mail-sources '((pop :server "pop.your-isp.com")))

;; Gnus がメールをどのように格納するかを指定します。
;; nnml グループを使うことにします。
(setq gnus-secondary-select-methods '((nnml "")))

;; Gnus をオフラインニュースリーダーにします。
;; (gnus-agentize) ; 旧式の設定。
;; (setq gnus-agent t) ; 現在のデフォルト。
```

基本的にはこれだけで良いはずですが。これを ‘~/gnus.el’ ファイルに入れて、必要に応じて編集し、PPP (や何か) を起動して、*M-x gnus* とタイプして下さい。

あなたが Gnus を走らせたのが初めてであれば、自動的にわずかなデフォルトのニュースグループが読めるようになります。おそらくもっとたくさんのグループを購読したくなるでしょう。そのためには、*A A* 命令でグループの完全なリストを NNTP サーバーに問い合わせなければなりません。これは普通はとても時間がかかりますが、一度だけしか実行する必要はありません。

読み込みと解析にしばらく時間を費やした後で、グループの一覧が現れます。そうしたら、読みたいグループを *u* 命令で購読できるようにして下さい。読みたいグループを全部購読できるようにしたら、*1* で killed (削除された) グループをすべて画面から消去しましょう。(*A k* で killed グループはすべて戻ってきます。)

今やすぐにグループを読むこともできるし、*J s* 命令で記事をダウンロードすることもできます。あとはこのマニュアルの残りを読んで、他の億千万の項目からカスタマイズしたいことを見つけ出して下さい。

6.9.13 一括エージェント処理

Gnus エージェントに記事を取得させるのは (そしてあなたの書いた何かのメッセージを投稿するのは)、いったんものごとを正しく設定してしまえば非常に簡単です。以下のシェルスクリプトは必要なことをすべてやってくれるでしょう。

以下の呪文をコマンドラインで使うことによって、完全なバッチコマンドを走らせることができます:

```
#!/bin/sh
emacs -batch -l ~/.emacs -l ~/.gnus.el -f gnus-agent-batch >/dev/null 2>&1
```

6.9.14 エージェントの問題点

Gnus エージェントは、よくある他のオフラインニュースリーダーのようにには動作しません。これらは架空の人々からの良くある質問です:

Plugged のときに記事を読んだら、それはエージェントに入るのですか?

いいえ。この動作をお望みなら `gnus-select-article-hook` に関数 `gnus-agent-fetch-selected-article` を加えて下さい。

Plugged のときに記事を読んで、エージェントに記事が存在している場合、
もう一回ダウンロードされるのですか？

いいえ、ただし `gnus-agent-cache` が `nil` でなかったら、ですが。

要約すると、Gnus が *unplugged* のときはローカルに保存された記事を見るだけです。*Plugged* のときは ISP と話し、かつローカルに持っている記事も使うでしょう。

7 スコア

ほかの人たちは「削除ファイル」(*kill files*)を使いますが、ここ Gnus タワーにいる私たちは削除よりもスコアの方が好きです。彼らとけんかをするよりは切り替えてしまう方がましでしょう。それらは完全に違うことをするので、真っ直ぐに座って注意を払って下さい!

すべての記事はデフォルトのスコア (`gnus-summary-default-score`) の値を持っていて、デフォルトでは 0 です。このスコアは対話的に、またはスコアファイル (`score file`) によって、上げられるか下げられるかします。`gnus-summary-mark-below` よりも低いスコアを持っている記事には既読の印が付きます。

Gnus は概略バッファを作成する前に、現在のグループに適用されるどんな「スコアファイル」も読み込みます。

現在の記事に基づいてスコアのエントリーを挿入する、複数の概略バッファの命令があります。例えば、Gnus に特定の表題の記事のスコアを下げたり上げたりするように求めることができます。

二種類のスコア・エントリーがあります: 永続的なものと一時的なものです。一時的なスコア・エントリーは、自分自身で期限切れ消去するエントリーです。例えば一週間以上使われていないエントリーは、スコアファイルの大きさを小さくしておくために静かに削除されます。

7.1 概略スコア命令

スコア・エントリーを変更するスコア命令は、実際に本当のスコアファイルを修正するわけではありません。それはあまりに非効率です。Gnus は以前にロードされたスコアファイルのキャッシュを保持していて、その一つが「現在のスコアファイルの連想リスト」だと見なされます。スコア命令は単にこのリストにエントリーを挿入し、グループから出るときに、このリストは保存されます。

現在のスコアファイルは、実際にそのようなスコアファイルが存在しない場合でも、デフォルトでグループのローカルスコアファイルになります。スコア命令を何か他のスコアファイル (例えば `'all.SCORE'`) に挿入するには、まずこのスコアファイルを現在のものにしなければなりません。

以下はスコアファイルを実際に変更しない、一般的なスコア命令です:

<code>V s</code>	現在の記事のスコアを設定します (<code>gnus-summary-set-score</code>)。
<code>V S</code>	現在の記事のスコアを表示します (<code>gnus-summary-current-score</code>)。
<code>V t</code>	現在の記事に使われているすべてのスコア規則を表示します (<code>gnus-score-find-trace</code>)。*Score Trace* バッファにおいて、現在の行のスコア規則に対応するスコアファイルを編集するには <code>e</code> を、スコアファイルの清書 (<code>gnus-score-pretty-print</code>) と編集を行なうためには <code>f</code> をタイプして下さい。
<code>V w</code>	スコアで使われている語のリストを表示します (<code>gnus-score-find-favourite-words</code>)。
<code>V R</code>	現在の概略でスコアの処理を実行します (<code>gnus-summay-rescore</code>)。Gnus には内緒でスコアファイルをいじり回して、その効果を見たいときに役立つでしょう。
<code>V c</code>	違うスコアファイルを現在のものにします (<code>gnus-score-change-score-file</code>)。
<code>V e</code>	現在のスコアファイルを編集します (<code>gnus-score-edit-current-scores</code>)。gnus-score-mode バッファが現れるでしょう (see Section 7.5 [Score File Editing], page 236)。
<code>V f</code>	スコアファイルを編集して、このスコアファイルを現在のものにします (<code>gnus-score-edit-file</code>)。

- V F* キャッシュされているスコアを捨てます (gnus-score-flush-cache)。これはスコアファイルを編集した後で役に立ちます。
- V C* 視覚的に快適な方法でスコアファイルをカスタマイズします (gnus-score-customize)。

以下の命令はローカルスコアファイルを変更します:

- V m* スコアの入力を求めて、それよりも低いスコアのすべての記事に既読の印を付けます (gnus-score-set-mark-below)。
- V x* スコアの入力を求めて、そのスコアより低いすべての記事を削除するためのスコア規則を現在のスコアファイルに付け加えます (gnus-score-set-expunge-below)。

スコア・エントリーを実際に作るためのキー操作は、非常に規則正しい様式にのっとっている、それらすべての (何百もある) 命令を列挙する必要は無いでしょう。

- 最初にタイプするキーは、スコアを増やすときは *I* (*i* の大文字) で、スコアを下げるときは *L* です。
- 二番目のキーは、どのヘッダーでスコアを付けるかを指定します。以下のキーを使うことができます:
 - a* 著者 (author) の名前でスコアを付けます。
 - s* 表題 (subject) の行でスコアを付けます。
 - x* Xref 行、すなわちクロスポスト行でスコアを付けます。
 - r* References 行でスコアを付けます。
 - d* 日付 (date) でスコアを付けます。
 - l* 行数 (number of lines) でスコアを付けます。
 - i* Message-ID ヘッダーでスコアを付けます。
 - e* NNTP サーバーが追加のヘッダーのデータを overview で捕捉していれば、その「追加」のヘッダー (すなわち (gnus-extra-headers に設定されているもの) の一つでスコアを付けます。
 - f* フォローアップ (followup) でスコアを付けます—これは著者名と合致するかどうかを調べて、この著者へのフォローアップでスコアを加えます。(このキーを使うことは、‘ADAPT’ ファイルの生成をもたらします。)
 - b* 記事の本文でスコアを付けます。
 - h* ヘッダーでスコアを付けます。
 - t* スレッドでスコアを付けます。(このキーを使うことは、‘ADAPT’ ファイルの生成をもたらします。)
- 三番目のキーは合致の型です。どの合致の型が有効なのは、どのヘッダーでスコアを付けようとしているかに依ります。

文字列 (strings)

- e* 正確な (exact) 合致です。
- s* 文字列の一部の (substring) 合致です。

<i>f</i>	大雑把な (fuzzy) 合致です (see Section 8.18 [Fuzzy Matching], page 269)。
<i>r</i>	正規表現 (regexp) の合致です。
日付 (date)	
<i>b</i>	日付の前 (before) です。
<i>a</i>	日付の後 (after) です。
<i>n</i>	その日付です。
数値 (number)	
<	数値より小さいものです。
=	数値と等しいものです。
>	数値より大きいものです。

4. 普通はこれで最後の四つ目のキーは、これが一時的な (すなわち期限切れ消去される) スコア・エントリーか、永続的な (すなわち期限切れ消去でない) スコア・エントリーか、またはスコアファイルに追加せずにただちにスコア付けを行なうか、のどれかを指定します。

<i>t</i>	一時的な (temporary) スコア・エントリーです。
<i>p</i>	永続的な (permanent) スコア・エントリーです。
<i>i</i>	ただちに (immediate) スコア付けを行ないます。

5. もし 'e' (追加の (extra)) ヘッダーでスコア付けを行なっていると、それでスコア付けをしたいヘッダーの名前を尋ねられるでしょう。これは `gnus-extra-headers` にある名前でなければなりません。'TAB' による補完ができます。

そういうわけで、現在の著者への、正確な合致に基づいて、永続的なスコアを、増やしたい、という場合のキーは *I a e p* です。表題への、文字列の一部合致に基づいて、一時的なスコア・エントリーを作り、そのスコアを下げたい、という場合のキーは *L s s t* です。ずいぶん簡単ですね。

ものごとを少し複雑にするためにショートカット・キーがあります。二番目が三番目のキーに大文字を使うと、Gnus は残る一つか二つのキーにデフォルト値を使います。デフォルトは「文字列の一部」と「一時的」です。ですから *IA* は *I a s t* と同じで、*I a R* は *I a r t* と同じです。

これらの関数は、数値接頭引数とシンボル接頭引数を受け付けます (see Section 8.3 [Symbolic Prefixes], page 250)。数値接頭引数はどのくらい記事のスコアを下げる (もしくは上げる) かを指定します。シンボル接頭引数 *a* は、現在のスコアファイルの代わりに 'all.SCORE' ファイルをその命令のために使うことを指示します。

`gnus-score-mimic-keymap` はこれらの命令がキーマップであるかのように振る舞うかどうかを指定します。

7.2 グループスコア命令

残念ながら、まだたくさんはありません。

<i>W e</i>	すべてのグループに適用される all.SCORE ファイルを編集します。gnus-score-mode のバッファが現れるでしょう (see Section 7.5 [Score File Editing], page 236)。
<i>W f</i>	何度もスコア連想リストを再読み込みすることを避けるために、Gnus はそれらのキャッシュを保持しています。この命令はキャッシュを空っぽにします (gnus-score-flush-cache)。

以下のようなやり方で、コマンド行からスコア付けをすることができます:

```
& emacs -batch -l ~/.emacs -l ~/.gnus.el -f gnus-batch-score
```

7.3 スコア変数

gnus-use-scoring

nil であれば、Gnus はスコアファイルを調べず、一般的にはスコア関連の仕事をまったくしません。これはデフォルトで t です。

gnus-kill-killed

この変数が nil であれば、Gnus はすでに削除 (kill) の処理を実行された記事に決してスコアファイルを適用しません。これはたくさんの時間を節約する一方、削除ファイルをグループに適用していて、削除ファイルを変更し、もっと多くの記事を削除するためにそれを再実行しても、それは動作しないということにもなります。それをするためにはこの変数を t にしなければなりません。(これはデフォルトで t です。)

gnus-kill-files-directory

すべての削除 (kill) とスコアのファイルはこのディレクトリーに格納されます。その値は、デフォルトでは環境変数 `SAVEDIR` によって初期化されます。デフォルトは `~/News/` です。

gnus-score-file-suffix

スコアファイルの名前を得るためにグループ名に加える接尾語です (デフォルトは `'SCORE'` です)。

gnus-score-uncacheable-files

通常すべてのスコアファイルは、スコアファイルの過剰な再読み込みを避けるためにキャッシュされます。しかし、このために Emacs が大きく肥大化するようならば、再び必要とされそうもないスコアファイルを取り除くためにこの正規表現を使うことができます。 `'all.SCORE'` のキャッシュをやめてしまうのは間違った考えですが、 `'comp.infosystems.www.authoring.misc.ADAPT'` をキャッシュしないのは良い考えかもしれません。実際のところ、この変数のデフォルトは `'ADAPT$'` で、適応スコアファイルはキャッシュされません。

gnus-save-score

もし本当に複雑なスコアファイルを持っていて、たくさんの一括 (batch) スコア付けを行なうのであれば、この変数を t に設定しても良いでしょう。これは Gnus にスコアを `./newsrc.eld` ファイルに保存させます。

これを t に設定しないと、手動で付けたスコア (`V s (gnus-summary-set-score)` で設定したようなもの) は訪れたグループ群を横切って保持されることはありません。

gnus-score-interactive-default-score

スコアを上げ/下げするために、すべての対話的スコア上げ/下げ命令によって使われるスコアです。デフォルトは 1000 で、過剰に思えるかもしれませんが、適応スコア付けをするための十分な余地を確保するためです。手で入力したデータを、適応スコア付けからの小さな変更で上書きされたくはないのです。

gnus-summary-default-score

記事のスコアのデフォルトで、デフォルトでは 0 になっています。

`gnus-summary-expunge-below`

この変数より低いスコアを持つ記事は概略の行に表示されません。デフォルトは `nil` で、どの記事も隠されないということです。これは各概略バッファーにおけるローカル変数で、`gnus-summary-mode-hook` によって設定されなければなりません。

`gnus-score-over-mark`

デフォルトのスコアより大きなスコアを持つ記事に対して（概略行の三桁目で）使われる印です。デフォルトは `+` です。

`gnus-score-below-mark`

デフォルトのスコアより小さなスコアを持つ記事に対して（概略行の三桁目で）使われる印です。デフォルトは `-` です。

`gnus-score-find-score-files-function`

現在のグループのためのスコアファイルを見つけるために使われる関数です。この関数はグループ名を引数として呼ばれます。

あらかじめ定義されている使用可能な関数は:

`gnus-score-find-single`

グループ自身のスコアファイルだけを適用します。

`gnus-score-find-bnews`

すべての合致するスコアファイルを `bnews` 構文を使って適用します。これがデフォルトです。例えば、現在のグループが `'gnu.emacs.gnus'` ならば、`'gnu.all.SCORE'`、`'not.alt.SCORE'` と `'gnu.all.SCORE'` がすべて適用されます。要するに、スコアファイル名の `'all'` が `'.*'` に変換され、それから正規表現の合致がなされます。

これは、すべてのグループに適用したいスコア・エントリーがいくつかある場合は、それらのエントリーを `'all.SCORE'` ファイルに入れるということです。

Gnus は、より一般的なスコアファイルを、より特定のスコアファイルより前に適用しようとするものの、スコアファイルはややランダムな順番で適用されます。これはスコアファイル名の要素の数を調べることによって行なわれます—`'all'` 要素を取り除いて。

`gnus-score-find-hierarchical`

すべての親グループから、順にすべてのスコアファイルを適用します。これは `'all.SCORE'` のようなスコアファイルを持つことはできないけれど、`'SCORE'`、`'comp.SCORE'` および `'comp.emacs.SCORE'` を、それぞれのサーバーに対して持つことができるということです。

この変数は関数のリストであることもできます。その場合、これらすべての関数がグループ名を引数として呼ばれ、返されたすべてのスコアファイルのリストが適用されます。これらの関数は、直接スコア連想リストのリストのリストを返すこともできます。その場合、それらのファイルでないスコア連想リストを返す関数は、返される最後のスコアファイルがローカルスコアファイルであることを確実にするために、おそらく「本当の」スコアファイル関数よりも前に置かれるべきでしょう。ふう。

例えば、サーバーを特定しない全体スコアファイルを使って、階層的に（親グループから順に）スコア付けを行なうならば、次の値を使えば良いでしょう。

```
(list (lambda (group) ("all.SCORE"))
      'gnus-score-find-hierarchical)
```

gnus-score-expiry-days

この変数は、使われていないスコアファイルエントリーが期限切れ消去されるまでに、どのくらいの日数が経過すべきかを指定します。この変数が `nil` であると、スコアファイルエントリーは削除されません。デフォルトは 7 です。

gnus-update-score-entry-dates

この変数が `nil` でないと、一時的に合致したスコア・エントリーは日付が更新されます。(これは Gnus が期限切れ消去を操作している方法です—すべての合致しないエントリーは古くなりすぎるのに対して、合致するエントリーは新鮮で若いままです。) しかし、この変数を `nil` に設定すると、合致するエントリーでさえも古くなって、ああ、そう、あの冷酷な死神と直面することになるでしょう。

gnus-score-after-write-file-function

スコアファイルが書かれた直後に、その名前を引数として呼ばれる関数です。

gnus-score-thread-simplify

この変数が `nil` でないと、記事の表題は表題でスコア付けを行なうために、スレッドと同じやり方で (現在の `gnus-simplify-subject-functions` の値に従って) 単純化されます。スコア・エントリーが「文字列の一部への合致」か「正確な合致」を使っていると、その合致もこのやり方で単純化されます。

7.4 スコアファイル様式

スコアファイルは普通は単一の様式だけを含む `emacs-lisp` ファイルです。無頓着な利用者はこれを編集しないで下さい。すべては概略バッファから変更することができます。

にもかかわらず、それを自分でいじってみたくなったのなら、例があります:

```
((("from"
  ("Lars Ingebrigtsen" -10000)
  ("Per Abrahamsen")
  ("larsi\\|lmi" -50000 nil R))
  ("subject"
  ("Ding is Badd" nil 728373))
  ("xref"
  ("alt.politics" -1000 728372 s))
  ("lines"
  (2 -100 nil <))
  (mark 0)
  (expunge -1000)
  (mark-and-expunge -10)
  (read-only nil)
  (orphan -10)
  (adapt t)
  (files "/hom/larsi/News/gnu.SCORE")
  (exclude-files "all.SCORE")
  (local (gnus-newsgroup-auto-expire t)
    (gnus-summary-make-false-root empty)))
```

(eval (ding)))

この例はたいていのスコアファイルの要素を説明しています。別のやり方については、Section 7.15 [Advanced Scoring], page 244 を見て下さい。

これがとても Lisp コードのように見えるとしても、実際はここにあるものは何も eval (評価) されません。しかしこの様式を読み込むために Lisp リーダーが使われるので、意味的に有効でないとしても、文法的には正当なものです。

この連想リストでは六つのキーがサポートされています:

文字列 (STRING)

キーが文字列だったら、それは合致が実行されるヘッダーの名前です。スコア付けはこれら八つのヘッダーだけで行なうことができます: From, Subject, References, Message-ID, Xref, Lines, Chars および Date です。これらのヘッダーに加えて、Gnus に、記事全体を取得して記事のより大きな部分で合致を行なわせる三つの文字列があります: Body は記事の本文で合致を行ない、Head は記事のヘッダーで合致を行ない、All は記事全体で合致を行ないます。これら最後の三つのキーを使うと、グループに入る速度を かなり 遅くしてしまうことに気を付けて下さい。スコアを付けることができる最後の「ヘッダー」は Followup です。これらのスコア・エントリーは、これらのスコア・エントリーに合致する記事へのすべてのフォローアップのための追加が行なわれている、新しいスコア・エントリーに帰着するでしょう。

このキーに続くのは任意の数のスコア・エントリーで、それぞれのスコア・エントリーは一つから四つまでの要素を持ちます。

1. 最初の要素は「合致要素」です。これはたいていのヘッダーでは文字列ですが、Lines と Chars ヘッダーでは整数でなければなりません。
2. もし二番目の要素があるなら、それは数値の「スコア要素」でなければなりません。この数値は負の無限大から正の無限大までの間の整数でなければなりません。合致が成功すると、この数値が記事のスコアに加えられます。この要素が存在していない場合は、代わりに gnus-score-interactive-default-score の数値が使われます。デフォルトは 1000 です。
3. もし三番目の要素があるなら、それは数値の「日付要素」でなければなりません。この日付は最後にこのスコア・エントリーが合致した時刻を示し、それはスコア・エントリーを期限切れ消去するための機構を提供します。この要素が存在していないと、スコア・エントリーは永続になります。日付は紀元前 1 年 12 月 31 日からの経過した日数で表されます。
4. もし四番目の要素があるなら、それはシンボルの「型要素」でなければなりません。この要素は、このスコア・エントリーが記事に合致するかどうかを調べるために、どの関数が使われるべきかを指定します。

From, Subject, References, Xref, Message-ID

たいていのヘッダー型のために、r と R (正規表現 (regex)), s と S (文字列の一部 (substring)) 型、e と E (正確な合致 (exact match))、および w (語の合致 ((word match)) 型) があります。もしこの要素が無いと、Gnus は文字列の一部の合致が用いられるべきであると仮定します。R, S, E は、合致が大文字と小文字を区別する方法で行なわれる点で他のものと異なります。これらすべての一文字型は、本当は regex, exact, word 型の短縮形で、この方が好みならば代わりに使うことができます。

Extra overview ヘッダーの標準の文字列に的を絞って `gnus-extra-headers` を使っていれば、それらのヘッダーの値でスコアを付けることができます。この場合スコア・エントリーの五番目の要素が、スコアを付けるヘッダーの名前になります。NNTP サーバーが `overview` で `'NNTP-Posting-Host'` を捕捉しているならば、`'all.SCORE'` ファイルの以下のエントリーは、単一のホストを起源とする spam の攻撃に対して有効です:

```
("111.222.333.444" -1000 nil s
  "NNTP-Posting-Host")
```

Lines, Chars

これらの二つのヘッダーは別の合致の型を使います: `<`, `>`, `=`, `>=`, `<=` です。

これらの述語は

```
(PREDICATE HEADER MATCH)
```

の評価が `nil` ではない場合に真です。例えば、上級合致 (`"lines" 4 <`) (see Section 7.15 [Advanced Scoring], page 244) は結果として以下の式になります:

```
(< header-value 4)
```

言い換えると、4 を合致として `<` を `Lines` で使っているときは、記事が 4 行よりも少ないときにスコアが加算されるということです。(混乱して、それが反対ではないかと考えがちです。でも、そうではないのです。私が思うに。)

合致を `Lines` で行なっていると、いくつかのバックエンド (`nndir` のようなもの) は `Lines` ヘッダーを作成しないので、すべての記事が 0 行であるとして扱われてしまうことに気を付けて下さい。これは、少しの行しかない記事のスコアを下げている場合に、変な結果を導くことがあります。

Date

`Date` (日付) ヘッダーには三つのなんとなくばかげている合致の型があります: `before`, `at`, `after` です。私は本当にこれが役立つような機会を想像できないのですが、この関数を提供しないのもなんとなくばかげています。そうした場合のためにあるのです。いつ必要になるかは誰にもわかりません。転ばぬ先の杖。糞 (あつもの) に懲りて膽 (なます) を吹く。本をカバーで判断してはいけません。初めてのデートでエッチしてはいけません。(しかし、私は少なくとも一人、引用しますが、「この関数は欠かせないものであることがわかった」と言った人がいると聞いています。)

もっと役に立つ合致の型は「正規表現」です。それによって、日付の文字列に正規表現を使って合致させることができます。日付はまず ISO8601 の短縮様式 (compact format) に標準化されます—`YYYYMMDDTHHMMSS` です。例えば、すべての年の 4 月 1 日に投稿されたすべての記事に合致させたいのであれば、合致文字列として `'....0401.....'` を使うことができます。(日付は元々の標準時で保存されているので、その記事が投稿された場所での 4 月 1 日に投稿された記事に合致することに注意して下さい。“Time zones”は

家族全員の健全な楽しみですね? (訳注: いくつかある“ Time zones ”というタイトルの曲のことを言っているのかもしれませんが。))

Head, Body, All

これらの三つの合致のキーは From ヘッダー (など) と同じ合致の型を使います。

Followup

この合致のキーはやや特別で、それは From ヘッダーに合致し、合致した記事だけでなくその記事へのすべてのフォローアップのスコアにも影響します。これは例えば、あなた自身の記事へのフォローアップのスコアを増やしたり、良く知られた問題児へのフォローアップ記事のスコアを下げたりするのに使われます。From ヘッダーが使うのと同じ型の合致を使います。(この合致キーを使うと、‘ADAPT’ ファイルを作ることになります。)

Thread

この合致キーは Followup 合致キーと同じ方針に沿って動作します。Message-ID *x* で始まっているスレッド (または副スレッド) にスコアを付けたいのであれば、‘thread’ 合致を付け加えて下さい。これは Reference ヘッダーに *x* を持つそれぞれの記事に、新しい ‘thread’ 合致を追加します。(これらの新しい ‘thread’ 合致はこれらの合致する記事の Message-ID を使います。) これはスレッドのいくつかの記事が完全な References ヘッダーを持っていなかったとしても、スレッド全体のスコアを上げ/下げできることを保証します。これを使うと、スレッドの記事に決定的でないスコアが付くかもしれないということに注意して下さい。(この合致キーを使うと、‘ADAPT’ ファイルを作ることになります。)

mark このエントリーの値は数値でなければなりません。この数値より低いスコアのどんな記事にも既読の印が付きます。

expunge このエントリーの値は数値でなければなりません。この数値より低いスコアのどんな記事も概略バッファーから削除されます。

mark-and-expunge

このエントリーの値は数値でなければなりません。この数値より低いスコアのどんな記事にも既読の印が付き、概略バッファーから削除されます。

thread-mark-and-expunge

このエントリーの値は数値でなければなりません。スコアの総計がこの数値より低いスレッドのすべての記事には既読の印が付き、概略バッファーから削除されます。gnus-thread-score-function はスレッドのスコアの総計をどのように計算するかを指定します。

files このエントリーの値は任意の数のファイル名でなければなりません。それらのファイルもスコアファイルであるとみなされ、これがされたのと同じ方法で読み込まれます。

exclude-files

このエントリーの手がかりは任意の数のファイル名でなければなりません。これらのファイルが何らかの理由で普通は読み込まれるようになっていたとしても、読み込まれません。

eval このエントリーの値は eval (評価) されます。この要素はグローバルスコアファイルを扱っているときは無視されます。

read-only

読み込み専用スコアファイルは更新されたり保存されたりしません。グローバルスコアファイルはこのアトムを使用すべきです (see Section 7.12 [Global Score Files], page 242)。 (注意: 「グローバル」はここでは本当に「全体的」という意味です。個人的なすべてのグループに適用するスコアファイルのことではありません。)

orphan

このエントリーの値は数値でなければなりません。親記事を持たない記事のスコアにこの数値が加えられます。‘comp.lang.c’のような流通量の多いニュースグループを追いかけていると想像して下さい。おそらくほんの少しのスレッドだけを追いたいでしょう。さらに新しいスレッドは見たいでしょう。

以下の二つのスコアファイルエントリーによって、それを行うことができます:

```
(orphan -500)
(mark-and-expunge -100)
```

最初にこのグループに入ったときは、新しいスレッドだけを見るでしょう。そうしたら、興味を持ったスレッドのスコアを上げ (*IT* または *IS* で)、残りを無視 (*Cy*) して下さい。次にグループに入ったときは、興味を持ったスレッドの新しい記事と、まったく新しいスレッドを見ることになります。

すなわち orphan (孤児) スコアアトムは、普通のスコア規則では自動的に発見できない、興味深いスレッドが少し存在しする、流通量が多いグループのためにあります。

adapt

このエントリーは適応スコア付けを制御します。これが *t* だったら、デフォルトの適応スコア規則が使われます。ignore だったら、このグループでは適応スコア付けは行なわれません。もしリストだったら、そのリストが適応スコア規則として用いられます。もしそれが存在しないか *t* や ignore 以外の何かだったら、デフォルトの適応スコア規則が使われます。たいていのグループで適応スコア付けを使いたいのであれば、gnus-use-adaptive-scorint を *t* に設定し、適応スコア付けをしたくないグループに (adapt ignore) を挿入すればよいでしょう。少しのグループでだけ適応スコアを行ないたいのであれば、gnus-use-adaptive-scoring を nil に設定し、それを行ないたいグループのスコアファイルに (adaptive *t*) を挿入しましょう。

adapt-file

すべての適応スコア・エントリーは、このエントリーによって名づけられたファイルに入ります。さらにそれはグループに入るときにも適用されます。このアトムは、多くのグループで同じ適応スコアファイルを用いることによって、複数のグループに一度に適応スコアを付けたいときに便利です。

local

このエントリーの値は (var value) の形式の対のリストでなければなりません。それぞれの var は現在の概略バッファでバッファローカルになり、指定された値 (value) に設定されます。これは少し風変わりですが、フックがあまり好きでないならば、いくつかのグループで変数を設定するのに便利な方法です。value は評価されないことに注意して下さい。

7.5 スコアファイルの編集

普通はすべてのスコア命令を概略バッファから発行しますが、手でそれらを編集したくなることもあるかもしれないので、そのためのモードを用意しています。

それは以下に列挙する命令を使えるように、少しカスタマイズしただけの emacs-lisp モードです:

- C-c C-c* あなたが行なった変更を保存して概略バッファに戻ります (`gnus-score-edit-done`)。
- C-c C-d* 現在の日付を数値の様式で挿入します (`gnus-score-edit-insert-date`)。これはどのようなものだろうと考えているのなら、これは本当に日の数値です。
- C-c C-p* 適応スコアファイルは整形されずに保存されます。もしこれらのファイルの一つを読むつもりなら、まず *pretty print* (整形して印字) したいでしょう。この命令 (`gnus-score-pretty-print`) がそれを行ないます。

このモードを使うには *M-x gnus-score-mode* とタイプして下さい。

`gnus-score-menu-hook` がスコアモードのバッファで実行されます。

概略バッファでは、*V f*、*V e* および *V t* のような命令でスコアファイルの編集を始めることができます。

7.6 適応スコア付け

これらのスコア付けはあなたを憂鬱にさせてしまうかもしれないので、Gnus にはこれらをすべて自動的に——まるで魔法でも使ったように作成する方法があります。いやむしろ、人工無能によって、という方が正確かな。

記事を読んだとき、記事に既読の印を付けたとき、あるいは記事を削除したときに、その印を残しておいて下さい。グループから出るときに、Gnus はそれらの印の辺りを嗅ぎ回り、何の印を見つけたかに応じてスコア要素を追加します。この機能は `gnus-use-adaptive-scoring` を *t* か (*line*) に設定すると有効になります。もしスコアを、表題に現れる個別の単語をもとに適応させたければ、この変数を (*word*) に設定して下さい。両方の適応方法を使いたければ、この変数を (*word line*) に設定して下さい。

スコア付けの処理を完全に制御するために `gnus-default-adaptive-score-alist` 変数をカスタマイズして下さい。例えば、こんな感じになるでしょう:

```
(setq gnus-default-adaptive-score-alist
      '( (gnus-unread-mark)
          (gnus-ticked-mark (from 4))
          (gnus-dormant-mark (from 5))
          (gnus-del-mark (from -4) (subject -1))
          (gnus-read-mark (from 4) (subject 2))
          (gnus-expirable-mark (from -1) (subject -1))
          (gnus-killed-mark (from -1) (subject -3))
          (gnus-kill-file-mark)
          (gnus-ancient-mark)
          (gnus-low-score-mark)
          (gnus-catchup-mark (from -1) (subject -1))))
```

ご覧のように、この連想リストの各要素は、キーとして印 (変数名か「実際の」印すなわち文字のいずれか) を持ちます。このキーの後には任意の数のヘッダー/スコアの組が続きます。もしそのキーの後にヘッダー/スコアの組が一つもなければ、そのキーが記事の印として付いている記事に対しては適応型スコア付けは行なわれません。例えば上記の例では、`gnus-unread-mark` が付いている記事は、適応型スコア付けのエントリーを持ちません。

各記事は一つしか印を持ち得ないので、それぞれの記事にはこれらの規則のうちの一つだけが適用されます。

gnus-del-mark を例に取りましょう—この連想リストでの意味は、この印 (すなわち ‘e’ の印) が付いている記事はすべて、From ヘッダーをもとに -4 下げられ Subject で -1 下げられるスコア・エントリーが追加されます。これをあなたの偏見に合わせて変更して下さい。

もし 10 個の記事に同じ subject で gnus-del-mark の印が付いていたとすると、この印に対する規則は十回適用されます。それはつまり、その subject は -1 の十倍のスコアを得ます。その値は、私が大きく誤解していないかぎり -10 のはずです。

もし自動期限切れ消去 (メール) グループ (see Section 6.3.9 [Expiring Mail], page 163) があれば、既読記事にはすべて ‘E’ 印が付けられます。これはおそらく適応型スコア付けをちょっとばかりやりにくくするので、自動期限切れ消去と適応型スコア付けは、一緒には現実にはあまりうまくやっ

ていけません。

スコアを付けられるヘッダーには from, subject, message-id, references, xref, lines, chars および date があります。さらに followup にもスコア付けができて、これは現在の記事の Message-ID を使って References ヘッダーに合致、すなわちこれに続くスレッドに合致する適応型スコア・エントリーを作成します。

この機構を使うならば、ときどき記事を既読にしてしまう小さな変更を避けるために、スコアファイルの mark アトムを何か小さい値—ひょっとすると -300 くらいに設定しておくべきです。

適応型スコア付けを一週間かそこら使ってくると、Gnus はそれ相応に調教され、あなたが何も言わなくても、あなたの好きな投稿者を強調し、あまり好きではない投稿者を消去するようになるはずです。

どのグループにおいて適応型スコア付けを作動させるかは、スコアファイル (see Section 7.4 [Score File Format], page 232) を使うことによって制御できます。またこれを使って、違ったグループに対して違った規則を使うようにもできます。

適応型スコア・エントリーは、グループ名に gnus-adaptive-file-suffix を付加した名前のファイルに入れます。デフォルトは ‘ADAPT’ です。

適応型スコアファイルは巨大になり得るので、人の手で編集されることは想定されていません。gnus-adaptive-pretty-print が nil (デフォルト) であると、それらのファイルは人に読めるような形式では書かれません。

適応型スコア付けを行なうときは、部分文字列一致やファジーな一致を行なった方が、おそらくほとんどの場合において良い結果が得られるでしょう。しかし、ヘッダーの一致する部分が短い場合、意図に反する動作をする可能性が大きいので、gnus-score-exact-adapt-limit より短い長さしか一致しない場合は完全一致が行なわれます。この変数が nil であれば、この問題が起こらないように常に完全一致が行なわれます。

上で述べたように、個別の単語でもヘッダー全体でも適応を行なうことができます。単語で適応を行なう場合には、それぞれの単語の事例が、ある印にどんなスコアを加えるかを、gnus-default-adaptive-word-score-alist 変数によって指定します。

```
(setq gnus-default-adaptive-word-score-alist
      '( (,gnus-read-mark . 30)
          (,gnus-catchup-mark . -10)
          (,gnus-killed-mark . -20)
          (,gnus-del-mark . -15)))
```

これがデフォルト値です。単語での適応を有効にすると、gnus-read-mark の印が付いている記事の表題に現れるすべての単語が、スコアに 30 点追加するというスコア規則を生み出します。

gnus-default-ignored-adaptive-words のリストに現れる単語は無視されます。無視したい単語を追加したいときは、この変数ではなく gnus-ignored-adaptive-words リストの方を使って下さい。

短い単語では適応型スコア付けを作動させるべきではないと思う人もいるでしょう。もしそうなら `gnus-adaptive-word-length-limit` に整数を設定することができ、この数値より短い単語は無視されます。この変数のデフォルトは `nil` です。

スコア付けが行なわれるとき、`gnus-adaptive-word-syntax-table` が実際に使われるシンタックステーブルです。これは標準シンタックステーブルと似ていますが、数字を単語の構成要素ではない文字だと認識します。

もし `gnus-adaptive-word-minimum` に数値が設定されていると、単語適応型スコア付け処理において、記事のスコアがこの数値よりも小さくなることはありません。デフォルトは `nil` です。

`gnus-acaptive-word-no-group-words` が `t` に設定されていると、Gnus はグループ名のすべての語について、単語適応型スコア付けをしません。ほとんどの表題が `'emacs'` という語を含んでいる `'comp.editors.emacs'` のようなグループで便利です。

この機構をしばらく使ってみた後で、規則を解析することによってあなたがどんな単語が好きでどんな単語が嫌いかを診断する `gnus-psychoanalyze-user` (利用者精神分析命令) を書いてみると良いかもしれません。いや、良くないかな。

単語適応型スコア付けは高度に実験的なものなので、将来変更されるであろうことは心に留めておいて下さい。第一印象では、これは現状ではまったく使い物にならないように思えます。これをもっと使えるようにするためには、(より厳密な統計的手法を巻き込んで) さらなる作業が行なわれる必要があるでしょう。

7.7 ホームスコアファイル

新しいスコアファイルエントリーが入れられるスコアファイルは、ホームスコアファイル *home score file* と呼ばれます。これは通常 (デフォルトで) そのグループ自身のためのスコアファイルになります。例えば `'gnu.emacs.gnus'` 用のホームスコアファイルは `'gnu.emacs.gnus.SCORE'` です。

しかしながら、これはあなたのお望みではないかもしれません。たくさんのグループの間で共通のホームスコアファイルを共有することはしばしば便利です—例えばすべての `'emacs'` グループが、これと同じホームスコアファイルを使うことができます。

これを制御する変数が `gnus-home-score-file` です。これは以下の値を取り得ます:

1. 文字列。この場合、このファイルがすべてのグループでホームスコアファイルとして使用されます。
2. 関数。この関数の結果がホームスコアファイルとして使用されます。この関数はグループの名前を引数として呼び出されます。
3. リスト。このリストの要素は以下の値を取り得ます:
 1. `(regexp file-name)`。 `regexp` がグループ名に合致すると、 `file-name` がホームスコアファイルとして使用されます。
 2. 関数。この関数が `nil` 以外を返せば、その戻り値がホームスコアファイルとして使用されます。グループ名が引数として関数に渡されます。
 3. 文字列。この文字列をホームスコアファイルとして使用します。

このリストは、合致するものを探すために先頭から終りに向かってなぞられます。

というわけで、単一のスコアファイルだけを使いたい場合は、以下のようにすれば良いでしょう:

```
(setq gnus-home-score-file
      "my-total-score-file.SCORE")
```

すべての 'gnu' グループに対して 'gnu.SCORE' を、すべての 'rec' グループに対して 'rec.SCORE' (等々) を使いたい場合は、このように設定することができます:

```
(setq gnus-home-score-file
      'gnus-hierarchical-home-score-file)
```

これは利用者の便宜のために、あらかじめ提供されている関数です。他に以下の関数があります:

`gnus-current-home-score-file`

「現在の」標準スコアファイルを返します。これはスコア命令群に「最深」の合致するスコアファイルにエントリーを加えさせます。

'emacs' グループ用に一つのスコアファイルを、それとは別のものを 'comp' グループ用に用意する一方、他のすべてのグループではそれぞれ独自のスコアファイルを使うようにしたいなら、こんな設定で良いでしょう:

```
(setq gnus-home-score-file
      ;; 正規表現 "\\..emacs" に合致するすべてのグループ
      '(("\\..emacs" "emacs.SCORE")
        ;; すべての comp グループを単一のスコアファイルで
        ("^comp" "comp.SCORE")))
```

`gnus-home-adapt-file` は `gnus-home-score-file` とまったく同じやり方で動作しますが、代わりにこれで、何をホーム適応スコアファイルにするかを指定します。すべての新しい適応ファイルエントリーは、この変数で指定されるファイルに入れられ、同じ文法を使うことができます。

`gnus-home-score-file` と `gnus-home-adapt-file` を使うことに加えて、グループパラメーター (see Section 2.10 [Group Parameters], page 23) とトピックパラメーター (see Section 2.16.5 [Topic Parameters], page 37) を使っても、ほぼ同様のことを成し遂げることができます。グループ、トピックパラメーターはこの変数よりも優先されます。

7.8 自分自身へのフォローアップ

Gnus は現在のバッファから Message-ID ヘッダーを見つけ出すための二つの命令を提供します。そして Gnus は、他の記事の References ヘッダーにあるこの Message-ID を使ってスコアを付けるためのスコア規則を追加します。これは事実上、現在のバッファにある記事に返答したすべての記事のスコアを増加させます。これは、あなたが言ったことに人々が答えたら、それに容易に気付かせてもらいたいときに、とても便利です。

`gnus-score-followup-article`

これは、あなた自身の記事に直接フォローアップした記事にスコアを加えます。

`gnus-score-followup-thread`

これは、あなたの記事より「下」のスレッドに現れるすべての記事にスコアを加えます。

これら二つの関数は、本来どちらも `message-sent-hook` のようなフックの中で、例えばこのように使うためのものです:

```
(add-hook 'message-sent-hook 'gnus-score-followup-thread)
```

自分の Message-ID をじっくりと眺めてみると、はじめの二～三文字は常に同じであることに気が付くでしょう。以下の二つは私のものです:

```
<x6u3u47icf.fsf@eyesore.no>
<x6sp9o7ibw.fsf@eyesore.no>
```

したがって、このマシンでは 'x6' で「私」かどうかを見分けることができます。これは使えます—以下の規則は、私自身へのすべてのフォローアップのスコアを上げるでしょう:

```
( "references"
  ("<x6[0-9a-z]+\.\fsf\(_-\)\)?@.*eyesore\\.no>"
   1000 nil r))
```

「あなたの」が最初の二文字になるか最初の三文字になるかは、システムに依存します。

7.9 他のヘッダーにスコアを付ける

Gnus が「伝統的」なヘッダー — ‘From’, ‘Subject’ など — にスコアを付けるのはとても速いです。ですが、他のヘッダーにスコアを付けるには head のスコアのための規則を書く必要があります、それは合致を探すために Gnus が毎回バックエンドから単独の記事を取り寄せなければならないことを意味します。これは大きなグループでは長い時間がかかります。

さて、これに関してニュースグループのためにできることは多くはありませんが、メールグループのためにはより優れた手段があります。Section 3.1.2 [To From Newsgroups], page 46 の章でこの機構がどう働くかが詳しく説明されていますが、ここではどうしたら nnnml で ‘To’ と ‘Cc’ ヘッダーにスコアを付けることができるかの調理の例を挙げましょう。

以下を ‘~/gnus.el’ ファイルに置いて下さい。

```
(setq gnus-extra-headers '(To Cc Newsgroups Keywords)
      nnnml-extra-headers gnus-extra-headers)
```

Gnus を再起動して、*M-x nnnml-generate-nov-databases* コマンドで nnnml の overview ファイルを作り直して下さい。たくさんのメールを持っていると、これには長い時間がかかるでしょう。

そして *I e s p To RET <your name> RET* のようにすると、‘To’ と ‘Cc’ ヘッダーに “extra headers” としてスコアを付けることができます。

わかったかな？ 簡単だね。

ヘッダーまたは本文の遅いスコア付けは、変数 `gnus-inhibit-slow-scoring` を設定することによって禁止することができます。もし `gnus-inhibit-slow-scoring` が正規表現だったら、グループがその正規表現に合致する場合に遅いスコア付けが禁止されます。それが `t` だったら、すべてのグループで遅いスコア付けが禁止されます。

7.10 スコア付けの奥義

クロスポスト

クロスポストのスコアを低くしたければ、合致させるべき行は Xref ヘッダーです。

```
("xref" (" talk.politics.misc:" -1000))
```

複数のクロスポスト

ある数、例えば三つ以上のグループにクロスポストされている記事のスコアを低くしたければ、

```
("xref"
 ("[:\n]+:[0-9]+ +[:\n]+:[0-9]+ +[:\n]+:[0-9]+"
  -1000 nil r))
```

本文への合致

これは一般的にはあまり良い考えではありません——とても長い時間がかかるからです。実際 Gnus は、それぞれの記事を個別にサーバーから取得してこななければならないのです。でも、とにかくあなたはやりたいのでしょうね。合致させるキーは三つ (Head, Body, All) あるのですが、それぞれのスコアファイルで一つを選んで、それに固執すべ

きです。もし二つを使うと、それぞれの記事は 二回 取得されてしまいます。もし Head でちょっとだけ、Body でもちょっとだけ合致させたい、というのであれば、素直に All を使って全部合致させて下さい。

既読として印を付ける

ある一定の値より低いスコアを持つ記事には、おそらく既読の印を付けてしまいたくないのでしょう。これは 'all.SCORE' ファイルに以下のものを入れておくことによって、最も簡単に実現できます。

```
((mark -100))
```

同様のことを expunge で行なうことを考えても良いでしょう。

否定文字クラス

もし [^abcd]* みたいなものを指定すると、期待外れの結果で終わるかもしれません。これは改行文字にも合致してしまうので、えーと、未知のものまで引きずり出してしまってもかもしれません。代わりに [^abcd\n]* を使いましょう。

7.11 逆スコア

もし、表題ヘッダーに 'Sex with Emacs' という文字がある記事だけを残して、その他の記事すべてを消去してしまいたければ、スコアファイルに以下のようなものを入れることができます:

```
(("subject"
  ("Sex with Emacs" 2))
 (mark 1)
 (expunge 1))
```

これで 'Sex with Emacs' に合致するすべての記事のスコアが上がって、残りの記事には既読の印が付き、おまけにそれらは消去されるでしょう。

7.12 グローバルスコアファイル

間違いなく、他のニュースリーダーは「グローバル削除ファイル (global kill file)」を持っています。それらは普通、すべてのグループに適用される、利用者のホームディレクトリーに格納されている一つの削除ファイル以上の何物でもありません。ふふん! つまらない、低能なニュースリーダーだね。

私がここで話しているのはグローバルスコアファイルです。全世界の、あらゆる地域の利用者のスコアファイル。それはすべての国家を巨大な一つの幸せなスコアファイル同盟に団結させる! スコア天使! 新しい、でもテストされていない!

他人のスコアファイルを使うためにしなければならないのは、gnus-global-score-files 変数を設定することがすべてです。それぞれのスコアファイルにつき一つ、またはそれぞれのスコアファイルディレクトリーにつき一つのエントリーになります。Gnus はどのスコアファイルをどのグループに使うのが適切であるかを自分で決定します。

例えばスコアファイル '/ftp@ftp.gnus.org:/pub/larsi/ding/score/soc.motss.SCORE' および '/ftp@ftp.some-where:/pub/score' ディレクトリーにあるすべてのスコアファイルを使いたければ、このように設定して下さい:

```
(setq gnus-global-score-files
      '("/ftp@ftp.gnus.org:/pub/larsi/ding/score/soc.motss.SCORE"
        "/ftp@ftp.some-where:/pub/score/"))
```

単純でしょう? ディレクトリー名は '/' で終わらなくてはなりません。これらのディレクトリーは、一般に Gnus を使う一回の期間中に一回だけしか読み込まれません。もし遠隔ディレクトリーを手動

で再読み込みすることが必要だと思ったら、`gnus-score-search-global-directories` 命令を使って下さい。

ただし、現時点ではこのオプションを使うと、グループに入るのがいくらか遅くなります。(つまり—かなり、ですけど。)

他の人たちに使ってもらうためにスコアファイルを維持管理したくなったら、単にあなたのスコアファイルを匿名 `ftp` に置いて、世界に公表して下さい。逆行司会者になりましょう! その後に続いて間違いなく起こる逆行司会者戦争、すなわち人々の共感を勝ち取るための戦いに参加して、彼らのスコアファイルに偽りの前提を使わせるように誘導するのです! やったね! これでネットは救われる!

‘retro-’を「逆行」と訳しました。日本では「レトロ」を「古き善き時代の」のような肯定的な意味で使うことが少なくないのですが、ここでは本来の「時代に逆行した」「使えねー」のような意味で使っています。

以下に、逆行司会者なりたがりのための秘技をいくつか、即席で述べます:

- 非常に多くの場所にクロスポストされている記事は間違いなく屑である。
- 一個の不適切な記事を減点するには、Message-ID で減点する。
- 特に素晴らしい投稿者たちは永続的な基準で加算して良い。
- そのグループの憲章を無視した投稿を頻繁に繰り返す投稿者は、絶滅させてしまって差し支えない。
- `mark` と `expunge` アトムを設定し、汚らしい記事を完全に葬り去る。
- 期限切れ消去のスコア・エントリーを使って、ファイルの大きさを小さく抑える。もっとも、サイトによって古い記事を長期間保存するように、おそらく長い期限切れ消去の期間を取るでしょうけれども。

... 果たして他のニュースリーダーは、将来グローバルスコアファイルをサポートするのでしょうか? うふふ。そう、どう考えてみたって、Blue Wave や `xrn` や `1stReader` とかいったニュースリーダーは、スコアをサポートするべきですね。今は固唾を飲んで見守ることにしましょうか?

7.13 消去ファイル

Gnus はまだ、あのうざったい古い消去ファイルをサポートしています。実際消去ファイルの項目はもう消してもよいのですが、それは Daniel Quinlan がスコアファイルを考え出す前に私が書いたものなので、そのコードはまだ残してあるのです。

要するに、消去処理はスコア処理よりもかなり (私に言わせれば ものすごく) 遅いので、あなたの消去ファイルはスコアファイルに書き換えた方が良いかもしれません。

いずれにせよ、消去ファイルは普通の `emacs-lisp` ファイルです。このファイルの中にはどんな形式でも入れることができます。つまり消去ファイルをグループに入ったときに実行する一種の原始的なフック関数のように使うことができます。まあそれがあまりいい方法ではないとしてもね。

通常の消去ファイルは以下ようになります:

```
(gnus-kill "From" "Lars Ingebrigtsen")
(gnus-kill "Subject" "ding")
(gnus-expunge "X")
```

これは私が書いたすべての記事に既読の印を付け、概略バッファから印の付いた記事を削除します。とっても便利です。あなたもそう思うでしょ。

他のプログラムではまったく違う消去ファイルの構文を使っています。Gnus は `rn` の消去ファイルらしきものに出会うと、何とかそれを解釈しようとします。

GNUS 消去ファイルを編集するための二つの概略バッファ関数があります:

M-k このグループの消去ファイルを編集します (`gnus-summary-edit-local-kill`)。

M-K 一般消去ファイルを編集します (`gnus-summary-edit-global-kill`)。

消去ファイルを編集する二つのグループモード関数があります:

M-k このグループの消去ファイルを編集します (`gnus-group-edit-local-kill`)。

M-K 一般消去ファイルを編集します (`gnus-group-edit-global-kill`)。

消去ファイル変数:

`gnus-kill-file-name`

‘`soc.motss`’グループ用の消去ファイルは通常 ‘`soc.motss.KILL`’ という名前です。このファイル名を得るためにグループ名に付加される接尾語は、`gnus-kill-file-name` 変数で与えられます。「グローバル」消去ファイルは (スコアファイルの意味での「グローバル」じゃないよ、もちろん) 単に ‘`KILL`’ という名前です。

`gnus-kill-save-kill-file`

この変数が `nil` 以外であれば、Gnus は処理の後に消去ファイルを保存します。これは期限切れ消去を行なう消去を使っているときに必要です。

`gnus-apply-kill-hook`

グループに消去ファイルを適用するために呼び出されるフック。これはデフォルトでは (`gnus-apply-kill-file`) です。同じグループのためのスコアファイルがある場合に消去ファイルが無視したければ、このフックを (`gnus-apply-kill-file-unless-scored`) に設定して下さい。消去ファイル进行处理させたくなければ、この変数を `nil` に設定して下さい。

`gnus-kill-file-mode-hook`

消去ファイルモードのパッファー内で呼び出されるフック。

7.14 消去ファイルの変換

あなたが古い消去ファイルをどっさり持っているのであれば、それらをスコアファイルに変換したくなるでしょう。もしそれらが「普通の」ものであれば、‘`gnus-kill-to-score.el`’ パッケージを使うことができます。そうでなければ、手で変換しなければならないでしょう。

消去ファイルからスコアファイルへの変換パッケージは、標準では Gnus には含まれていません。<http://www.stud.ifi.uio.no/~larsi/ding-various/gnus-kill-to-score.el> から入手することができます。

あなたの消去ファイルが非常に複雑なのであれば—それに `gnus-kill` 形式以外のものがたくさん含まれているのなら、それらを手で変換しなければならないでしょう。あるいは、単そのままにしておいて下さい。Gnus は以前と同様にそれらを使ってくれるでしょう。

7.15 上級スコア付け

表題や From ヘッダーにスコアを付けるのは十分素敵ですが、本当に興味があるのが、特定の表題に関してある人が言っていることだけだった場合はどうすれば良いでしょう? もしくは、A さんが B さんにフォローアップしているときは彼女が言っていることを読みたくないけれど、C さんにフォローアップしているときは何を言っているかを知りたいという場合は?

上級スコア規則を使えば、どんな複雑なスコアのパターンでも作成することができます。

7.15.1 上級スコア付け構文

普通のスコア規則では、規則の最初の要素が文字列です。上級スコア付け規則では、最初の要素はリストです。二番目の要素は、最初の要素が `nil` でない値として評価されたときに適用されるスコアです。

これらのリストは三つの論理演算子、一つのリダイレクト演算子 (訳注: 本文では間接演算子と表記されています)、および様々な合致演算子で構成することができます。

論理演算子:

```
&
and      この論理演算子は、それぞれの引数を順に評価して、ある評価の結果が false になっ
          たら停止します。すべての引数が true の値に評価された場合、この演算子は true を
          返します。

|
or       この論理演算子は、それぞれの引数を順に評価して、ある評価の結果が true になっ
          たら停止します。どの引数も true でなかったら、この演算子は false を返します。

!
not
æ,A,    この論理演算子はたった一つの引数を取ります。その引数の値の論理否定を返します。
```

スコア付けされている現在の記事の先祖たちに対して、その引数群を適用する「間接演算子」があります。例えば `1-` は、現在の記事の親にもスコア規則を適用します。`2-` は現在の記事の祖父母にスコア規則を適用します。代わりに `^^` を書くこともでき、`^` (caret==キャレット) の数でどのくらい祖先の記事までさかのぼるかを示します。

最後に合致演算子があります。これらが本当の仕事をするものです。合致演算子はヘッダー名の文字列で、その後に合致と合致の型が続きます。典型的な合致演算子は `('("form" "Lars Ingebrigtsen" s)')` のようなものです。ヘッダー名は単純なスコア付けをするときのものと同じで、合致の型も同じです。

7.15.2 上級スコア付けの例

以下の例はスコアファイルの規則であることに注意して下さい。それらを使って完璧なスコアファイルを作るには、別の括弧の組でそれらを囲んで下さい。

Lars が Gnus に関して話をしているときに、彼によって書かれた記事のスコアを増やしたいとしましょう:

```
((&
  ("from" "Lars Ingebrigtsen")
  ("subject" "Gnus"))
 1000)
```

ふん、簡単すぎるかな?

彼が長い記事を書くとき、時々何か素敵なことを言います:

```
((&
  ("from" "Lars Ingebrigtsen")
  (|
    ("subject" "Gnus")
    ("lines" 100 >)))
 1000)
```

しかし、彼が Reig Eigil Logge によって書かれたものに反応しているときは、彼が書いたものを読みたいありません:

```
((&
  ("from" "Lars Ingebrigtsen")
  (1- ("from" "Reig Eigil Logge"))))
-100000)
```

Redmondo が消えた靴下について書いたときにフォローアップしたすべての人のスコアが上げられますが、それは彼らが白い靴下について語っているときのみです。しかし Lars が靴下について話をしているときは、たいていあまりおもしろくありません:

```
((&
  (1-
    (&
      ("from" "redmondo@.*no" r)
      ("body" "disappearing.*socks" t)))
    (! ("from" "Lars Ingebrigtsen"))
    ("body" "white.*socks"))
  1000)
```

大量の記事が流れているグループを読んでいて、返答にしか興味が無いとしましょう。そういう場合にやることは、"Re:"、"Fw:" または "Fwd:" で始まる表題を持っていないすべての記事のスコアを下げて、返答の印で始まる表題を持っている記事のすべての親のスコアを上げることです。

```
(((! ("subject" "re:\\|fwd?:" r))
  -200)
  ((1- ("subject" "re:\\|fwd?:" r))
    200)
```

可能性は無限大です。

7.15.3 上級スコアのちょっとした秘訣

& と | 論理演算子は、無意味な処理を迂回する論理 (原典: short-circuit logic) に基づいて動作します。すなわち、その処理の結果が明らかになった時点で、引数を処理することを止めます。例えば & の引数の一つが false に評価されると、残りの引数を評価する意味がありませんから。これは遅い合致 ('body' や 'header') を最後に持ってきて、速い合致 ('from' や 'subject') を最初に持ってくるべきであることを示唆します。

間接演算子 (1- など) は、それらの引数をスレッドの一世代前に作用させます。次のようなことをすると:

```
...
(1-
  (1-
    ("from" "lars")))
...
```

これは「現在の記事の祖父母の from ヘッダーでスコアを付ける」ということを意味します。間接演算子の処理はとても速いのですが、以下のやりの方が:

```
(1-
  (&
    ("from" "Lars"))
```



```
("subject" "Gnus"))))
```

次のものより良いです:

```
(&
  (1- ("from" "Lars"))
  (1- ("subject" "Gnus"))))
```

7.16 スコアを減衰させる

スコアは（特に適応スコアを使っていると）際限無く膨れ上がる傾向があることに気が付くでしょう。スコアが大きくなりすぎると、それらはすべての意味を失います—それらは単に最大値に達してしまうので、意味のある方法で使うことは難しくなります。

Gnus はこの問題の解決を助けるためにスコアを減衰させる機構を提供します。スコアファイルが読み込まれて、gnus-decay-scores が nil ではないと、Gnus はスコアファイルを減衰機構を通して、すべての永続でないスコア規則のスコアを下げます。もし gnus-decay-scores が正規表現だったら、それに合致するスコアファイルだけが扱われます。例えば *adaptive* スコアファイルだけを減衰させるには、それを ‘\\ADAPT\\’ に設定すれば良いでしょう。減衰そのものは gnus-decay-score-function 関数によって実行され、デフォルトは gnus-decay-score です。以下はその関数の定義です:

```
(defun gnus-decay-score (score)
  "Decay SCORE according to 'gnus-score-decay-constant'
  and 'gnus-score-decay-scale'."
  (let ((n (- score
              (* (if (< score 0) -1 1)
                  (min (abs score)
                        (max gnus-score-decay-constant
                             (* (abs score)
                                gnus-score-decay-scale)))))))
    (if (and (featurep 'xemacs)
             ;; XEmacs' floor can handle only the floating point
             ;; number below the half of the maximum integer.
             (> (abs n) (lsh -1 -2)))
        (string-to-number
         (car (split-string (number-to-string n) "\\.")))
        (floor n))))
```

gnus-score-decay-constant はデフォルトで 3、gnus-score-decay-scale は 0.05 です。これは以下のようなことを引き起こします:

1. この関数が呼ばれたときに -3 から 3 の間のスコアは 0 に設定されます。
2. 3 から 60 までの間の大きさのスコアは 3 減らされます。
3. 60 より大きいスコアはスコアの 5% が減らされます。

もしこの減衰関数がお気に召さないなら、自分用の関数を書いて下さい。それは減衰させるべきスコアを唯一の引数として呼ばれ、新しいスコアを整数で返さなければなりません。

Gnus は一日に一回スコアを減衰させようとします。例えば Gnus を四日間走らせていないと、Gnus はスコアを四回減衰させます。

8 いろいろ

8.1 プロセス/接頭引数

多くの関数、その中でも記事の移動、デコード、保存をするための関数は、「プロセス/接頭引数の習慣」として知られているものを使います。

これは、利用者がどの記事に命令を実行したいかを見つけるための方法です。

それはこのような感じです:

数値接頭引数が *N* だったら、現在の記事を含めた次の *N* 個の記事に対して作業を実行します。もし数値接頭引数が負だったら、現在の記事を含めた前の *N* 個の記事に対して作業を実行します。

`transient-mark-mode` が `nil` ではなく、リージョンが設定されていたら、リージョンにあるすべての記事で作業が行なわれます。

数値接頭引数が無くても、いくつかの記事はプロセス印が付いている場合には、プロセス印が付いている記事で作業が実行されます。

数値接頭引数やプロセス印の付いている記事が無い場合は、現在の記事でだけ作業を実行します。

これは実際とても単純なのですが、びっくりされないためにも、はっきりさせておく必要があります。

プロセス印に反応するコマンドは、現在プロセス印が付いている記事のリストをスタックに積んで、記事のすべてのプロセス印を消去します。前回の設定を `M P y` で復旧させることができます (see Section 3.7.6 [Setting Process Marks], page 60)。

多くの人々をぎょっとさせ、恐がらせると思われることの一つは、例えば `3 d` が、本当に `d d d` と同じことをすることです。それぞれの `d` (これは現在の記事に既読の印を付けます) は、デフォルトでは印を付けた後で次の未読記事に移動するので、`3 d` は概略バッファがどうなっているか、次の三つの未読記事を既読にします。動作をもっと分かりやすくするには、`gnus-summary-goto-unread` を `nil` に設定して下さい。

多くのコマンドはプロセス/接頭引数の習慣を使いません。それをしないすべてのコマンドは、このマニュアルで明記されています。そういうコマンドにプロセス/接頭引数の習慣を適用するには、`M-&` コマンドを使って下さい。例えば、そのグループのすべての記事を期限切れ消去可能として印を付けるには `M P b M-& E` とします。

8.2 利用者との相互作用

`gnus-novice-user`

この変数が `nil` でないのは、あなたは Usenet の世界の新参者が非常に慎重な人のどちらかだということです。これは本当に良いことです。何か危険なことをする前に、「本当にこれをしたいのですか?」というような質問を受けます。これはデフォルトでは `t` です。

`gnus-expert-user`

この変数が `nil` でないと、あなたが Gnus から質問を受けることは滅多に無いでしょう。これは単純に、あなたがどんな変なことをしても、何をしているかをわかっていると見なします。

`gnus-interactive-catchup`

`nil` でないと、グループに追いつく (`catchup`, 未読の記事を読んだことにしてしまう) 前に、確認を求めます。デフォルトで `t` です。

gnus-interactive-exit

Gnus を終了する前に確認を求めます。デフォルトで t です。

8.3 シンボルの接頭引数

非常に多くの Emacs の命令が (数値) 接頭引数に反応します。例えば `C-u 4 C-f` はポイントを 4 文字先に移動し、`C-u 900 I s s p` は 900 のスコア (永続、Subject、文字列の一部、という規則) を現在の記事に加えます。

これはすべて素敵で良いのですが、命令にもう少し追加の情報を与えたいときはどうすれば良いのでしょうか？ えーと、たいていの命令がしていることは「生の」接頭引数を何らかの特別な方法で解釈することです。例えば `C-u 0 C-x C-s` は、現在の記事を保存するときにバックアップファイルを作らないで欲しいことを意味します。でも、バックアップファイルを作らないで保存すると同時に、Emacs に閃光を放って、素敵な音楽を演奏して欲しいときはどうすれば良いのでしょうか？ それができなくても、あなたは申し分なく幸せですね (?)。

私はそうではありません。そこで、私は二つめの接頭引数「シンボル接頭引数」を加えました。接頭キーは `M-i` (gnus-symbolic-argument) で、次に押される文字が値です。いくらでも `M-i` 接頭語を積み重ねることができます。`M-i a C-M-u` は「`C-M-u` 命令にシンボル接頭引数 `a` を与える」ということです。`M-i a M-i b C-M-u` は「`C-M-u` 命令にシンボル接頭引数 `a b` を与える」ということです。趣旨はわかりましたね。

シンボル接頭引数を受け付けない命令にそれを打ち込んでも何も悪いことをしませんが、良いことも何もありません。現在のところ、あまり多くの関数がシンボル接頭引数を利用しているわけではありません。

Gnus がこれを実装しているやり方に興味があるなら、Section 10.8.7 [Extended Interactive], page 352 を見て下さい。

8.4 書法仕様変数

このマニュアルを通して、おそらく `gnus-group-line-format` または `gnus-summary-mode-line-format` のように呼ばれるたくさんの変数があることに気付いたでしょう。これらは Gnus が色々なバッファでどのように行を出力するかを制御します。非常にたくさんのものがあります。幸運なことに、それらはすべて同じ構文を使うので、あまり嫌な目には会わないでしょう。

書法仕様 (format) 指定の例です (グループバッファより):
`'%M%S%5y: %(%g%)\n'`。それは極めて醜く、たくさんのパーセント記号がいたるところにあります。

現在のところ Gnus は以下の書法仕様変数を使います: `gnus-group-line-format`, `gnus-summary-line-format`, `gnus-server-line-format`, `gnus-topic-line-format`, `gnus-group-mode-line-format`, `gnus-summary-mode-line-format`, `gnus-article-mode-line-format`, `gnus-server-mode-line-format`, および `gnus-summary-pick-line-format`。

これらすべての書法仕様変数は任意の elisp 式であることもできます。その場合、それらは要求される行に挿入するために eval (評価) されます。

Gnus は、あなたが自分用の書法仕様指定を作っているときに、手助けをする命令を備えています。`M-x gnus-update-format` は現在の式を eval し、当の仕様を更新し、行を生成するための Lisp 式を検査することができるバッファに移動します。

8.4.1 書法仕様の基本

それぞれの ‘%’ の要素は、当のバッファが作成されるときに何らかの文字列や他のもので置き換えられます。‘%5y’ は「‘y’ 指定を挿入して、5 文字の場所を得るために空白を詰め込みなさい」ということです。

普通の C や Emacs Lisp の書法仕様 (format) 文字列と同じように、‘%’ と書法仕様の型の文字の間の数値修飾子は、常に少なくともその長さになるように、出力に (空白文字などを) 「詰め込み」ます。‘%5y’ はその場所が常に (少なくとも) 5 文字の長さになるように、左に空白を詰め込みます。もし ‘%-5y’ とすれば、代わりに右側に詰め込みます。

特に広い幅の値に対して保護するために、その場所の長さを制限したいこともあるでしょう。そのために ‘%4,6y’ などと指定することができます。これは、その場所は決して 6 文字を超える幅にはならず、かつ 4 文字より少ない幅にもならないということです。

Gnus は ‘%&user-date;’ のような、いくつかの拡張様式指示もサポートします。

8.4.2 モード行書法仕様

モード行書法仕様変数 (例えば `gnus-summary-mode-line-format`) は、以下の二つの違い以外は、バッファ一行に適応した書法仕様変数 (see Section 8.4.1 [Formatting Basics], page 251) と同じ規則に従います:

1. 最後に改行 (‘\n’) があってはなりません。
2. 特別な ‘%b’ 仕様をバッファ名を表示するために使うことができます。えーと、実はそれは仕様ではないのです—‘%%’ というものは、Emacs が ‘%b’ を受け取って、そのモード行表示機能に「バッファ名を表示しなさい」と解釈させるために、単に書法仕様の処理系を無傷で通り抜けることができるように ‘%’ を囲う方法なのです。Emacs が理解するモード行指定の完全な一覧については、変数 `mode-line-format` の説明文を見て下さい。

8.4.3 上級書法仕様

表示された領域を後で何らかの方法で処理するのは、しばしば役に立ちます。詰め込み、制限、切り取り、および特定の値の抑制は、「チルダ修飾子」を使うことによって実現することができます。よくあるチルダ仕様は、‘%~(cut 3)~(ignore "0")y’ のように見えます。

これらは有効な修飾子です:

```
pad
pad-left   領域の左側に、要求された長さになるまで空白を詰め込みます。
pad-right  領域の右側に、要求された長さになるまで空白を詰め込みます。

max
max-left   指定された長さになるように、文字列の左側を切り取ります。
max-right  指定された長さになるように、文字列の右側を切り取ります。

cut
cut-left   指定された数の文字を左側から切り落とします。
cut-right  指定された数の文字を右側から切り落とします。

ignore     領域が指定された値と等しい (equal) ならば、空文字列を返します。
```

form ‘@’ 仕様が使われたときに、指定された式を領域の値として使います。

これは例です:

```
"~(form (current-time-string))@"
```

例を出してみましょう。概略モード行での ‘%o’ 仕様は ISO0861 様式の凝縮された日付 (‘19960809T230410’ のようなもの) を返します。これはとても発音しにくいので、世紀を表す数と時刻を削ぎ落として、6 文字の日付を残したいと思います。それは ‘%~(cut-left 2)~(max-right 6)~(pad 6)o’ となるでしょう。(切り落とし (cutting) は 最大幅の制限 (maxing) より先に行なわれるので、表示欄での見栄えを良くするために、日付が 6 文字より少なくなならないことを保証する詰め込み (padding) が必要になります。)

無視 (ignore) が最初に行なわれます。それから切り落とし (cutting)、次に最大幅の制限 (maxing)、そして最後の操作である詰め込み (padding) が行なわれます。

もしあなたが、これらの上級参照をたくさん使っているなら、Gnus がとても遅くなるのがわかるでしょう。これはあなたが行の外見に満足したときに *M-x gnus-compile* を実行することによって、格段に速度低下を減らすことができます。See Section 8.7 [Compilation], page 258.

8.4.4 利用者定義の指定

すべての仕様に、利用者が定義した ‘u’ で始まる述語を挿入することができます。書法仕様文字列の次の文字は、アルファベットでなければなりません。‘%u’ に続くアルファベットが ‘X’ だったら、Gnus は関数 *gnus-user-format-function-X* を呼びます。関数には単一の引数が与えられますが、その引数の意味は関数がどのバッファから呼ばれているかによって変わります。関数は文字列を返さなければなりません。それは他の述語によって生成される情報とまったく同じように、バッファに挿入されます。関数は意味の無い値と共に呼ばれる場合もあるので、その対策をしておくべきです。

Gnus は利用者定義仕様を拡張した ‘%u&foo;’ のような形式もサポートします。この場合は *gnus-user-format-function-foo* という関数を呼び出します。

新しい関数を定義しなくても、ほとんど同じことをチルダ修飾子 (see Section 8.4.3 [Advanced Formatting], page 251) を使って達成できるでしょう。例です:

```
‘%~(form (count-lines (point-min) (point)))@’
```

ここで与えられた式は評価されて現在の行番号をもたらし、それから挿入されます。

8.4.5 書法仕様フォント

すべての書法仕様変数によって共有される、ハイライト (強調表示) のための仕様があります。述語 ‘%(’ と 述語 ‘%)’ で囲まれたテキストには特別な *mouse-face* 属性が与えられ、そこにマウスのポインターを置いたときに (*gnus-mouse-face* によって) ハイライトされます。

述語 ‘%{’ と 述語 ‘%}’ で囲まれたテキストには、普通のフェースである *gnus-face-0* (デフォルトで *bold*) が与えられます。‘%{1’ を使うと、代わりに *gnus-face-1* が与えられ、以下同様です。欲しいだけたくさんのフェースを作ってください。同じことが *mouse-face* 仕様にも言えます。‘hello’ がマウスを置いたときに *gnus-mouse-face-3* でハイライトされるためには、‘%3(hello%)’ とすれば良いでしょう。

述語 ‘%《’ と 述語 ‘%》’ で囲まれたテキストでは、特別な *balloon-help* 属性が *gnus-balloon-face-0* に設定されます。‘%1《’ とすると、*gnus-balloon-face-1* が使われ、以下同様です。*gnus-balloon-face-** 変数は、文字列か文字列を返す関数を指すシンボルのどちらかでなければなりません。この属性が設定されているテキストの上をマウスが通過すると、吹き出しが現れて文字列を表示します。これの詳細情報は See section “Tooltips” in *The Emacs Editor*, (GNU

Emacs) または `balloon-help-mode` (XEmacs) の説明を参照して下さい。(技術的な理由のために、ギィメ (guillemets) はこの節では ‘《’ と ‘》’ で近似されました。)

訳注: guillemets (仏語) はギユメとも表記されます。実際に Gnus で有効なのは次の二つです:

```
(string (make-char 'latin-iso8859-1 43)) ;; 《
(string (make-char 'latin-iso8859-1 59)) ;; 》
```

日本語の「」に当たるもので、口頭表現を表記したり、強調したい単語を囲む、何かからの引用部分を囲む、書物等のタイトルを記す等様々に使われます。

これはグループバッファで使うことができる、もう一つの調理法です:

```
;; 三つのフェースを作ります。
(setq gnus-face-1 'bold)
(setq gnus-face-3 'italic)

;; 記事の数をボールドで緑のフェースにしたいので、
;; my-green-bold という新しいフェースを作ります。
(copy-face 'bold 'my-green-bold)
;; 色を設定します。
(set-face-foreground 'my-green-bold "ForestGreen")
(setq gnus-face-2 'my-green-bold)

;; 新しい特製の書法仕様を設定します。
(setq gnus-group-line-format
  "%M%S%3{%5y}%2[:%] %(1{%g%})\n")
```

あなたがこの案を使って、完全に読めなくて非常に下品な表示を作ることができることを確信しています。楽しんで下さい!

‘%(’ 指定 (やその類のもの) は、モード行変数ではまったく意味をなさないことに注意して下さい。

8.4.6 ポイントの移動

Gnus は通常ほとんどのバッファで、ポイントを各行のあらかじめ決められた場所に移動します。デフォルトでは、ポイントは行の最初のコロンの移動します。この振るまいは、三つの違う方法でカスタマイズすることができます。

また、コロンを行のどの場所にでも移動することができます。

コロンの位置にポイントを移動させるための関数を定義し直すことができます。その関数は `gnus-goto-colon` と呼ばれています。

でも、行にコロンを含めたくないならば、これを扱うためのおそらく最も手ごろな方法は ‘%*’ という述語を使うことです。あなたの行の書法仕様の定義に ‘%*’ を入れておけば、Gnus はそこにポイントを置きます。

8.4.7 整列

通常は、空白文字を詰め込んだり端を切り落とすことによって、文字列をディスプレイに並べることができます。でも大きさが違う異なる文字列を連結させる場合は、単に文字列を出力してしまうのがより手ごろであることが多いはずで、しかしそうするとその後続くテキストを並べるのに悩むことになります。

それを行なうために、Gnus は整列子 (tabulator) の仕様 ‘%=’ を備えています。これには二つの形式 *hard tabulators* および *soft tabulators* があります。

‘%50=’ は文字列が 50 桁までの場所を占めるように空白文字を詰め込みます。もし挿入するテキストの末端が 50 桁より後ろになる場合は何も挿入しません。これは穏やか (soft) な整列子です。

‘%-50=’ もまた、文字列が 50 桁までの場所を占めるように空白文字を詰め込みます。ですが、もし挿入するテキストの末端が 50 桁より後ろになる場合は、50 桁より後ろになる過剰なテキストは削除されます。これは厳密 (hard) な整列子です。

8.4.8 Wide Characters

多くの地域において、固定幅フォントは同じ幅の文字を持っています。しかしいくつかの地域、よく知られている東アジアの国々では、ラテン文字と幅の広い文字が混在して使われています。

整形において、Gnus は文字列が 10 個の文字の幅だとしたら、スクリーンでも 10 個分のラテン文字の幅になると仮定しますが、それは問題です。かの国々では、それは正しくありません。

それを救済するために、あなたは `gnus-use-correct-string-widths` を `t` に設定することができます。これはバッファの生成を遅くしますが、より美しい結果を得ることができるでしょう。デフォルト値は XEmacs では `t` ですが Emacs では `nil` です。(訳注: 日本語などを Emacs で表示する場合に、書体仕様によっては `t` にしないと概略バッファの表示が不揃いになることがあります。)

8.5 ウィンドウの配置

いえ、X に関することはありませんから、おとなしくして下さい。

もし `gnus-use-full-window` が `nil` でないと、Gnus はすべての他のウィンドウを消して、Emacs の画面全体を占有します。これはデフォルトで `t` です。

この変数を `nil` に設定してもそれなりに動作しますが、問題もあります。危険を覚悟の上で使ってください。

`gnus-buffer-configuration` はそれぞれの Gnus のバッファがどのくらいの空間を与えられるべきかを現します。これはこの変数の抜粋です:

```
((group (vertical 1.0 (group 1.0 point)
                        (if gnus-carpal (group-carpal 4))))
 (article (vertical 1.0 (summary 0.25 point)
                      (article 1.0))))
```

これは連想リストです。「キー」は何らかの動作を名付けるためのシンボルです。例えば、グループバッファを表示するときは、ウィンドウを設定するための関数は `group` をキーとして使います。使用可能な名前の完全な一覧は下に挙げられています。

「値」(すなわち「分割」) は、それぞれのバッファがどれくらいの空間を占めるべきかを指定します。`article` の分割を例に取ると

```
(article (vertical 1.0 (summary 0.25 point)
                  (article 1.0)))
```

この「分割」は概略バッファが画面の上の 25% を占めるべきで、それは記事バッファの上に配置されると言っています。お気づきのように、100% + 25% は実際は 125% です (ええ、皆さんの計算はこの様になったと思います。) しかし、特別な数値 1.0 は、残りのバッファが必要なものを取り去った後に、使用可能な残りの空間すべてを吸い取る、ということを合図するために使われます。1.0 の大きさを指定するバッファは、一つの分割につき一つだけでなくはなりません。

ポイント (カーソル) は省略可能な三つ目の要素、`point` を持つバッファに置かれます。`frame` 分割では、`frame-focus` タグが含まれている枝葉の分割を持っている副分割の、最後のもののフレー

ムがフォーカスを得ることになります (frame-focus タグは、それを含んでいる枝葉リストにおいて、point タグが無ければ三番目の、あれば四番目の要素になります)。

次はもっと複雑な例です:

```
(article (vertical 1.0 (group 4)
                      (summary 0.25 point)
                      (if gnus-carpal (summary-carpal 4))
                      (article 1.0)))
```

もし大きさの指定が浮動小数点数の代わりに整数だったなら、それは割合ではなく、どのくらい多くの行をバッファが占めるべきかを指定するために使われます。

もし「分割」が eval (評価) されるもののように見えるときは (正確に言うと—分割の car が関数か原始関数 (subr) であるときは)、この分割は eval されます。結果が nil でないなら、それは分割として用いられます。これは、gnus-carpal が nil であれば三つのバッファが、gnus-carpal が nil でないなら、四つのバッファが存在することになるということです。

まだ複雑ではないですって? それでは、大きさとしてこれを試してみてください:

```
(article (horizontal 1.0
                   (vertical 0.5
                           (group 1.0)
                           (gnus-carpal 4))
                   (vertical 1.0
                           (summary 0.25 point)
                           (summary-carpal 4)
                           (article 1.0))))
```

おっと。二つのバッファに謎の 100% タグが付いています。そして horizontal って何でしょう?

もし分割の一つの最初の要素が horizontal であったなら、Gnus はウィンドウを水平に分割し、二つのウィンドウを横に並べます。これらのそれぞれの小片の中では、あなたのやりたいことをすべて普通の流儀で行なうことができます。horizontal の後の数値は、この小片に画面のどれくらいの割合が与えられるかを指定します。

それぞれの分割では、100% のタグを持つ要素が 必ず 一つある必要があります。分割は決して正確ではないので、分割によって余ったすべての行を、このバッファが分捕ります。

もう少し形式的に、有効な分割がどのようなになるかの定義を挙げておきましょう:

```
split      = frame | horizontal | vertical | buffer | form
frame      = "(frame " size *split ")"
horizontal = "(horizontal " size *split ")"
vertical   = "(vertical " size *split ")"
buffer     = "(" buf-name " " size *["point"] *["frame-focus"] ")"
size       = number | frame-params
buf-name   = group | article | summary ...
```

制限として、frame は最も上位階層の分割としてしか現れることができないというものがあります。form は有効な分割を返す Emacs Lisp の式 (form) でなければなりません。それぞれの分割は完全に再帰的で、任意の数の vertical と horizontal 分割を含むことができます。

正しい大きさを見つけることは、少し複雑になります。どのウィンドウも gnus-window-min-height (デフォルトは 1) の文字の高さよりも小さくしてはならないし、少なくとも gnus-window-min-width (デフォルトは 1) の文字幅でなくてはなりません。Gnus は分割を適用する前にこれ

を強制しようと試みます。もし標準の Emacs のウィンドウの幅/高さ制限を使いたいなら、この二つの変数を nil にするだけで良いです。

Emacs の用語になじんでいないのなら、horizontal と vertical の分割は、期待したものと反対の動作をするでしょう。horizontal 分割の中のウィンドウは横に並んで表示され、vertical 分割の中のウィンドウは上下に表示されます。

ウィンドウの配置に関して実験をしてみたいのであれば、良い方法は分割を引数にして直接 `gnus-configure-frame` を呼ぶことです。これはバッファを分割するときにすべての実際の仕事をする関数です。下のものは五つのウィンドウを作るかなりばかげた設定です。二つをグループバッファに、三つを記事バッファのために充てます。(だから、ばかげていると言ったでしょ。) もし下の文を評価すると、普通の Gnus の経路を使わないで、すぐにそれがどのように見えるかの直観を得ることができます。満足するまでそれで遊んで、それから `gnus-add-configuration` を使って新しい作品をバッファ配置リストに加えて下さい。

```
(gnus-configure-frame
  '(horizontal 1.0
    (vertical 10
      (group 1.0)
      (article 0.3 point))
    (vertical 1.0
      (article 1.0)
      (horizontal 4
        (group 1.0)
        (article 10))))))
```

複数のフレームも欲しいかもしれません。問題ありません—frame 分割を使うだけです:

```
(gnus-configure-frame
  '(frame 1.0
    (vertical 1.0
      (summary 0.25 point frame-focus)
      (article 1.0))
    (vertical ((height . 5) (width . 15)
      (user-position . t)
      (left . -1) (top . 1))
      (picon 1.0))))
```

この分割の結果は、最初の (もしくは「主たる」) フレームに見慣れた概略/記事ウィンドウを配置し、小さな追加のフレームが picon を表示するために作られます。ご覧の通り、普通の最上位階層の 1.0 の定の代わりに、それぞれの追加の分割が大きさの指定として、フレームパラメーターの連想リストを持たなければなりません (see section “Frame Parameters” in *The GNU Emacs Lisp Reference Manual*). XEmacs では、フレームプロパティリストも使えます—例えば (height 5 width 15 left -1 top 1) がそのような plist です。gnus-buffer-configuration で使うことができるすべてのキーの一覧は、そのデフォルト値で見つけることができます。

キー message は gnus-group-mail および gnus-summary-mail-other-window の両方で使われることに注意して下さい。もし二つを区別するほうが望ましいなら、このような物を使うことができます:

```
(message (horizontal 1.0
  (vertical 1.0 (message 1.0 point))
  (vertical 0.24
```

```
(if (buffer-live-p gnus-summary-buffer)
    '(summary 0.5))
(group 1.0)))
```

良くある複数のフレーム分割の要望は、メールとニュースの作成には別のフレームを使い、元のフレームはそのままに残すというものです。これの達成には、以下のようなものでできます。

```
(message
  (frame 1.0
    (if (not (buffer-live-p gnus-summary-buffer))
        (car (cdr (assoc 'group gnus-buffer-configuration)))
        (car (cdr (assoc 'summary gnus-buffer-configuration))))
    (vertical ((user-position . t) (top . 1) (left . 1)
              (name . "Message"))
              (message 1.0 point)))))
```

訳注: これを高度に発展させたものが <http://www.jpl.org/elips/message-multiple-frames.el.gz> として入手できます。使い方はファイルの冒頭に書かれています。

変数 `gnus-buffer-configuration` はとても長く複雑なので、単一の設定の変更を簡単にするための関数があります: `gnus-add-configuration` です。例えば `article` の設定を変えたいのなら、次のようにできます:

```
(gnus-add-configuration
  '(article (vertical 1.0
                    (group 4)
                    (summary .25 point)
                    (article 1.0)))))
```

普通はこれらの `gnus-add-configuration` の呼び出しを `~/.gnus.el` ファイルに入れるか、何らかの起動時のフックに入れるでしょう—それらは Gnus が読み込まれた後で実行されなければなりません。

もし分割の設定で指定されたすべてのウィンドウがすでに見えているのであれば、Gnus はウィンドウの配置を変更しません。常に「正しい」ウィンドウ設定を強制したいのであれば、`gnus-always-force-window-configuration` を `nil` でない値に設定して下さい。

木表示 (see Section 3.24 [Tree Display], page 101) を使っていて、木ウィンドウが垂直方向に次の別のウィンドウで表示されるなら、ウィンドウの大きさが変更されることを避けるために `gnus-tree-minimize-window` をいじるのが良いでしょう。

8.5.1 ウィンドウ配置の例

- 左側を狭めてグループバッファに。右側を分割して概略バッファ (上 1/6) と記事バッファ (下) に。

```
(gnus-add-configuration
  '(article
    (horizontal 1.0
      (vertical 25 (group 1.0))
      (vertical 1.0
        (summary 0.16 point)
        (article 1.0)))))

(gnus-add-configuration
```

```
'(summary
  (horizontal 1.0
    (vertical 25 (group 1.0))
    (vertical 1.0 (summary 1.0 point)))))
```

8.6 フェースとフォント

かつてフォントとフェースをいじくのは非常に難しかったのですが、今日では非常に簡単です。単に *M-x customize-face* とやって、変えたいフェースを選び出して、標準のカスタマイズインターフェースを使って変更することができます。

8.7 コンパイル

あの行書法仕様指定変数を覚えていますか? `gnus-summary-line-format`, `gnus-group-line-format` などなどです。さて、Gnus はこれらの変数が何であっても注意を払いますが、残念ながらそれらを変更すると大変重大な速度低下を引き起こすことになります。(これらの変数のデフォルト値は、それらに関連付けられたバイトコンパイルされた関数を持っていますが、利用者作成のものはもちろんそうではありません。)

これを改善するために、変数をいじくりまわして、(なんとなく) 満足したと感じた後で、*M-x gnus-compile* を実行することができます。これは新しい指定がバイトコンパイルされ、もう一度最高速度に復帰できるということです。Gnus はこれらのバイトコンパイルされた指定を `‘.newsrsrc.elc’` ファイルに保存します。(もっとも利用者が定義した関数は、この関数によってコンパイルされません—それらを `‘~/gnus.el’` ファイルに突っ込んでから、自分でそのファイルをバイトコンパイルしなければなりません。)

8.8 モード行

`gnus-updated-mode-lines` はどのバッファがそれらのモード行を常に最新のものにしておくかを指定します。それはシンボルのリストです。使うことのできるシンボルは `group`, `article`, `summary`, `server`, `browse`, `tree` などです。もし対応するシンボルが存在すると、Gnus は該当する情報でモード行を更新します。この変数が `nil` ならば、画面の再描画はもっと速いでしょう。

デフォルトでは、Gnus は概略バッファと記事バッファのモード行に現在の記事の情報を表示します。Gnus が表示したい情報 (例えば記事の表題) はしばしばモード行よりも長いことがあるので、どこかで切り落とされなければなりません。変数 `gnus-mode-non-string-length` はその行の他の要素 (すなわち情報でない部分) がどのくらいの長さであるかを指定します。もしモード行に追加の要素を入れたなら、この変数を修正する必要があります:

```
(add-hook 'display-time-hook
  (lambda () (setq gnus-mode-non-string-length
    (+ 21
      (if line-number-mode 5 0)
      (if column-number-mode 4 0)
      (length display-time-string)))))
```

もしこの変数が `nil` であるなら (これがデフォルトですが)、モード行は切り落とされず、詰め込みもされません。デフォルトでは、バッファの完全なパーセント表示さえもモード行から追いやられる可能性もあるので、おそらく望ましい設定ではないことに注意して下さい。利用者が自分の設定に合うようにこの変数を適切に設定しなければなりません。

8.9 ハイライトとメニュー

変数 `gnus-visual` は Gnus を素敵にするたいていの方面の操作をします。`nil` であると、Gnus はメニューを作ったり、素敵な色やフォントを使ったりしようとしません。これはさらに '`gnus-vis.el`' ファイルを読み込むことも禁止します。

この変数は有効にされている視覚的なプロパティのリストであることができます。以下の要素は有効で、デフォルトですべて含まれています:

`group-highlight`
グループバッファでハイライト (強調表示) をします。

`summary-highlight`
概略バッファでハイライトをします。

`article-highlight`
記事バッファでハイライトをします。

`highlight`
すべてのバッファでハイライトをするようにします。

`group-menu`
グループバッファでメニューを作成します。

`summary-menu`
概略バッファでメニューを作成します。

`article-menu`
記事バッファでメニューを作成します。

`browse-menu`
ブラウズバッファでメニューを作成します。

`server-menu`
サーバーバッファでメニューを作成します。

`score-menu`
スコアバッファでメニューを作成します。

`menu` すべてのバッファでメニューを作成します。

ですから、記事バッファだけをハイライトしたくて、すべてのバッファでメニューを作りたい場合は、このようにすることができます:

```
(setq gnus-visual '(article-highlight menu))
```

もしハイライトだけで、メニューの類は欲しくないときは、次のようにできます:

```
(setq gnus-visual '(highlight))
```

`gnus-visual` が `t` であると、ハイライトとメニューはすべての Gnus のバッファで使用されます。

他のすべてのバッファの外見に影響する総合的な変数は:

`gnus-mouse-face`
これは Gnus でマウスのハイライトに使われるフェイス (すなわちフォント) です。`gnus-visual` が `nil` であると、マウスハイライトはなされません。

まったく違ったメニューを作成するために、関連するフックがあります:

`gnus-article-menu-hook`

記事モード (article mode) のメニューを作成した後に呼ばれるフックです。

`gnus-group-menu-hook`

グループモード (group mode) のメニューを作成した後に呼ばれるフックです。

`gnus-summary-menu-hook`

概略モード (summary mode) のメニューを作成した後に呼ばれるフックです。

`gnus-server-menu-hook`

サーバーモード (server mode) のメニューを作成した後に呼ばれるフックです。

`gnus-browse-menu-hook`

概観モード (browse mode) のメニューを作成した後に呼ばれるフックです。

`gnus-score-menu-hook`

スコアモード (score mode) のメニューを作成した後に呼ばれるフックです。

8.10 ボタン

最新流行のマウス *mouse* 装置が、近ごろではちゃんとした操作法を学びたがらない若者やかっこいい子どもたちの間で大人気です。それでは私が Tops 20 システム上で Emacs を使っていた頃の、'89 年の夏を思い起こしてみましょう。300 人の利用者が、一つのマシン上で、みんなが Simula コンパイラを走らせていました。ああ、ばかばかしい!

ほんとうに。

さて、`gnus-carpal` を `t` に設定することによって、クリックするだけで何でもできるボタンだらけのバッファを Gnus に表示させることができます。とっても簡単です、ほんとうに。指圧療法の先生に教えてあげて下さい (訳注: *carpal* とは手首の骨のこと)。

`gnus-carpal-mode-hook`

すべての手首モードバッファで実行するフックです。

`gnus-carpal-button-face`

ボタンに使われるフェースです。

`gnus-carpal-header-face`

手首バッファのヘッダーで使用されるフェースです。

`gnus-carpal-group-buffer-buttons`

グループバッファのボタンです。

`gnus-carpal-summary-buffer-buttons`

概略バッファのボタンです。

`gnus-carpal-server-buffer-buttons`

サーバーバッファのボタンです。

`gnus-carpal-browse-buffer-buttons`

閲覧バッファのボタンです。

すべての `buttons` 変数はリストです。これらのリストの要素は、その `car` の項が表示される文を含んでいて、その `cdr` の項が関数シンボルになっている `cons` セルか、もしくはただの文字列のどちらかです。

8.11 デーモン

Gnus、それは（言い伝えによれば）かつて書かれたいかなるプログラムよりも大きく、あなたがやって欲しいと思うさまざまな奇妙なことを、あなたのいないところで行なってくれるものです。例えば、あなたは時たま新着メールをチェックしてもらいたいかもしれません。あるいは Emacs をしばらく放っておいたときすべてのサーバーの接続を切断してもらいたくなるかもしれません。他にも何かそういったことです。

Gnus はさまざまな「ハンドラー」（処理を行なわせるためのもの）を定義することによってそのようなことを可能にします。各ハンドラーは三つの要素から成ります: 「関数」、「時間」、「アイドル」（何もしていない状態を示すもの）パラメーターです。

これは Emacs のアイドル状態が三十分続いたときに接続を切断するハンドラーの例です:

```
(gnus-demon-close-connections nil 30)
```

これは Emacs がアイドルのとき、一時間毎に PGP ヘッダーを走査するハンドラーです:

```
(gnus-demon-scan-pgp 60 t)
```

この「時間」パラメーターと「アイドル」パラメーターは、奇妙かつ素晴らしいやり方で一緒に動きます。基本的に「アイドル」が nil だったら、関数は「時間」分毎に呼び出されます。

「アイドル」が t だったら、関数は Emacs がアイドルだったときに限って「時間」分後に呼び出されます。したがって Emacs がアイドルにならなければ、関数は呼び出されません。いったん Emacs がアイドル状態になると、この関数は「時間」分毎に呼び出されます。

「アイドル」が数値で「時間」も数値だった場合、Emacs のアイドル状態が「アイドル」分続いた場合に限って、「時間」分毎に関数が呼び出されます。

「アイドル」が数値で「時間」が nil だった場合、関数は Emacs のアイドル状態が「アイドル」分続く度に一度呼び出されます。

そして「時間」が文字列だった場合（それは '07:31' のような形式でなければなりません）、関数は毎日その時刻の頃になると一度呼び出されます。もちろん「アイドル」パラメーターによって動作が変わります。

（ここで「分」と言ったとき、それは実際には gnus-demon-timestep 秒のことです。これはデフォルトでは 60 です。もしこの変数を変更すると、すべてのハンドラーの計時に影響を与えます。）

というわけで、ハンドラーを追加したければ、'~/gnus.el' ファイルに以下のようなものを書き込めば良いでしょう:

```
(gnus-demon-add-handler 'gnus-demon-close-connections 30 t)
```

このための既製の関数がいくつか作成されています: gnus-demon-add-nocem, gnus-demon-add-disconnection, gnus-demon-add-nntp-close-connection, gnus-demon-add-scan-timestamps, gnus-demon-add-rescan, および gnus-demon-add-scanmail です。これらの機能を必要とするならば、単にこれらの関数を '~/gnus.el' に入れて下さい。

gnus-demon-handlers に直接ハンドラーを追加した場合には、それを効かせるために gnus-demon-init を実行して下さい。すべてのデーモンを取り消すには、gnus-demon-cancel 関数を使うことができます。

デーモンの追加をやりすぎるのはかなりマズいことです。すべてのサーバーからすべてのニュースとメールを二秒毎に調べまわす関数を付け加えたりすることは、どんな立派なシステムからも確実に追い出される方法です。お行儀良くしましょう。

8.12 NoCeM

Spam とは、同じ記事を何回も何回も何回も投稿することです。Spam は悪いことです。Spam は凶悪です。

Spam はさまざまな反 spam 機関によって、通常一日かそこらで取り消しされます。通常これらの機関は、一緒に NoCeM メッセージも送り出します。NoCeM は“ no see-'em ”(彼らを見たくない) と発音され、意味はその名前の通りです—これらのメッセージは気に触る記事を、つまり、消してしまいます。

どうせそれらの記事が取り消しされてしまうのなら、これらの NoCeM メッセージは何に使われるのでしょうか？ あるサイトでは取り消しメッセージを重視しません。また、あるサイトでは特定の数人からの取り消しメッセージだけを尊重します。そこで、あなたは NoCeM メッセージを使いたくなるかもしれないわけです。これらは ‘alt.nocem.misc’ ニュースグループで配布されています (訳注: 他に ‘fj.news.lists.filters’ や ‘news.lists.filters’ などでも独自の NoCeM メッセージが配布されています)。

Gnus はこのグループのメッセージを自動的に読み、解釈することができ、これで spam を消去します。

もちろん、これらをカスタマイズするための変数がいくつかあります:

gnus-use-nocem

ものごとを始めさせるには、この変数を `t` に設定して下さい。デフォルトでは `nil` です。

この変数にグループレベルとして正の数値を設定することもできます。その場合、この値が `gnus` や `gnus-group-get-new-news` などのコマンドの接頭引数として与えるグループレベル以下だったら、Gnus は新着ニュースをチェックするときに NoCeM メッセージを走査します。さもなければ、それらのコマンドにグループレベルを与えると、Gnus は NoCeM メッセージを走査しません。例えば、メールグループで 1 か 2 を使っていてニュースグループのレベルがデフォルトのままだったら、3 が最も良い選択です。

gnus-nocem-groups

Gnus はこのグループのリストから NoCeM メッセージを探します。デフォルトは次の通りです:

```
("news.lists.filters" "news.admin.net-abuse.bulletins"
 "alt.nocem.misc" "news.admin.net-abuse.announce")
```

gnus-nocem-issuers

NoCeM メッセージを発行する人はたくさんいます。このリストでは、誰のいうことに従いたいかを指定します。デフォルトは次の通りです:

```
("Automoose-1" "clewis@ferret.ocunix.on.ca"
 "cosmo.roadkill" "SpamHippo" "hweede@snafu.de")
```

彼らはみんな、立派で高潔な市民です。

このリストに含まれる有名な反 spam 家たちは <http://www.xs4all.nl/~rosalind/nocemreg/no> に載っています。

これらすべての人々の NoCeM メッセージに留意する必要はありません—言うことを聞きたい人だけで良いのです。また、それらの人たちからの NoCeM メッセージを、すべて受け入れる必要もありません。それぞれの NoCeM メッセージは、そのメッセージの厳密な (多少は厳密、たいていはそうでもない) 定義を与える種別 `type` ヘッダーを

持っています。一般的な種別は 'spam', 'spew', 'mmf', 'binary', および 'troll' です。これを指定するには、リストの中で（発行者 条件...）という要素を使う必要があります。それぞれの条件は、文字列（使いたい種別に合致する正規表現）または（not 文字列）という形式のリスト（この場合の「文字列」は使いたくない種別に合致する正規表現）のどちらかです。

例えば、Chris Lewis からの、'troll' メッセージ以外のすべての NoCeM メッセージを欲しい場合には、こうすれば良いでしょう：

```
("clewis@ferret.ocunix.on.ca" ".*" (not "troll"))
```

一方、彼の 'spam' と 'spew' メッセージ以外は何も要らないのであれば、以下のようにできます：

```
("clewis@ferret.ocunix.on.ca" (not ".*") "spew" "spam")
```

この指定は左から右に適用されます。

gnus-nocem-verifyer

これは NoCeM 発行者が本人であることを検証する関数でなくてはなりません。ディフォルトは pgg-verify で、これは検証に成功したら非-nil を返し、そうでなければ（NoCeM メッセージが署名されていない場合を含みます）nil を返します。もしこれが非常に遅くて、検証結果を気にしない（これはたぶん危険です）のであれば、この変数を nil にすることができます。

以前、ディフォルトは Mailcrypt の関数である mc-verify でした。まだそれを使うことができますが、PGP の公開鍵を GnuPG の鍵束に加えることを厭わなければ、GnuPG とともに動作するディフォルトの関数に変えることができます。

gnus-nocem-directory

これは Gnus が NoCeM キャッシュファイルを保存する場所です。ディフォルトは '~/News/NoCeM/' です。

gnus-nocem-expiry-wait

古い NoCeM 項目をキャッシュから消すまでの日数。ディフォルトは 15 です。これを短くするほど Gnus は速くなりますが、古い spam を見ることになってしまうかもしれません。

gnus-nocem-check-from

非-nil では、記事のボディーにある発行人の正当性を調べます。そうでない場合は、著者が正しい発行人でなくても気にせずに記事を取り込みますが、もしあなたが正しい発行人を見分けられるならば、そうした方がとても速くなるでしょう。

gnus-nocem-check-article-limit

すべての NoCeM グループにおけるチェックする記事の最大数を指定します（訳注: nil で無制限）。NoCeM グループは巨大になることがあり、そうなると処理がとても遅くなります。

NoCeM を使うと、もしかするとメモリ喰いになるかもしれません。あなたがたくさんの生きたグループ（つまり購読あるいは非購読グループ）を持っていると、Emacs のプロセスは大きくなってしまいうでしょう。もしこれが問題であれば、非購読のグループを全部（あるいはその多くを）消し去って（kill して）しまうべきです（see Section 2.4 [Subscription Commands], page 18）。

8.13 やり直し

実行したことのやり直しができると、とても便利です。Emacs の普通のバッファでは十分に簡単です—単に undo ボタンを押すだけです。しかし Gnus のバッファでは、それは簡単ではありません。

Gnus がバッファ内に表示しているものは、Gnus にとってはまったく何の価値もありません—これはみんな、利用者に綺麗に見えるようにデザインされているただのデータなのです。C-k でグループバッファからグループを消去すると、その行は消え去りますが、それは実際の動作—当のグループを Gnus の内部構造体から削除すること、の単なる副作用でしかありません。これらのやり直しは、通常の Emacs の undo 関数では行なうことができません。

Gnus は利用者がすることを憶えておいて、利用者がすることの逆を行なうことによって、これを多少は救済しようとします。利用者が undo キーを押すと、一段階または数段階前までの操作を元に戻すコードを実行します。しかし、すべての操作が簡単に逆戻りできるわけではないので、現在 Gnus は、やり直し可能なキーの機能を僅かしか提供していません。これらはグループの削除、グループの貼り付け、およびグループの既読記事のリストの変更です。実際それだけです。将来はもっと機能が追加されるかもしれませんが、追加されるそれぞれの機能は保存すべきデータを増やすので、決して Gnus は完全にやり直し可能にはならないでしょう。

やり直し機能は gnus-undo-mode マイナーモードによって提供されます。これは gnus-use-undo が nil 以外であれば使用され、これがデフォルトです。C-M-_ キーが gnus-undo 命令を実行します。これは通常の Emacs の undo 命令にいくぶん似ているはずです。

8.14 述語指示子

いくつかの Gnus の変数は「述語指示子」(predicate specifiers) です。これは、その多くをすべて記述する必要なしに、述語の仕様に融通を効かせることができる特別な形式です。

これらの指示子は関数、シンボルおよびリストからなるリストです。

例です:

```
(or gnus-article-unseen-p
    gnus-article-unread-p)
```

利用できるシンボルは or、and および not です。関数はすべて一つのパラメーターを受け取ります。

呼ぶことができる関数を作るために、Gnus はこれらの指示子について内部的に gnus-make-predicate を使います。この関数へのこの入力パラメーターは、述語指示子の中のすべての関数に渡されます。

8.15 司会役

もしあなたが司会者 (モデレーター) ならば、'gnus-mdrtn.el' パッケージを使うことができます。これは標準の Gnus パッケージには含まれていません。'larsi@gnus.org' に、どのグループの司会を行なうのかを述べたメールを書いて下さい。そうすればコピーを手に入れることができます。

司会者用パッケージは概略バッファのマイナーモードとして実装されています。

```
(add-hook 'gnus-summary-mode-hook 'gnus-moderate)
```

をあなたの '~/.gnus.el' ファイルに入れて下さい。

あなたが 'rec.zoofle' の司会者だとすると、これは以下のように動作するようになっていきます:

1. 受信したメールを 'Newsgroups:.*rec.zoofle' に合致させることによって分割します。これは投稿されようとしているすべての記事を、あるメールグループ—例えば 'nnml:rec.zoofle'—に入れます。
2. あなたは時折このグループに入り、`e` (edit-and-post) あるいは `s` (just send unedited) 命令を使って記事を投稿します。
3. 'rec.zoofle' ニュースグループを読んでいる途中で、もしあなたが承認していない記事をたまたま見つけたとしたら、`c` 命令で取り消しできます。

二つのグループで司会者モードを使うとすれば、こうなります:

```
(setq gnus-moderated-list
      "^nnml:rec.zoofle$\\|^rec.zoofle$")
```

8.16 グループを取得する

時々「Gnus が起動しているかどうかを気にしないでこのグループを読みたい。」ということができれば便利ことがあります。これは、利用者よりもプログラムのコードを書く人に便利な機能ですが、どちらにしろ `gnus-fetch-group` コマンドはこの機能を提供します。それはグループの名前を引数としてとります。

8.17 画像の拡張

XEmacs それに v21 以上の Emacs は絵やその種のものを表示することができる¹ ので、Gnus はこれを利用することにしました。

8.17.1 X-Face

X-Face ヘッダーは、メッセージの著者を表わすことになっている 48 × 48 画素の白黒 (深さ 1 bit の) の絵を描きます。これは進化し続けるあまたのメールとニュースリーダーによってサポートされるでしょう。

X-Face ヘッダーを見るには 'compface' をサポートしている Emacs (ほとんどの XEmacs の版がサポート) か、変換または表示のための適切なプログラムをインストールしてあることが必要です。あなたの Emacs が自前で画像の表示をサポートしているならば、デフォルトで From ヘッダーの前に顔が表示されます。Emacs が自前で X-Face をサポートしていない場合、Gnus は pbmplus パッケージとその仲間の外部プログラム (下記参照) を使って X-Face ヘッダーを変換しようとします。X-Face をサポートするようにコンパイルされている XEmacs は速いです。画像をサポートしていない Emacs では、デフォルトでは表示のための処理を `display` というプログラムに委ねます。

GNU/Linux システムの場合、ImageMagick パッケージに含まれている `display` プログラムを使います。外部プログラムとしては `netpbm`、`libgr-progs` および `compface` のような名前のもを探します。Windows では <http://gnuwin32.sourceforge.net> にある `netpbm` および `compface` パッケージを使っても良いです。PATH 環境変数に `bin` ディレクトリーを追加する必要があります。

変数 `gnus-article-x-face-command` で、X-Face ヘッダーを表示するために何のプログラムを使うかを制御します。この変数が文字列ならば、この文字列がサブシェルで実行されます。関数ならば、この関数が顔を引数として呼ばれます。もし `gnus-article-x-face-too-ugly` (これは正規表現です) が From 欄に合致すれば、顔は表示されません。

(注: 変数/関数名には `xface` ではなく `x-face` が使われます。)

¹ MS ウィンドウズの Emacs 21 は画像をサポートしていません。Emacs 22 はします。

フェースと変数:

`gnus-x-face`

X-Face を表示するためのフェース。このフェースの色が表示される X-Face の前景色と背景色として使われます。デフォルトの色は黒と白です。

`gnus-face-properties-alist`

Face (see Section 8.17.2 [Face], page 267) と X-Face 画像に適用される、画像の形式とプロパティの連想リストです。デフォルト値は Emacs 用の `((pbm . (:face gnus-x-face)) (png . nil))` または XEmacs 用の `((xface . (:face gnus-x-face)))` です。例を挙げましょう:

```
;; From ヘッダーにおける Face と X-Face の高さを指定します。
(setq gnus-face-properties-alist
      '((pbm . (:face gnus-x-face :ascent 80))
        (png . (:ascent 80))))

;; Face と X-Face を凹んだボタンのように表示します。
(setq gnus-face-properties-alist
      '((pbm . (:face gnus-x-face :relief -2))
        (png . (:relief -2))))
```

いろいろな画像の形式で利用可能なプロパティについては See section “Image Descriptors” in *The GNU Emacs Lisp Reference Manual*, を参照して下さい。今のところ Emacs では pbm が X-Face 画像に使われ、png が Face 画像に使われます。XEmacs では、それが ‘libcompface’ ライブラリーとともに構築されていれば、xface 画像形式に :face プロパティだけが効果を及ぼします。

投稿様式 (posting style) を使うのであれば、`gnus-posting-styles` に `x-face-file` の項を加えれば良いでしょう (see Section 5.6 [Posting Styles], page 128)。さもないと、外に出すメッセージに X-Face ヘッダーを簡単に挿入できるようにするために Gnus が提供する、いくつかの便利な関数と変数を利用することができます。これらの機能のためには、前述の ImageMagick、netpbm または他の画像を変換するパッケージ (何が必要かは、下記の変数群の値によります) も必要です。

`gnus-random-x-face` は `gnus-x-face-directory` にあるすべての ‘pbm’ ファイルをくまなく探してランダムに一つを選び取り、シェルコマンド `gnus-convert-pbm-to-x-face-command` を使ってそれを X-Face の形式に変換します。‘pbm’ ファイルは 48 × 48 画素の大きさでなければなりません。それは X-Face ヘッダーのデータを文字列で返します。

`gnus-insert-random-x-face-header` は `gnus-random-x-face` を呼んで、ランダムに生成されたデータによる X-Face ヘッダーを挿入します。

`gnus-x-face-from-file` はパラメーターとして GIF ファイルを受け取り、シェルコマンド `gnus-convert-image-to-x-face-command` を使ってそのファイルを X-Face の形式に変換します。

一番目の関数の一般的な使い方を示します。以下のようなものを ‘~/gnus.el’ ファイルに書き込んで下さい:

```
(setq message-required-news-headers
      (nconc message-required-news-headers
              (list '(X-Face . gnus-random-x-face))))
```

最後の関数を使うのは、このようになるでしょう:

```
(setq message-required-news-headers
  (nconc message-required-news-headers
    (list '(X-Face . (lambda ()
                      (gnus-x-face-from-file
                       "~/My-face.gif"))))))
```

8.17.2 Face

Face ヘッダーは、本質的に X-Face をよりファンキーに変形したものです。それらは、メッセージを書いた人を象徴することになっている 48 × 48 画素のカラー画像を描きます。

Face ヘッダーの内容は base64 でエンコードされた PNG の画像でなければなりません。正確な仕様について、<http://quimby.gnus.org/circus/face/> を参照して下さい。

変数 `gnus-face-properties-alist` は表示される Face 画像の外観に影響します。See Section 8.17.1 [X-Face], page 265.

Face ヘッダーを見るには Emacs が PNG 画像を表示できる必要があります。

Gnus は外に出すメッセージに Face ヘッダーを簡単に挿入できるようにするための、便利な関数と変数を少しばかり提供します。

`gnus-convert-png-to-face` は 726-byte 以下の 48 × 48 の PNG の画像を受け取って、それを Face に変換します。

`gnus-face-from-file` は JPEG ファイルの名前をパラメーターとして受け取り、シェルコマンド `gnus-convert-image-to-face-command` を使ってそのファイルを Face フォーマットに変換します。

この関数の代表的な使い方を挙げておきましょう。以下のようなものを ‘`~/.gnus.el`’ ファイルに入れて下さい:

```
(setq message-required-news-headers
  (nconc message-required-news-headers
    (list '(Face . (lambda ()
                  (gnus-face-from-file "~/face.jpg"))))))
```

8.17.3 スマイリー

スマイリー *smiley* は Gnus とは別のパッケージですが、スマイリーを使っているパッケージは現在 Gnus だけなので、ここで説明します。

ひとこと言えば—Gnus でスマイリーを使うには、以下を ‘`~/.gnus.el`’ ファイルに書き込んで下さい。

```
(setq gnus-treat-display-smileys t)
```

スマイリーは、文字の顔マーク—‘`: -)`’, ‘`8 -)`’, ‘`: - (`’ などといったもの—を絵に割り当てて、文字の顔マークの代わりにその絵を表示します。この変換は文字に合致する正規表現と、それに割り当てられたファイル名のリストで制御されます。

使われる連想リストは、変数 `smiley-regexp-alist` で設定します。各要素の最初の項目は合致する正規表現で、二番目の要素は絵で置き換えられる正規表現のグループ番号、そして三番目の要素は表示されるファイルの名前です。

以下の変数は、スマイリーがこれらのファイルを探す場所をカスタマイズします:

```
smiley-data-directory
```

スマイリーが顔ファイルを探す場所です。

`gnus-smiley-file-types`

スマイリーのファイル名として試してみる拡張子のリストです。

8.17.4 Picons

それで...、あなたはこのニュースリーダーをさらにもっと遅くしたいってわけですね! これはそうするのにぴったりの方法です。さらにこれは、あなたがニュースを読んでいるんだということを、あなたの肩越しに見つめている人に印象づけるための素晴らしい方法でもあります。

Picon とはなんでしょう? Picons ウェブサイトから直接引用しましょう。

Picon とは「個人アイコン (personal icons)」の略です。これは、ある電子メールアドレスのための適切な画像を見つけることができるように、無理矢理小さくしてデータベースにまとめられた画像たちで、ネット上の利用者やドメインを表現するために使われます。利用者とドメイン以外に、Usenet ニュースグループや天気予報のための picon データベースがあります。picon は白黒の XBM 形式、またはカラーの XPM 形式および GIF 形式のいずれでも構いません。

Picon データベースの入手とインストールの手順については、ウェブブラウザで <http://www.cs.indiana.edu/picons/ftp/index.html> を訪ねてみて下さい。

もし Debian GNU/Linux を使っているのなら、`'apt-get install picons.*'` と言えば、Gnus が見つけることができる picon がインストールされます。

Picon の表示ができるようにするためには、picon データベースがあるディレクトリーが、ただ単に `gnus-picon-databases` に設定されているようにして下さい。

変数 `gnus-picon-style` は picon をどのように表示するかを制御します。inline だったらテキスト形式の表現が置き換えられます。right だったら、テキスト形式の表現の右側に picon が加えられます。

ものごとの所在を管理するために、以下の変数を設けています。

`gnus-picon-databases`

Picon データベースの場所です。これは `'news'`, `'domains'`, `'users'` (などなど) のサブディレクトリーが含まれているディレクトリーのリストです。(`"/usr/lib/picon"/usr/local/faces`) がデフォルトです。

`gnus-picon-news-directories`

`gnus-picon-databases` からニュースグループ用のフェースを探すためのサブディレクトリーのリストです。デフォルトは (`"news"`) です。

`gnus-picon-user-directories`

`gnus-picon-databases` から利用者のフェースを探すためのサブディレクトリーのリストです。(`"local" "users" "usenix" "misc"`) がデフォルトです。

`gnus-picon-domain-directories`

`gnus-picon-databases` からドメイン名のフェースを探すためのサブディレクトリーのリストです。デフォルトは (`"domains"`) です。このリストに `"unknown"` を追加しておきたい人もいますでしょう。

`gnus-picon-file-types`

Picon のファイル名として試してみる順に並べられた拡張子のリストです。デフォルトは (`"xpm" "gif" "xbm"`) から Emacs に組み込まれていないものを除外したものです。

8.17.5 さまざまな XEmacs 変数

`gnus-xmas-glyph-directory`

これは Gnus が絵を探す場所です。Gnus は通常このディレクトリーを自動検出しますが、もし標準的でないディレクトリー構造を持っている場合は、これを手動で設定することができます。

`gnus-xmas-modeline-glyph`

すべての Gnus のモード行で表示される画像。これはデフォルトではちいさなヌー (gnu) の頭です。

8.17.5.1 ツールバー

`gnus-use-toolbar`

この変数はツールバーを表示する位置を指定します。nil だったらツールバーを表示しません。非-nil の場合、それは default, top, bottom, right または left の中の一つのシンボルでなければなりません。default だったらデフォルトのツールバーを使い、他のものだったらその名前が示す場所にツールバーを表示します。デフォルトは default です。

`gnus-toolbar-thickness`

高さの幅のコンス (cons) で、ツールバーの厚さを指定します。高さは上辺か下辺に表示するツールバーで使われ、幅は右端か左端に表示するツールバーで使われます。デフォルトはデフォルトのツールバーの値です。

`gnus-group-toolbar`

グループバッファ内のツールバーです。

`gnus-summary-toolbar`

概略バッファ内のツールバーです。

`gnus-summary-mail-toolbar`

メールグループの概略バッファ内のツールバーです。

8.18 ファジーな一致

Gnus はスコア付け、スレッドの形成、およびスレッドの比較などを行なうときに、Subject 行のファジーな合致 *fuzzy matching* を提供します。

正規表現による合致とは違って、ファジーな合致はとってもファジーです。あまりにもファジーすぎて、何がファジーであるかという定義さえ無いし、実装も何度も変更されています。

基本的に、これは比較の前に行から邪魔物を取り除こうとします。‘Re: ’、挿入句の印、および空白文字等々が文字列から除去され、その結果を比較します。これはほとんどの場合妥当な結果をもたらします—たとえニュースリーダーの仮面をかぶった文字列切り刻み機で生成された文字列が差し出されても、です。

8.19 spam メールの裏をかく

ここ最近の USENET では、宣伝のハゲタカどもが彼らの詐欺や製品を押し付けるための電子メールアドレスを探そうとして、気違いのようにニュース上をうろついて grep しまくっています。これに対する反動として、多くの人々が無意味なアドレスを From 行に入れはじめるようになってしまいました。私はこれは逆効果を招くと思います—あなたが書いたことに対する返信として人々が正当な

メールを送ることを面倒にさせるだけでなく、誰が書いたものなのかを分かりづらくします。こんな書き換えは、結局は押し付け宣伝メールそれ自身よりも大きな脅威となるかもしれません。

私にとっての spam メール最大の問題は、嘘の口実で入ってくるからです。私が *g* を押すと、Gnus は十通の新着メールがありますと陽気に私に教えてくれます。私は「おおっ、わーい! 僕って幸せ!」と言ってメールグループを選択します。しかしそこには、二つのネズミ講と、七つの広告（「最新! 奇跡の育毛トニック、ふさふさでつやつやの髪をあなたのつま先 () に!」）と、悔い改め神を信じよ、という一つのメールがあるだけなのです。

これは迷惑千万です。あなたがそれに関してできることがあります。

訳注 : ホビット族用の育毛トニック。たぶん。

8.19.1 Spam の問題

初めに spam の背景から。

あなたが電子メールを使っているならば、spam (専門用語としては Unsolicited Commercial E-mail—望まれない商用電子メール—の頭文字 UCE) のことはよく知っているでしょう。簡単に言えばそれは紙のメールに比べて電子メールの配送がとて安くつくために存在し、非常に小さな割合の人々が UCE に応答するだけで広告主に利益をもたらすのです。皮肉なことに最も一般的な spam の一つは、さらに spam を助長するための電子メールアドレスのデータベースを提供します。Spam の送信者はふつう *spammers* と呼ばれますが、*vermin*、*scum*、*sociopaths* および *morons* のような用語もよく使われています。

Spam は種々さまざまな出どころからやって来ます。有用なメッセージを捨てずにすべての spam を単に始末することは不可能です。良い例は TMDA (訳注: 送信する度にユニークなアドレスを使う) システムで、それは、あなたが知らない送信者からの電子メールがあなたのもとに届くことができる前に、彼らに対して彼ら自身が正当な送信者であることの確認を求めます。正当な出どころからの電子メールが、それらの出どころが TMDA システムを通して確認できない、または行なわれない場合は捨てられてしまうかもしれないというマイナス面は、TMDA の技術的な側面に立ち入らなくても明白です。もう一つの TMDA の問題は、電子メールの配送と処理への基本的な理解を、利用者に求めていることです。

Spam の除去 (filtering) への最も単純な取り組みは、メールサーバーで、あるいは入ってきたメールを分類するときに濾過すること (filtering) です。毎日 'random-address@vmin.com' から 200 通の spam メッセージを受け取るのならば、'vmin.com' を阻止すれば良いでしょう。「バイアグラ」に関するメッセージを 200 通受け取るのならば、「バイアグラ」を含むすべてのメッセージを捨ててしまえば良いでしょう。例えばブルガリアからたくさんの spam がやって来るのならば、ブルガリアの IP から来るすべてのメールを濾過すれば良いでしょう。

これは、残念ながら正当な電子メールを捨てるためのすぐれた方法です。あなたに連絡しようとする国 (ブルガリア、ノルウェー、ナイジェリア、中国、等) 全体、または大陸 (アジア、アフリカ、ヨーロッパ、等) さえも封じ込めてしまう危険は明らかなので、あなたに選択権があるのならば、そんなことはしないで下さい。

もう一つの例として、とても示唆に富んで有益な RISKS ダイジェストは、それが spam メッセージと共通の語を含んでいるために、熱心すぎるメール濾過器によって阻止されてしまいます。それでもなお孤立した環境では、注意深く使うことによって直接の濾過は有益になり得ます。

もう一つの電子メール濾過への取り組みは分散型 spam 処理で、DCC (訳注: Distributed Checksum Clearinghouse—<http://www.rhyolite.com/anti-spam/dcc/>) がそのようなシステムを導入しています。本質的には、世界中の *N* 個のシステムが、ガーナ、エストニアあるいはカリフォルニアにあるマシン *X* が spam 電子メールを送出していることを認めたら、それら *N* 個のシステムは *X* または *X* からやって来た spam メールをデータベースに記入します。Spam 検出の基準は

一様ではありません。それは送られたメッセージの数やメッセージの内容などであるかもしれません。メッセージが spam かどうかを分散処理システムの利用者が知りたい場合、彼はそれらの N 個のシステムのうちの一つを調べます。

分散型 spam 処理は一度にたくさんのメッセージを送る spammers と非常によく戦ってくれますが、それには利用者がかなり複雑なチェックを設定することが必要です。商用とフリーな分散型 spam 処理システムがあります。分散型 spam 処理は、それ自体の危険もはらんでいます。例えば、正当な送信者が spam を送ったかで非難され、彼らのウェブサイトやメーリングリストがその事件のために暫くの間閉鎖されてしまう、とか。

Spam の濾過への統計的な取り組みもまた普及しています。それは過去の spam メッセージの統計的な分析に基づいています。通常その分析は、おそらく単語の対か三つの単語の組合せの合成による、単語の出現頻度の単純な計数です。Spam の統計分析はほとんどの場合にとってもよく働くのですが、時として正当な電子メールを spam として分類してしまうことがあります。分析には時間がかかります。すべてのメッセージを分析しなければなりません。そして利用者は spam を分析するためのデータベースを蓄えなければなりません。サーバーでの統計分析は人気を得ています。これには、利用者は単にメールを読めば良いという長所と、しかしサーバーにそれが過ってメールを分類したことを伝えるのが困難だという短所があります。

余人の言を待たずとも、spam との戦いは楽ではありません。ママからの電子メールとバイアグラ広告を区別する魔法のスイッチはありません。人々は非-spam と spam を区別するのに手を焼いているというのに。それは、spammers が懸命にそれらをママだと思わせようとしているのが本質だからです。Spamming は、世界が彼らに恩義があると思っている人々の一団からの、腹立たしく、無責任で、ばかげた行為です。以下の各章が spam なる疫病との戦いの助けになることを望みます。

8.19.2 Spam 退治の基礎

Spam に対処する一つの方法は、Gnus にすべての spam を 'spam' メールグループに分離させてしまうことです (see Section 6.3.3 [Splitting Mail], page 147)。

最初に、あなたに連絡することができる正しいメールアドレスを一つ選び、それをすべてのあなたのニュース記事の From ヘッダーに入れましょう。(ここでは 'larsi@trym.ifi.uio.no' を選びましたが、'larsi+usenet@ifi.uio.no' の形式のたくさんのアドレスの方が良い選択です。あなたのサイトの sendmail の設定がメールアドレスのローカル部としてどんなキーワードを受け付けるかは、あなたのサイトのシステム管理者に聞いて下さい。)

```
(setq message-default-news-headers
  "From: Lars Magne Ingebrigtsen <larsi@trym.ifi.uio.no>\n")
```

そして nnmail-split-fancy に以下の分割規則を入れて下さい (see Section 6.3.6 [Fancy Mail Splitting], page 157)。

```
(...
  (to "larsi@trym.ifi.uio.no"
    (| ("subject" "re:.*" "misc")
      ("references" ".*@.*" "misc")
      "spam")))
...)
```

この意味は、このアドレスに届いたすべてのメールが疑わしいが、'Re:' で始まる Subject が付いているか、References ヘッダーが付いていればおそらく OK だろう、ということです。残りはすべて 'spam' グループに行きます。(このアイデアはおそらく Tim Pierce 氏によるものです。)

これに加えて、多くのメール spam 屋は、あなたのところの SMTP サーバーと直接話して、To ヘッダーにあなたのメールアドレスが明示されないようにします。なんでそんなことをするのかはわ

かりませんが—もしかしたら、この裏をかく機構の裏をかくためかな？ どちらにしても、対処は簡単なことです—特級分割規則を以下のように終端させることによって、あなた宛てでないものを全部 ‘spam’ グループに入れるだけです：

```
(
  ...
  (to "larsi" "misc")
  "spam")
```

私の経験では、これで実質的にはすべてが正しいグループに分類されます。まあ、それでもときどき ‘spam’ グループをチェックして、正しいメールがあるかチェックしなくてはいいかもしれませんけど。あなたが自分が良いネットワーク市民であると思っているなら、それぞれの押し付け宣伝メールの関係当局に苦情を送り付けることさえもできます—暇なときにでもね。

これで私のところでは動いています。これでみんなは簡単な方法で私に連絡を取ることができ（普通に `r` を押すだけでできる）、私は spam に煩わされることはまったくありません。お互いに有利な状況です。私に言わせれば、From ヘッダーを偽造して存在しないドメインに送らせるのはすごく良くないです。

訳注：以上の文章は 1997 年 4 月に書かれました。

この手法には注意して下さい。Spammers はそれに気付いています。

8.19.3 SpamAssassin, Vipul’s Razor, DCC, etc

Spam を避けるための前章のヒントが十分だった日々は過ぎ去りました。今では受け取ったたくさんさんの spam を減らすという触れ込みの多くの道具があります。この章は、新しい道具が古いものに取って代わって行くにつれてすぐに時代遅れになってしまうでしょうが、幸いなことにこれらのほとんどの道具は類似のインターフェースを持っているようです。この章は例として SpamAssassin を使っていますが、他のほとんどの道具にも簡単に適合するはずです。

この章は spam.el パッケージとは関係無いことに注意して下さい。それは次の章で論じられます。すべての spam.el の機能に関心が無いのならば、これらの単純なレシピで間に合わせることができます。

もしあなたが使う道具がメールサーバーにインストールされていないならば、あなた自身がそれと呼び出す必要があります。以下に :postscript メールソース指示子 (see Section 6.3.4.1 [Mail Source Specifiers], page 148) を使う場合の考え方を示します。

```
(setq mail-sources
  '((file :prescript "formail -bs spamassassin < /var/mail/%u"
    (pop :user "jrl"
      :server "pophost"
      :postscript
        "mv %t /tmp/foo; formail -bs spamc < /tmp/foo > %t"))))
```

いったんメールを受けるスプールをどうにかして処理する、例えばそのメールに spam であることを表示するヘッダーを含めるようにすれば、それをふるい落とす準備は完了です。使うのは普通の分割方式 (see Section 6.3.3 [Splitting Mail], page 147) です：

```
(setq nnmail-split-methods '(("spam" "^X-Spam-Flag: YES")
  ...))
```

または特級分割方式 (see Section 6.3.6 [Fancy Mail Splitting], page 157) です：

```
(setq nnmail-split-methods 'nnmail-split-fancy
  nnmail-split-fancy '(| ("X-Spam-Flag" "YES" "spam")
```

```
...))
```

いくらかの人たちは `:prescript` を使ってメールをいろんなプログラムにパイプすることを嫌うかもしれません (もし何かのプログラムにバグがあったら、すべてのメールを失ってしまうかもしれません)。あなたがそれらの一人ならば、別の解は分割するときに外部の道具を呼ぶことです。特級分割方式の例です:

```
(setq nnmail-split-fancy '(| (: kevin-spamassassin)
...))

(defun kevin-spamassassin ()
  (save-excursion
    (widen)
    (if (eq 1 (call-process-region (point-min) (point-max)
                                   "spamc" nil nil nil "-c"))
        "spam")))
```

Nnimap バックエンドの場合、デフォルトでは記事のボディがダウンロードされないことに注意して下さい。それをするためには `nnimap-split-download-body` を `t` に設定する必要があります (see Section 6.5.1 [Splitting in IMAP], page 190)。

以上がこれに関することです。ある種の spam はどうしても素通りしてしまいがちなので、spam を読むはめになったときに呼ぶための気の利いた関数が必要でしょう。これがその気の利いた関数です:

```
(defun my-gnus-raze-spam ()
  "SPAM の処理を Vipul の Razor に委ねてから、
それに期限切れ消去可能の印を付けます。"
  (interactive)
  (gnus-summary-show-raw-article)
  (gnus-summary-save-in-pipe "razor-report -f -d")
  (gnus-summary-mark-as-expirable 1))
```

8.19.4 Hashcash

Spam と戦うための斬新な技法は、いくらか負担にはなるが明らかに独特なことを、送信するメッセージに対して送信者が行なうことを求めることです。これはインターネット標準の一部ではないので、世界中のすべての人がこの技法を使うことは当てにできないという明らかな欠点がありますが、小規模な集団では役に立つでしょう。

前章の道具類は実際にうまく働きますが、それらは新しい形式の spam が現れるたびにしょっちゅう更新かつ整備されることによってのみ動作します。このことは、小さなパーセンテージの spam がいつも素通りしてしまうことを意味します。それはまた、どこかでだれかがそれらの道具を更新するために、たくさんの spam を読まなければならないことをも意味します。Hashcash はそれを回避しますが、代わりにあなたが電子メールで連絡するすべての人たちに、なるべくその仕組みをサポートしてもらいたいのです。あなたは实际的 (pragmatic) か独断的 (dogmatic) かの観点で、二つの取り組みを考えることができます。それらのやり方には、それら自体の利点もあれば不利な点もありますが、現実の世の中ではしばしばあるように、それらを連係させたものはどちらか一方より強力です。

「いくらか負担にはなる」とは CPU 時間を消費することで、もっと具体的には一定数のビットまでハッシュの衝突 (hash collision) を計算することです。その結果としての hashcash クッキーは 'X-Hashcash:' ヘッダーに挿入されます。もっと詳しいこと、そしてこの機能を使うためにインストールする必要がある外部アプリケーションの hashcash については <http://www.hashcash.org/> を参照して下さい。さらなる情報が <http://www.camram.org/> で見つかるでしょう。

送信するメッセージのそれぞれについて `hashcash` を生成させようと思うなら、以下のように `message-generate-hashcash` (see section “メールヘッダー” in *The Message Manual*) をカスタマイズして下さい:

```
(setq message-generate-hashcash t)
```

いくつかの追加の変数の設定もしなければなりません:

`hashcash-default-payment`

この変数はハッシュの衝突を成すデフォルトのビット数を示します。デフォルトは 20 です。提唱されている有効な値は 17 から 29 までの数です。

`hashcash-payment-alist`

何人かの受取人は、あなたにデフォルトより多くの CPU 時間を費やすことを要求するかもしれません。この変数は `(addr amount)` の形式の要素のリストで、`addr` は受取人 (メールアドレスかニュースグループ)、`amount` は衝突で必要とされるビット数です。これはまた `(addr string amount)` の要素を持つことも可能で、`string` は文字列 (通常はメールアドレスかニュースグループ名) として使われます。

`hashcash-path`

`hashcash` バイナリーがインストールされている場所です。この変数は `executable-find` によって自動的に設定されるはずですが、それが `nil` だった (ありがちなのは `hashcash` バイナリーが実行 `path` 中に無い) 場合は、`hashcash payments` をチェックするときに警告され、`hashcash payments` を生成するときはエラーになるでしょう。

Gnus は `hashcash` クッキーを認証することができますが、手でカスタマイズしたメール濾過スクリプトで行なうこともできます。メッセージ中の `hashcash` クッキーを認証するには、`hashcash.el` ライブラリーの `mail-check-payment` 関数を使って下さい。入ってきたメールの `hashcash` クッキーを確認し、それによってメールを濾過するために、`spam-use-hashcash` バックエンドで `spam.el` を使うこともできます (see Section 8.20.6.4 [Anti-spam Hashcash Payments], page 285)。

8.20 Spam パッケージ

Spam パッケージは `spam` を検出して濾過するために集結された機構を Gnus に提供します。それは新着メールを濾過し、`spam` か `ham` かに応じてメッセージを処理します。(Ham は `spam` ではないメッセージを示すために、このマニュアルを通して使われる名前です。)

8.20.1 Spam パッケージ序説

Spam パッケージがどのように働くかを理解するために、必ずこの章を読んで下さい。読み飛ばし、速読、または斜め読みしてはいけません。

`spam.el` シーケンスのイベントの章をちゃんと読みましょう。Section 8.20.7 [Extending the Spam package], page 291 を参照して下さい。

Spam パッケージを使うには、必ず最初に `spam-initialize` 関数を実行させて下さい:

```
(spam-initialize)
```

これは `spam.el` を自動読み込み (autoload) して、Spam パッケージにその仕事をさせるために必要な諸機能が使えるようにします。Spam パッケージを利用するために、いくつかのグループパラメーターと変数を設定しなければなりません。それらは以下で説明します。Spam パッケージを制御するすべての変数は、`'spam'` カスタマイズグループで見つかるでしょう。

Spam パッケージと Gnus には二つの「接点」があります。それは新着メールが `spam` かどうかを検査するときと、グループを抜け出るときです。

新着メールが spam かどうかの検査は、やって来たメールを分割するときか、グループに入るときにどちらかで行なわれます。

最初のやり方、つまりやって来たメールを分割するときに検査をするのは、新着メールが単一のスプールファイルに入れられる `nnml` や `nnimap` のようなメールバックエンドに適しています。Spam パッケージはやって来たメールを処理し、spam と見なすメールを“ spam ”用に指定したグループに送ります。See Section 8.20.2 [Filtering Incoming Mail], page 276.

二番目のやり方は、`nntp` のような (やって来たメールのためのスプールがない) バックエンドや、やって来たメールの分割をサーバーが担当するバックエンドに適しています。この場合 Gnus のグループに入ると、そのグループにあるまだ読まれたことが無い、または未読になっているメッセージに対して spam かどうかの検査が行なわれます。検出された spam メッセージには spam 印が付けられます。See Section 8.20.3 [Detecting Spam in Groups], page 277.

どちらの場合でも、spam メッセージの検出にどの方法を使うかを Spam パッケージに指示しなければなりません。選択肢として複数の方法、と言うか「spam バックエンド」があります (Gnus のバックエンドと混同しないで下さい): spam の「ブラックリスト」と「ホワイトリスト」、辞書に基づいた濾過器、などです。See Section 8.20.6 [Spam Back Ends], page 283.

Gnus の概略バッファで spam だと同定されたメッセージには、常に '\$' 印が付きます。

Spam パッケージは Gnus のグループを三つに分類します: ham グループ、spam グループ、および分類されないグループです。講読している各グループが ham グループと spam グループのどちらなのかを、`spam-contents` グループパラメーターを使って指定して下さい (see Section 2.10 [Group Parameters], page 23)。Spam グループには特別な属性があり、spam グループに入ると、まだ読まれたことが無いすべてのメッセージに spam 印が付きます。そのため、spam グループに分割されたメールには自動的に spam 印が付きます。

Spam メッセージを同定することは、Spam パッケージの仕事の半分に過ぎません。もう半分は、グループを抜け出るときに実行します。このとき Spam パッケージは複数のことを行ないます。

最初に spam か ham かに応じて記事を処理するために *spam and ham processors* を呼び出します。各々の spam バックエンドと関係している spam と ham のプロセッサの対があって、プロセッサが行なうことはバックエンドに依存しています。現在のところ spam と ham プロセッサの主な役割は、辞書に基づいた spam 濾過のためのものです: それらは将来の spam を検出する性能を改良するために、グループにあるメッセージの内容を濾過器の辞書に追加します。`spam-process` グループパラメーターで、どの spam プロセッサを使うかを指定します。See Section 8.20.4 [Spam and Ham Processors], page 278.

Spam 濾過器が spam メッセージに印を付けそこなったら、グループを抜け出るときにそのメッセージが spam として処理されるようにするために、あなた自身がそれに印を付けても良いでしょう。

M-d

M s x

S x 現在の記事に spam 印を付けて、'\$' 印を表示します (`gnus-summary-mark-as-spam`)。

同様に、記事に誤って付けられた spam 印を消すこともできます。See Section 3.7.4 [Setting Marks], page 58.

普通 ham ではないグループで見つかった ham メッセージは ham として処理されません。つまり、さらに処理されるために、それは ham グループに移動させるべきであるということです (以下を見て下さい)。しかし `spam-process-ham-in-spam-groups` および `spam-process-ham-in-nonham-groups` を設定することによって、それらの記事を ham として処理することを強制することもできます。

グループを抜け出るときに、二番目に Spam パッケージが行なうことは、ham 記事を spam グループの外へ、spam 記事を ham グループの外へ移動させることです。Spam グループの ham 記事は、変数 `gnus-ham-process-destinations` またはグループパラメーター `ham-process-destination` で指定されたグループに移動させられます。Ham グループの spam 記事は、変数 `gnus-spam-process-destinations` またはグループパラメーター `spam-process-destination` で指定されたグループに移動させられます。これらの変数が設定されていない場合、記事はそれらの現在のグループに残されます。記事を移動させることができない場合 (例えば NNTP のような読み出し専用のバックエンドでは)、代わりに記事がコピーされます。

記事が別のグループに移動させられると、その新しいグループを訪れたときに、記事は再び処理されます。普通これは問題になりませんが、それぞれの記事が一回だけ処理されるようにしたいならば、`gnus-registry.el` パッケージを読み込んで、変数 `spam-log-to-registry` を `t` に設定して下さい。See Section 8.20.5 [Spam Package Configuration Examples], page 280.

普通 spam グループは `gnus-spam-process-destinations` を無視します。しかし `spam-move-spam-nonspam-groups-only` を `nil` に設定すると、`spam-process-destination` パラメーターに従って spam は spam グループの外へ移動させられます。

最後に Spam パッケージが行なうことは、spam 記事に期限切れ消去の印を付けることです。普通それは正しい行ないです。

これらのすべてがわけがわからなくても、心配は要りません (訳注: でも訳文が正確ではないかもしれないので、変だと思ったら原文を見てね :-p)。すぐにそれは神経インターフェース上に Lisp で小話を書くように自然なことになります... え`ごめん、それにはまだ 50 年早いですね。ただ私たちを信頼して下さい。それは捨てたものではありません。

8.20.2 やって来るメールの濾過

やって来るメールを濾過するために Spam パッケージを使うには、最初に特級メール分割を使うための設定を行なって下さい。See Section 6.3.6 [Fancy Mail Splitting], page 157. Spam パッケージは、特級分割のための変数 (メールバックエンドによるが `nnmail-split-fancy` または `nnimap-split-fancy`) に追加することができる、特別な分割関数を定義します:

```
(: spam-split)
```

`spam-split` 関数は、あなたが選んだ spam バックエンド (一つまたは複数) に応じて、やって来たメールを走査します。デフォルトでは spam グループは 'spam' という名前のグループですが、`spam-split-group` をカスタマイズすることによって変更することができます。`spam-split-group` の値に Gnus のバックエンド名やサーバー名を含めないで下さい。例えば 'your-server' という `nnimap` のサーバーでは、'spam' という名前は 'nnimap+your-server:spam' を意味します。したがって 'nnimap+server:spam' という値は誤りで、それは 'nnimap+your-server:nnimap+server:spam' というグループを指すことになってしまいます。

`spam-split` はいかなる方法によってもメッセージの内容を変更しません。

IMAP の利用者への注意: spam バックエンドとして `spam-check-bogofilter`、`spam-check-ifile` および `spam-check-stat` を使う場合は、さらに変数 `nnimap-split-download-body` を `t` に設定しなければなりません。これらの spam バックエンドがメッセージの本文を「走査」(scan) することができれば、非常に有用です。デフォルトでは `nnimap` バックエンドはメッセージヘッダーだけを取り込みますが、`nnimap-split-download-body` でメッセージの本文も取り込むことを指示することができます。それは IMAP を遅くしてしまうので、デフォルトでは設定されません。そして、利用者に代わってそれを行なうことは、適切な判断ではありません。See Section 6.5.1 [Splitting in IMAP], page 190.

spam-use-* 変数を使って、spam-split が使う一つ以上の spam バックエンドを設定しなければなりません。See Section 8.20.6 [Spam Back Ends], page 283. 通常 spam-use-* は、あなたがこのようにして設定したすべての spam バックエンドを、単に使います。しかし、それらのいくつかだけを使うように、spam-split に指示することもできます。それがなぜ役に立つのかって？ Spam バックエンドとして spam-use-regex-headers と spam-use-blackholes を使っていて、かつ以下の分割規則を使っているとする:

```
nnimap-split-fancy '(|
                    (any "ding" "ding")
                    (: spam-split)
                    ;; デフォルトのメールボックス。
                    "mail")
```

問題は ding 宛てのメッセージをすべて ding フォルダーに入れようとしていることです。でもそれは、ding メーリングリスト宛てに送られた明らかな spam (例えば SpamAssassin と spam-use-regex-headers によって検出される spam) を許してしまうでしょう。一方、いくつかの ding 宛てのメッセージはブラックホールリストに載っているメールサーバーからやって来るので、ding の規則より前に spam-split を発動することができません。

解決策は SpamAssassin ヘッダーに ding の規則を置き換えさせ、ding の規則の後で別の spam-split の規則 (二つ目の正規表現によるヘッダーの検査を含む) を作動させることです。これはパラメーターを spam-split に渡すことによって行なわれます:

```
nnimap-split-fancy
'(|
  ;; spam-use-regex-headers で検出された
  ;; spam は 'regex-spam' へ。
  (: spam-split "regex-spam" 'spam-use-regex-headers)
  (any "ding" "ding")
  ;; spam-split で検出された他のすべての spam は
  ;; spam-split-group へ。
  (: spam-split)
  ;; デフォルトのメールボックス。
  "mail")
```

これは、あなたの特別な必要に応じた特定の spam-split 検査を起動し、それらの検査の結果で特定の spam グループを指し示します。すべてのメールに対して、すべての spam 検査を行なう必要はありません。これが良いもう一つの理由は、分割規則を設定してあるメーリングリスト宛てのメッセージに対して、資源集約的なブラックホール・チェックを実行する必要がないということです。さらに、nnmail の分割のために nnimap のものとは異なる spam 検査のやり方を設定することもできるのでしょう。気が狂うー。

使用するどんな spam バックエンドにも spam-use-* 変数を設定すべきです。そのわけは、'spam.el' を読み込むときに、どんな spam-use-xyz 変数を設定したかによって、何らかの条件付きの読み込みが行なわれるからです。See Section 8.20.6 [Spam Back Ends], page 283.

8.20.3 グループにおける spam の検出

グループに入ったときに spam を検出するためには、そのグループの spam-autodetect と spam-autodetect-methods グループパラメーターを設定して下さい。これらは通常とおり G c か G p で行なうことができます (see Section 2.10 [Group Parameters], page 23).

使用するどんな spam バックエンドにも `spam-use-*` 変数を設定するべきです。そのわけは、`'spam.el'` を読み込むときに、どんな `spam-use-xyz` 変数を設定したかによって、何らかの条件付きの読み込みが行なわれるからです。

デフォルトでは、まだ読まれたことがない記事だけが spam かどうかを検査されます。spam-autodetect-recheck-messages を `t` に設定することによって、グループにあるすべての記事の再検査を Gnus に強制することができます。

Spam の検査に spam-autodetect の手段を使う場合は、異なるグループで違う spam 検出手段を指定することができます。例えば `'ding'` グループは自動検出手段として `spam-use-BBDB` を持つことができる一方で、`'suspect'` グループでは `spam-use-blacklist` および `spam-use-bogofilter` の手段を使うことができます。spam-split と違って検査の順序を制御できませんが、これはたぶん重要ではありません。

8.20.4 Spam と Ham プロセッサー

Spam と ham プロセッサーには、グループバッファを抜け出るときに行なう動作に関して特別な性質があります。Spam プロセッサーは spam メッセージに作用し、ham プロセッサーは ham メッセージに作用するということです。現在のところ、これらのプロセッサーの主な役割は、Bogofilter (see Section 8.20.6.7 [Bogofilter], page 286) や Spam 統計パッケージのような辞書に基づいた spam バックエンドの辞書を更新することです (see Section 8.20.6.10 [Spam Statistics Filtering], page 289)。

それぞれのグループに適用される spam と ham プロセッサーは、そのグループの spam-process グループパラメーターで決定されます。このグループパラメーターが定義されていないと、それらは変数 `gnus-spam-process-newsgroups` によって決められます。

Gnus はあなたが受け取った spam から学びます。あなたは一つ以上の spam グループに spam 記事を集めて、変数 `spam-junk-mailgroups` を適切に設定もしくはカスタマイズしなければなりません。また、spam を含めるグループを、そのグループパラメーター `spam-contents` を `gnus-group-spam-classification-spam` に設定するか、またはそれに対応する変数 `gnus-spam-newsgroup-contents` をカスタマイズすることによって宣言することができます。spam-contents グループパラメーターと `gnus-spam-newsgroup-contents` 変数は、それらの種別を `gnus-group-spam-classification-ham` に設定することによって、ham グループであることを宣言するために使うこともできます。グループが `spam-junk-mailgroups`, `spam-contents` または `gnus-spam-newsgroup-contents` であることを示す分類が行なわれていないと、それらは未分類であると解釈されます。すべてのグループはデフォルトでは未分類です。

Spam グループでは、デフォルトですべてのメッセージが spam であると解釈されます: そのグループに入ると、それらに '\$' 印 (`gnus-spam-mark`) が付きます。あるメッセージを見て、いったんそれに spam の印を付けても、後で取り消せば、その後そのグループに入ったときに、それには spam 印は付きません。spam-mark-only-unseen-as-spam パラメーターを `nil` にすれば、そういう動作をやめさせる、つまりすべての未読メッセージに spam 印が付くようにすることができます。そのグループの概略バッファにいるときに、やっぱり spam ではなかったとわかった記事があったら、それらのすべてから '\$' 印を消さなければなりません。'\$' 印を消すには `M-u` でその記事を「未読」にするか、あるいは `d` を使って spam ではないものとして読んだことを宣言すれば良いでしょう。グループを抜けるとき、すべての spam 印 ('\$') が付いた記事は spam プロセッサーに送られ、それらを spam の標本として学習します。

メッセージは他のいろいろな方法によっても消去されるかもしれないし、ham-marks グループパラメーターが無効にされなければ、'R' 印と 'r' 印、および 'X' 印と 'K' は、'Y' 印と同様に、すべて spam では無い記事に関連付けられるものと解釈されます (それぞれ 'R' はデフォルトの既読の印、

‘r’ 明示的な消去の印、‘X’ は自動的な削除の印、‘K’ は明示的な削除の印、そして ‘Y’ は低いスコアのため印です)。この仮定は、特に真性の spam を検出するために消去 (kill) ファイルかスコアファイルを使っている場合は、間違いかもしれません。そうであれば ham-marks グループパラメーターを調整すべきです。

ham-marks [Variable]

このグループまたはトピックパラメーターを ham であると解釈したい印のリストに設定することができます。デフォルトでは、消去 (deleted)、既読 (read)、削除 (killed)、kill ファイルにあるもの (kill-filed) および低いスコア (low-score、既読だけれども spam ではないと考える) 印のリストです。Ham 印のリストに可視 (tick) 印を含めることが役立つこともあります。未読印を ham 印にすることは、通常それが分類されていないことを表すので、勧められません。しかし、あなたがそれを行なうことはできるし、私たちに不満はありません。

spam-marks [Variable]

このグループまたはトピックパラメーターを spam であると解釈したい印のリストに設定することができます。デフォルトでは spam 印だけを持つリストです。それを変更することは勧めませんが、本当にそうしたいのならご勝手に。

グループを抜けるときに (そのグループが何であっても)、その spam-contents の分類にかかわらず、spam 印が付いているすべての記事は spam プロセッサに送られ、それらを spam の標本として学習します。意図的にたくさんの消去を行なうと、たまにそれは見ていない ‘K’ 印が付いた記事群で終わるかもしれません。そしてそれらは偶然に spam を含んでいるかもしれません。最も良いのは、本当の spam に ‘\$’ が付いていて、他に何も印が無いことを確かめることです。

Spam グループを抜けるときに、spam 印が付いているすべての記事には spam プロセッサで処理した後で期限切れ消去の印が付けられます。これは 未分類 または ham グループに対しては行なわれません。さらに spam グループにあるどの ham 記事も、ham-process-destination グループパラメーターが示す場所か gnus-ham-process-destinations 変数の中で合致する場所のどちらかに移されます。後者はグループ名に合致する正規表現のリストです (*M-x customize-variable* (RET) *gnus-ham-process-destinations* によってこの変数をカスタマイズするのが最も簡単です)。変数を手でカスタマイズの方が好きな人のために言っておくと、それぞれのグループ名のリストは普通の Lisp の list です。ham-process-destination パラメーターが設定されていないと、ham 記事は移動させられません。spam-mark-ham-unread-before-move-from-spam-group パラメーターが設定されていると、ham 記事には移動させられる前に未読の印が付けられます。

例えば NNTP のような読み込み専用バックエンドであるために ham が移動できない場合、それはコピーされます。

グループごとに、または正規表現に合致するグループごとに、複数の移動先を指定できることに注目して下さい! これによって ham 記事を正規のメールグループと ham トレーニング グループに送ることができます。

Ham グループを抜けるときに、ham 印が付いているすべての記事は ham プロセッサに送られ、それらを spam ではない標本として学習します。

変数 spam-process-ham-in-spam-groups はデフォルトでは nil です。Spam グループで見つかった ham がプロセッサに送られるようにしたい場合は t にして下さい。通常これは行なわれません。その代わりに、あなたが自分で ham 記事を ham グループに送って、そこで処理することが想定されています。

変数 spam-process-ham-in-nonham-groups はデフォルトでは nil です。Ham ではない (spam または未分類の) グループで見つかった ham がプロセッサに送られるようにしたい場合は

t にして下さい。通常これは行なわれません。その代わり、あなたが自分で ham 記事を ham グループに送って、そこで処理することが想定されています。

Ham または 未分類 グループを抜けるときに、すべての spam 記事は、spam-process-destination グループパラメーターが示す場所か gnus-spam-process-destinations 変数の中で合致する場所のどちらかに移されます。後者はグループ名に合致する正規表現のリストです (*M-x customize-variable* ([RET](#)) *gnus-spam-process-destinations* によってこの変数をカスタマイズするのが最も簡単です)。変数を手でカスタマイズする方が好きな人のために言っておくと、それぞれのグループ名のリストは普通の Lisp の list です。spam-process-destination パラメーターが設定されていないと、spam 記事は単に期限切れ消去されます。グループ名は完全形であること、すなわちグループバッファでグループ名の前に 'nntp:servername' のようなものが見える場合は、ここでもそれを使う必要があります。

例えば NNTP のような読み込み専用バックエンドであるために spam が移動できない場合、それはコピーされます。

グループごとに、または正規表現に合致するグループごとに、複数の移動先を指定できることに注目して下さい! これによって spam 記事を正規のメールグループと *spam* トレーニング グループに送ることができます。

Ham と spam に関する問題は、Gnus がデフォルトではこの処理を追跡してくれないことです。複数回にわたって処理することを回避するために、spam.el が gnus-registry.el を使って処理された記事を追跡するように、spam-log-to-registry 変数を有効にして下さい。gnus-registry.el が登録する数を制限してしまうと、制限が無い場合のように動作しないことを覚えておいて下さい。

Spam グループにある、まだ読まれたことが無い記事だけに spam の印を付けたい場合は、この変数をセットして下さい。デフォルトではセットされています。これを nil にすると、未読の記事にも spam 印が付けられます。

Ham が spam グループから移動される前に印を消したい場合は、この変数をセットして下さい。これは ham に印を付けるために可視 (tick) 印 ('!') のようなものを使う場合に、とても役に立ちます。記事はあたかもそれがメールサーバーから来たばかりのように、無印で ham-process-destination に置かれるでしょう。

この変数は spam.el が spam の自動検出を行なう場合に、まだ読まれたことが無い記事だけか、またはすべての未読記事のどちらに対して spam 検出を行なうかを指示します。これはそのままにしておくことを勧めます。

8.20.5 Spam パッケージの設定例

Ted の設定

From Ted Zlatanov <tzz@lifelogs.com>.

```
;; gnus-registry-split-fancy-with-parent と spam の自動検出のため。
;; 詳細は 'gnus-registry.el' を参照。
(gnus-registry-initialize)
(spam-initialize)

(setq
 spam-log-to-registry t ;; Spam の自動検出のため。
 spam-use-BBDB t
 spam-use-regex-headers t ;; X-Spam-Flag (SpamAssassin) を捕まえる。
 ;; 名前に 'spam' を含むすべてのグループには spam 記事がある。
```

```

gnus-spam-newsgroup-contents '("spam"
                                gnus-group-spam-classification-spam))
;; これらの docstring を参照。
spam-move-spam-nonspam-groups-only nil
spam-mark-only-unseen-as-spam t
spam-mark-ham-unread-before-move-from-spam-group t
nnimap-split-rule 'nnimap-split-fancy
;; あなたの設定に追加する前に、これが何をするか理解せよ!
nnimap-split-fancy '(|
                    ;; References を親まで辿ってそれらのグループ
                    ;; を入れる。
                    (: gnus-registry-split-fancy-with-parent)
                    ;; これはサーバー側の SpamAssassin タグを捕ま
                    ;; える。
                    (: spam-split 'spam-use-regex-headers)
                    (any "ding" "ding"))
                    ;; Spam 記事はデフォルトで 'spam' に行く
                    ;; ことに注意。
                    (: spam-split)
                    ;; デフォルトのメールボックス。
                    "mail"))

;; G p で設定した私のパラメーター。

;; すべての nnml グループと、'nnimap+mail.lifelogs.com:train' と
;; 'nnimap+mail.lifelogs.com:spam' を除いたすべての nnimap グループ
;; のためのパラメーター:
;; それは手動で検出したはずなので、どの spam も nnimap のトレーニング
;; グループに送り込む。

((spam-process-destination . "nnimap+mail.lifelogs.com:train"))

;; すべての NNTP グループのためのパラメーター:
;; Spam を blacklist で、ham を BBDB で自動検出。
((spam-autodetect-methods spam-use-blacklist spam-use-BBDB)
;; すべての spam をトレーニンググループに送る。
 (spam-process-destination . "nnimap+mail.lifelogs.com:train"))

;; 私が spam を自動検出させたい、ほんのいくつかの NNTP グループ
;; のためのパラメーター:
((spam-autodetect . t))

;; 私の nnimap 'nnimap+mail.lifelogs.com:spam' グループ (これは
;; spam グループ) のためのパラメーター:

((spam-contents gnus-group-spam-classification-spam)

```

```
;; どんな spam も私が ham 印を付けなければ
;; 'nnimap+mail.lifelogs.com:train' に送り込まれる。(前述の
;; gnus-spam-newsgroup-contents の設定により、すべての
;; まだ読まれたことが無いメッセージを読むとそうなる。)

(spam-process-destination "nnimap+mail.lifelogs.com:train")

;; どんな ham も私の 'nnimap+mail.lifelogs.com:mail' フォルダー
;; に送り込まれるが、私の 'nnimap+mail.lifelogs.com:trainham'
;; フォルダーにもトレーニングのために送り込む。

(ham-process-destination "nnimap+mail.lifelogs.com:mail"
                          "nnimap+mail.lifelogs.com:trainham")
;; このグループでは '!' 印が付いているものだけが ham。
(ham-marks
 (gnus-ticked-mark))
;; グループを抜けるときに blacklist に送信者を覚えさせる—これは
;; 明らかに不要で、単に私の鬱憤を晴らすためにある。
(spam-process (gnus-group-spam-exit-processor-blacklist)))

;; その後 IMAP サーバー上で、私は SpamAssassin が spam を
;; 認識するトレーニングのために 'train' グループを、ham を
;; 認識するトレーニングのために 'trainham' グループを使う。
;; でも Gnus はそういうことはやってくれない。
```

サーバー上の IMAP サーバーで、統計的な濾過器と spam.el を使う

From Reiner Steib <reiner.steib@gmx.de>.

私のプロバイダーは (IMAP) メールサーバー上で (DCC と連係した) bogofilter を立ち上げました。認識された spam は 'spam.detected' へ行き、残りには通常の濾過規則が適用される、すなわち 'some.folder' か 'INBOX' に行きます。誤検出と見逃しのトレーニングは、それぞれ 'training.ham' または 'training.spam' に記事をコピーするか移動させることによって行なわれます。サーバー上の cron ジョブが、それらを適切な ham または spam オプションとともに bogofilter に与え、'training.ham' および 'training.spam' フォルダーからそれらを削除します。

以下の gnus-parameters の要素群によって、spam.el はほとんどの仕事を私のためにこなします:

```
("nnimap:spam\\.detected"
 (gnus-article-sort-functions '(gnus-article-sort-by-chars))
 (ham-process-destination "nnimap:INBOX" "nnimap:training.ham")
 (spam-contents gnus-group-spam-classification-spam))
("nnimap:\\(INBOX\\|other-folders\\)")
 (spam-process-destination . "nnimap:training.spam")
 (spam-contents gnus-group-spam-classification-ham))
```

- **The Spam folder:** 'spam.detected' フォルダーにおいて、私は誤検出 (すなわち bogofilter が DCC が spam であると誤って判定した正当なメール) のチェックをしなければなりません。gnus-group-spam-classification-spam の項のために、すべてのメッセージには spam の印 (\$) が付けられます。誤検出を見つけたら、私は記事にいくつかの他の ham 印 (ham-marks,

Section 8.20.4 [Spam and Ham Processors], page 278) を付けます。グループを出るとき、それらの記事は ‘INBOX’ (私が記事を置いておきたいところ) と ‘training.ham’ (bogofilter のトレーニング用) の両方のグループにコピーされ、‘spam.detected’ フォルダーから削除されます。

gnus-article-sort-by-chars の項は、私の誤検出の発見を簡単にします。私は、すべて似たサイズの、たくさんのワーム (sweN, ...) を受け取ります。それらをサイズ (つまり文字数) でまとめると、他の誤検出を見つけやすくなるのです。(もちろん厳密にはワームは *spam* (UCE, UBE) ではありません。ともあれ、それらの要らないメールを濾過するのに bogofilter は私にとって優秀な道具です。)

- **Ham folders:** 私の ham フォルダーで、認識されなかった spam メール (見逃し) を見つけたときはいつでも、私は単に *S x* (gnus-summary-mark-as-spam) を叩きます。グループを出るとき、それらのメッセージは ‘training.spam’ に移されます。

spam-report.el で Gmane グループの spam を報告する

From Reiner Steib <reiner.steib@gmx.de>.

以下の gnus-parameters に納めた要素によって、*S x* (gnus-summary-mark-as-spam) で gmane.* グループの spam 記事に印を付け、グループを出るときに Gmane に報告します:

```
("^gmane\\.")
(spam-process (gnus-group-spam-exit-processor-report-gmane)))
```

加えて、私は news.gmane.org からではなくローカルニュースサーバー (leafnode) を通して記事を読んでいるので、(setq spam-report-gmane-use-article-number nil) を使っています。つまり、記事番号が news.gmane.org におけるものと異なるので、正しい記事番号を見つけるために spam-report.el に X-Report-Spam ヘッダーを検査させなければなりません。

8.20.6 Spam バックエンド

Spam パッケージは spam を検出するための様々なバックエンドを提供します。それぞれのバックエンドでは、spam を検出する手段の組 (see Section 8.20.2 [Filtering Incoming Mail], page 276, see Section 8.20.3 [Detecting Spam in Groups], page 277) と spam および ham プロセッサーの対が定義されています (see Section 8.20.4 [Spam and Ham Processors], page 278)。

8.20.6.1 ブラックリストとホワイトリスト

spam-use-blacklist [Variable]

入ってくるメールを分割するときにブラックリストを使いたい場合は、この変数を *t* に設定して下さい。送信者がブラックリストに載っているメッセージは spam-split-group に送られます。これは、送信者が spammer であることが定義されているメールに対してだけ動作する、明示的な濾過器です。

spam-use-whitelist [Variable]

入ってくるメールを分割するときにホワイトリストを使いたい場合は、この変数を *t* に設定して下さい。送信者がホワイトリストに載っていないメッセージは、次の spam-split 規則 (による検査工程) に送られます。これは、ホワイトリストに載っていない誰かのメッセージは spam とともに ham とも見なされないことを意味する、明示的な濾過器です。

spam-use-whitelist-exclusive [Variable]

送信者がホワイトリストに載っていないすべてのメッセージが spam だと見なされることを意味する暗黙の濾過器としてホワイトリストを使いたい場合は、この変数を *t* にして下さい。注意して使して下さい。

`gnus-group-spam-exit-processor-blacklist` [Variable]

このシンボルを、グループパラメーターのカスタマイズによってグループの `spam-process` パラメーターに加えるか、または `gnus-spam-process-newsgroups` 変数に加えて下さい。このシンボルがグループの `spam-process` パラメーターに加えられると、spam 印が付いた記事の送信者がブラックリストに追加されます。

警告

旧式の `gnus-group-spam-exit-processor-blacklist` の代わりに (`spam spam-use-blacklist`) を使うことを推奨します。すべて同等に動作することは保証されます。

`gnus-group-ham-exit-processor-whitelist` [Variable]

このシンボルを、グループパラメーターのカスタマイズによってグループの `spam-process` パラメーターに加えるか、または `gnus-spam-process-newsgroups` 変数に加えて下さい。このシンボルがグループの `spam-process` パラメーターに加えられると、ham 印が付いた *ham* グループの記事の送信者がホワイリストに追加されます。

警告

旧式の `gnus-group-ham-exit-processor-whitelist` の代わりに (`ham spam-use-whitelist`) を使うことを推奨します。すべて同等に動作することは保証されます。

ブラックリストは、あなたが spam の送信者だと考えるアドレスに合致する正規表現のリストです。例えば `'vmail.com'` の誰からでも来るメールを阻止するには、あなたのブラックリストに `'vmail.com'` を入れて下さい。空のブラックリストで始めましょう。ブラックリストの各項目は Emacs の正規表現の構文を使います。

逆に、ホワイリストは何のアドレスが正当だと考えられるかを告げます。ホワイリストにあるアドレスからやって来たすべてのメッセージは、非-spam だと見なされます。Section 8.20.6.2 [BBDB Whitelists], page 284 も見て下さい。ホワイリストの各項目は Emacs の正規表現の構文を使います。

ブラックリストとホワイリストのファイルの所在は、`spam-directory` 変数 (デフォルトは `'~/News/spam'`) または直接 `spam-whitelist` と `spam-blacklist` 変数でカスタマイズすることができます。ホワイリストとブラックリストのファイルは、デフォルトでは `spam-directory` のディレクトリーにあり、それぞれ `'whitelist'` と `'blacklist'` という名前が付けられます。

8.20.6.2 BBDB ホワイリスト

`spam-use-BBDB` [Variable]

`spam-use-whitelist` (see Section 8.20.6.1 [Blacklists and Whitelists], page 283) に似ていますが、ホワイリストのアドレスの源として BBDB を使います。正規表現はありません。spam-use-BBDB をちゃんと動作させるには BBDB を読み込まなければ (load しなければ) なりません。その送信者が BBDB に載っていないメッセージは、次の `spam-split` 規則 (による検査工程) に送られます。これは、BBDB に載っていない誰かのメッセージは spam とも ham とも見なされないことを意味する、明示的な濾過器です。

`spam-use-BBDB-exclusive` [Variable]

送信者が BBDB に載っていないすべてのメッセージが spam だと見なされることを意味する暗黙の濾過器として BBDB を使いたい場合は、この変数を `t` にして下さい。注意して使して下さい。BBDB に載っている送信者だけが通行を許され、他のすべては spammers として分類されます。

spam.el に関する限りは、spam-use-BBDB の別名として spam-use-BBDB-exclusive を使うことができますが、それは別のバックエンドではありません。spam-use-BBDB-exclusive を t に設定すれば、BBDB による分割はすべて排他的になります。

gnus-group-ham-exit-processor-BBDB [Variable]

このシンボルを、グループパラメーターのカスタマイズによってグループの spam-process パラメーターに加えるか、または gnus-spam-process-newsgroups 変数に加えて下さい。このシンボルがグループの spam-process パラメーターに加えられると、ham 印が付いた ham グループの記事の送信者が BBDB に追加されます。

警告

旧式の gnus-group-ham-exit-processor-BBDB の代わりに、(ham spam-use-BBDB) を使うことを推奨します。すべて同等に動作することは保証されます。

8.20.6.3 Gmane Spam 報告

gnus-group-spam-exit-processor-report-gmane [Variable]

グループパラメーターか変数 gnus-spam-process-newsgroups をカスタマイズして、このシンボルをグループの spam-process パラメーターに加えて下さい。これに加えられると、spam 印が付いた記事のグループが HTTP 経由で Gmane の管理者に報告されます。

Gmane は <http://gmane.org> で見つけることができます。

警告

旧式の gnus-group-spam-exit-processor-report-gmane の代わりに (spam spam-use-gmane) を使うことを推奨します。すべて同等に動作することは保証されます。

spam-report-gmane-use-article-number [Variable]

この変数はデフォルトで t です。例えばあなた自身がニュースサーバーを運営しているなどの理由によって、ローカルな記事番号が Gmane の記事番号と合わない場合は、nil に設定して下さい。spam-report-gmane-use-article-number が nil であると、spam-report.el はその番号の記事のヘッダーから取得します。

spam-report-user-mail-address [Variable]

Gmane への spam の報告に付加される User-Agent に現れるメールアドレスです。これは、誤った報告が行なわれたときに、Gmane の管理者があなたに連絡できるようにするためのものです。デフォルトは user-mail-address です。

8.20.6.4 非-spam Hashcash 印

spam-use-hashcash [Variable]

spam-use-whitelist (see Section 8.20.6.1 [Blacklists and Whitelists], page 283) に似ていますが、送信者のアドレスの代わりに、潔白なメッセージの証しとして hashcash の印 (tokens) を使います。Hashcash 印が無いメッセージは次の spam-分割 (spam-split) 規則 (による検査工程) に送られます。これは hashcash 印が見当たらないメッセージは spam とも ham とも見なされないことを意味する、明示的な濾過器です。

8.20.6.5 ブラックホール

spam-use-blackholes [Variable]

このオプションはデフォルトで無効になっています。このオプションをセットすると、Gnus にブラックホール型の分散 spam 処理システム (例えば DCC) を調べさせることができます。

変数 `spam-blackhole-servers` は、Gnus が意見を求めるブラックホール・サーバーのリストを持ちます。現在のリストはかなり広範囲に渡っていますが、もし時代遅れなサーバーを含んでいたら必ず私たちに知らせるようにして下さい。

ブラックホール・チェックは `'dig.el'` パッケージを使います。しかし `spam-use-dig` を `nil` に設定すれば、より良い性能のために `'dns.el'` を代わりに使うことを `spam.el` に指示することができます。現状では `spam-use-dig` を `nil` に設定することは、いく人かの利用者が使えないかもしれないので、それが可能な性能改善であるにもかかわらず推奨されません。しかし、それが動くかどうかを確かめることはできます。

`spam-blackhole-servers` [Variable]
ブラックホール・チェックのために意見を求めるサーバーのリストです。

`spam-blackhole-good-server-regex` [Variable]
ブラックホール・サーバーのリストと照合されてはならない IP の正規表現です。 `nil` に設定されると無効になります。

`spam-use-dig` [Variable]
`'dns.el'` パッケージの代わりに `'dig.el'` パッケージを使います。デフォルトの設定である `t` が推奨されます。

ブラックホール・チェックは入って来るメールに対してだけ行なわれます。ブラックホールに `spam` または `ham` プロセッサはありません。

8.20.6.6 正規表現によるヘッダーの合致検査

`spam-use-regex-headers` [Variable]
このオプションはデフォルトで無効になっています。このオプションをセットすると、Gnus に正規表現のリストとメッセージヘッダーを照合させることができます。変数 `spam-regex-headers-spam` および `spam-regex-headers-ham` が正規表現のリストを持ちます。メッセージが `spam` か `ham` かどうかをそれぞれの変数を使って決めるために、Gnus はメッセージヘッダーを検査します。

`spam-regex-headers-spam` [Variable]
メッセージヘッダーの中で一致した時に、それが `spam` であることを断定するための正規表現のリストです。

`spam-regex-headers-ham` [Variable]
メッセージヘッダーの中で一致した時に、それが `ham` であることを断定するための正規表現のリストです。

正規表現によるヘッダーの検査は、入ってきたメールに対してだけ行なわれます。正規表現のために特有な `spam` または `ham` プロセッサはありません。

8.20.6.7 Bogofilter

`spam-use-bogofilter` [Variable]
Eric Raymond の迅速な Bogofilter を `spam-split` に使用したい場合は、この変数をセットして下さい。

Spam 記事に '\$' 印を関連付ける最小限度の世話だけで、Bogofilter トレーニングはすべてかなり自動的にになります。Spam とそうでないもののそれぞれの種類について数百通ずつの記事

を入手するまで、これをやらなければなりません。デバッグまたは好奇心のどちらかのために概略モードで `st` コマンドを使うことによって、現在の記事の *spam* 度 (spamicity) スコア (0.0 ~ 1.0) を表示させることができます。

Bogofilter はメッセージが spam かどうかを、ある明確な閾値に基づいて見極めます。閾値はカスタマイズできます。Bogofilter のドキュメントを調べて下さい。

Path に bogofilter の実行ファイルが無い場合、Bogofilter の処理は取り消されます。

`spam-use-bogofilter-headers` を使う場合は、これを有効にはいけません。

M s t

S t Bogofilter の spam 度スコアを得ます (`spam-bogofilter-score`)。

`spam-use-bogofilter-headers` [Variable]

メッセージヘッダーだけを調べるために Eric Raymond の迅速な Bogofilter を `spam-split` に使用したい場合は、この変数をセットして下さい。これは `spam-use-bogofilter` と同じように動作しますが、あらかじめ `X-Bogosity` ヘッダーがメッセージに存在しなければなりません。通常これは `procmail` の技法か、何かそれに似たもので行なうことになるでしょう。Bogofilter のインストールに関する文書を調べて下さい。

`spam-use-bogofilter` を使う場合は、これを有効にはいけません。

`gnus-group-spam-exit-processor-bogofilter` [Variable]

このシンボルを、グループパラメーターのカスタマイズによってグループの `spam-process` パラメーターに加えるか、または `gnus-spam-process-newsgroups` 変数に加えて下さい。このシンボルがグループの `spam-process` パラメーターに加えられると、spam 印が付いた記事が bogofilter の spam データベースに加えられます。

警告

旧式の `gnus-group-spam-exit-processor-bogofilter` の代わりに (`spam spam-use-bogofilter`) を使うことを推奨します。すべて同等に動作することは保証されます。

`gnus-group-ham-exit-processor-bogofilter` [Variable]

このシンボルを、グループパラメーターのカスタマイズによってグループの `spam-process` パラメーターに加えるか、または `gnus-spam-process-newsgroups` 変数に加えて下さい。このシンボルがグループの `spam-process` パラメーターに加えられると、ham 印が付いた *ham* グループの記事が非-spam 記事用の Bogofilter データベースに追加されます。

警告

旧式の `gnus-group-ham-exit-processor-bogofilter` の代わりに (`ham spam-use-bogofilter`) を使うことを推奨します。すべて同等に動作することは保証されます。

`spam-bogofilter-database-directory` [Variable]

これは Bogofilter がそのデータベースを格納するディレクトリーです。デフォルトでは設定されていないので、Bogofilter はそれ自身のデフォルトのデータベース・ディレクトリーを使います。

Bogofilter のメール分類器は、意図と目的の点で `ifile` に似ています。Ham および spam のプロセッサが提供され、記事で Bogofilter が使われるべきか、または既に使われたかを `spam-split` に示すための `spam-use-bogofilter` 変数と `spam-use-bogofilter-headers` があります。この機能を検査するために Bogofilter のバージョン 0.9.2.1 が使われました。

8.20.6.8 SpamAssassin back end

`spam-use-spamassassin` [Variable]

`spam-split` に SpamAssassin を使いたい場合は、この変数をセットして下さい。

SpamAssassin は、規則と分析のセット (ベイジアンフィルタを含む) に基づいて、それぞれの記事のスコアを裁定します。ベイジアンフィルタは、spam 記事に '\$' 印に関連させることによって訓練することができます。Spam のスコアは、概略モードで `st` コマンドを使うことによって見るすることができます。

この変数をセットすると、それぞれの記事は `spam-split` が呼ばれるときに SpamAssassin によって処理されます。メールが SpamAssassin で処理されるようになっていて、SpamAssassin ヘッダーだけを使いたいのであれば、代わりに `spam-use-spamassassin-headers` をセットして下さい。

`spam-use-spamassassin-headers` を使う場合、これを有効にはいけません。

`spam-use-spamassassin-headers` [Variable]

メールが SpamAssassin で処理されるようになっている場合に、SpamAssassin ヘッダーに基づいて `spam-split` に分割を行なわせたいのであれば、この変数をセットして下さい。

`spam-use-spamassassin` を使う場合、これを有効にはいけません。

`spam-spamassassin-program` [Variable]

この変数は SpamAssassin の実行形式を指します。`spamd` を稼働しているならば、より速い処理のために、この変数に `spamc` の実行形式を設定することができます。`spamd/spamc` の更なる情報は、SpamAssassin のドキュメントを見て下さい。

SpamAssassin は、spam を同定するために広範な分析を行なう、強力で融通性のある spam 濾過器です。Ham および spam のプロセッサが提供され、記事で SpamAssassin が使われるべきか、または既に使われたかを `spam-split` に示すための `spam-use-spamassassin` 変数と `spam-use-spamassassin-headers` 変数があります。この機能を検査するために SpamAssassin のバージョン 2.63 が使われました。

8.20.6.9 ifile による spam の濾過

`spam-use-ifile` [Variable]

Bogofilter に似た統計分析器である ifile を `spam-split` に使いたい場合は、この変数を有効にして下さい。

`spam-ifile-all-categories` [Variable]

`spam-use-ifile` に、単なる spam/非-spam ではなくて ifile のすべての区分 (カテゴリー) を与えてもらいたいならば、この変数を有効にして下さい。これを使う場合は、その文献に書かれているように ifile をトレーニングしておかなければなりません。

`spam-ifile-spam-category` [Variable]

ifile に関する限り、これは spam メッセージのカテゴリーです。実際に使われる文字列は無関係ですが、たぶんあなたは 'spam' のデフォルト値を残しておきたいでしょう。

`spam-ifile-database` [Variable]

これは ifile データベースのファイル名です。デフォルトでは定義されていないので、ifile はそれ自身のデフォルトのデータベース名を使います。

ifile のメール分類器は、意図と目的の点で Bogofilter に似ています。Spam と ham のプロセッサが提供され、ifile が使われるべきであることを spam-split に示すための spam-use-ifile 変数があります。この機能を検査するために ifile のバージョン 1.2.1 が使われました。

8.20.6.10 Spam 統計濾過

このバックエンドは、統計に基づいた濾過を行なう Spam 統計 Emacs Lisp パッケージを使います (see Section 8.20.8 [Spam Statistics Package], page 292)。これを使う前に、あなたの Spam 統計辞書を初期化するための、いくつかの追加の処理を行なう必要があるでしょう。See Section 8.20.8.1 [Creating a spam-stat dictionary], page 293。

spam-use-stat [Variable]

gnus-group-spam-exit-processor-stat [Variable]

このシンボルを、グループパラメーターのカスタマイズによってグループの spam-process パラメーターに加えるか、または gnus-spam-process-newsgroups 変数に加えて下さい。このシンボルがグループの spam-process パラメーターに加えられると、spam 印が付いた記事が spam 記事用の spam-stat データベースに追加されます。

警告

旧式の gnus-group-spam-exit-processor-stat の代わりに (spam spam-use-stat) を使うことを推奨します。すべて同等に動作することは保証されます。

gnus-group-ham-exit-processor-stat [Variable]

このシンボルを、グループパラメーターのカスタマイズによってグループの spam-process パラメーターに加えるか、または gnus-spam-process-newsgroups 変数に加えて下さい。このシンボルがグループの spam-process パラメーターに加えられると、ham 印が付いた ham グループの記事が非-spam 記事用の spam-stat データベースに追加されます。

警告

旧式の gnus-group-ham-exit-processor-stat の代わりに (ham spam-use-stat) を使うことを推奨します。すべて同等に動作することは保証されます。

これは spam.el が 'spam-stat.el' と働き合うことを可能にします。'spam-stat.el' は (Lisp だけの) spam 内部データベースを提供しますが、それは ifile や Bogofilter と違って外部プログラムを必要としません。Spam と ham のプロセッサ、および spam-split のための spam-use-stat 変数が提供されます。

8.20.6.11 Gnus で SpamOracle を使うには

気軽に spam を濾過する一つのやり方は SpamOracle を使うことです。統計的にメールを濾過するための道具である SpamOracle は、Xavier Leroy によって書かれました。これは別にインストールする必要があります。

Gnus で SpamOracle を使うには、複数のやり方があります。すべての場合に、メールは mark モードで動作している SpamOracle にパイプされます。すると SpamOracle は、そのメールを spam だと見なしたかどうかを示す 'X-Spam' ヘッダーを記入します。

実現可能な一つは、SpamOracle を :prescript として Section 6.3.4.1 [Mail Source Specifiers], page 148 から走らせることです。この方法には、利用者が X-Spam ヘッダーを見ることができるという利点があります。

もっとも手軽な方法は、'spam.el' (see Section 8.20 [Spam Package], page 274) が SpamOracle を呼ぶようにすることです。

`spam.el` で SpamOracle を利用できるようにするためには、変数 `spam-use-spamoracle` を `t` にして、`nnmail-split-fancy` または `nnimap-split-fancy` を設定して下さい。See Section 8.20 [Spam Package], page 274. この例では `nnimap` サーバーの `'INBOX'` が SpamOracle を使って濾過されます。Spam であると認定されたメールは、`spam-split-group` (この場合は `'Junk'`) に移動させられます。Ham なメッセージは `'INBOX'` に残ります:

```
(setq spam-use-spamoracle t
      spam-split-group "Junk"
      nnimap-split-inbox '("INBOX")
      nnimap-split-rule 'nnimap-split-fancy
      nnimap-split-fancy '(| (: spam-split) "INBOX"))
```

`spam-use-spamoracle` [Variable]
Gnus に SpamOracle を使って spam の濾過をさせたい場合に `t` にして下さい。

`spam-spamoracle-binary` [Variable]
Gnus は利用者の PATH で見つかった `'spamoracle'` という SpamOracle のバイナリーを使います。これにはカスタマイズ可能な変数 `spam-spamoracle-binary` を使います。

`spam-spamoracle-database` [Variable]
SpamOracle はその解析結果をデータベースとして格納するために、デフォルトで `'~/spamoracle.db'` ファイルを使います。これは変数 `spam-spamoracle-database` で制御され、デフォルトは `nil` です。それは、デフォルトの SpamOracle データベースが使われることを意味します。データベースがどこか特別な場所に置きたい場合は、`spam-spamoracle-database` をそのパスに設定して下さい。

SpamOracle はメッセージが spam か ham かを見極めるために統計的な手法を使います。間違いや見逃しの少ない良い結果を得るために、SpamOracle はトレーニングを必要とします。SpamOracle は spam メールの特徴を学びます。`add` モード (トレーニング・モード) を使って、良いメール (ham) と spam を SpamOracle に与えなければなりません。これは、概略バッファで `|` を押すことによってメールを SpamOracle のプロセスにパイプするか、または `'spam.el'` の spam および ham プロセッサを使うことによって行なうことができます。See Section 8.20 [Spam Package], page 274.

`gnus-group-spam-exit-processor-spamoracle` [Variable]
このシンボルを、グループパラメーターのカスタマイズによってグループの `spam-process` パラメーターに加えるか、または `gnus-spam-process-newsgroups` 変数に加えて下さい。このシンボルがグループの `spam-process` パラメーターに加えられると、spam 印が付いた記事が spam のサンプルとして SpamOracle に送られます。

警告

旧式の `gnus-group-spam-exit-processor-spamoracle` の代わりに (`spam spam-use-spamoracle`) を使うことを推奨します。すべて同等に動作することは保証されます。

`gnus-group-ham-exit-processor-spamoracle` [Variable]
このシンボルを、グループパラメーターのカスタマイズによってグループの `spam-process` パラメーターに加えるか、または `gnus-spam-process-newsgroups` 変数に加えて下さい。このシンボルがグループの `spam-process` パラメーターに加えられると、ham グループにある ham 印が付いた記事が ham の記事のサンプルとして SpamOracle に送られます。

警告

旧式の `gnus-group-ham-exit-processor-spamoracle` の代わりに (`ham spam-use-spamoracle`) を使うことを推奨します。すべて同等に動作することは保証されます。

例: これらは ham グループとして分類された、つまり ham の記事しかないグループのためのグループパラメーターです。

```
((spam-contents gnus-group-spam-classification-ham)
 (spam-process ((ham spam-use-spamoracle)
                (spam spam-use-spamoracle))))
```

このグループでは spam-use-spamoracle が ham と spam の両方の処理を行ないます。このグループに spam 記事があって (SpamOracle が十分なサンプルを食べさせてもらっていないければ、そうなりますね)、かつ利用者がいくつかの記事に spam の印を付けたならば、それらの記事は SpamOracle によって処理されます。そのプロセッサは、新しい spam のサンプルとして SpamOracle に記事を送ります。

8.20.7 Spam パッケージの拡張

Blackbox という新しいバックエンドを追加したいとしましょう。入ってくるメールを濾過するために以下のものを用意して下さい:

1. コード

```
(defvar spam-use-blackbox nil
  "Blackbox を使うときは t にする。")
```

Blackbox が入ってくるメールを検査できるのであれば、spam-check-blackbox を書いて下さい。

Blackbox が spam と ham を登録または登録解除できるのであれば、手始めに bogofilter の登録/解除ルーチンを使って、またはもっと Blackbox にふさわしい他の登録/解除ルーチンを使って spam-blackbox-register-routine および spam-blackbox-unregister-routine を書いて下さい。

2. 関数

spam-check-blackbox 関数は、他の習慣に倣って 'nil' が spam-split-group を返さなければなりません。あなたに何ができるかの例として、既存の spam-check-* 関数を参照して下さい。また、あなたがそうでない理由を完全に理解していないならば、テンプレートに齧り付いて下さい。

Spam と ham メッセージを処理するために、以下のものを用意して下さい:

1. コード

Spam または ham のプロセッサを用意する必要はありません。Blackbox が spam または ham の処理をサポートする場合だけ、それらを用意して下さい。

さらに ham と spam のプロセッサは単一の変数ではなくされつつあり、代わりに (spam spam-use-blackbox) または (ham spam-use-blackbox) の形式が推奨されます。今のところ spam/ham プロセッサ変数はまだあちこちにありますが、長く存続することはないでしょう。

```
(defvar gnus-group-spam-exit-processor-blackbox "blackbox-spam"
  "概略を出るときに呼ばれる blackbox の spam プロセッサ。
  Spam グループだけに適用される。")
```

```
(defvar gnus-group-ham-exit-processor-blackbox "blackbox-ham"
  "概略を出るときに呼ばれる blackbox の ham プロセッサ。
  Spam ではない (未分類または ham) グループだけに適用される。")
```

2. Gnus のパラメーター

gnus.el にあるグループパラメーター spam-process に

```
(const :tag "Spam: Blackbox" (spam spam-use-blackbox))
(const :tag "Ham: Blackbox" (ham spam-use-blackbox))
```

を加えて下さい。それを必ずパラメーターと変数のカスタマイズの二回について行なうようにして下さい。

Blackbox が入ってくるメールが spam かどうかを検査できるのであれば、

```
(variable-item spam-use-blackbox)
```

を gnus.el のグループパラメーター spam-autodetect-methods に加えて下さい。

最後に、spam.el にある適切な spam-install-*-backend 関数を使って下さい。利用できる関数は次の通りです。

1. spam-install-backend-alias

この関数は、オリジナルのバックエンドのようにすべてを行なうバックエンドのために、別名を使うことができるようにするだけです。今のところ、これは spam-use-BBDB-exclusive を spam-use-BBDB のように働かせるためだけに使われます。

2. spam-install-nocheck-backend

この関数は、検査する機能は無いけれども ham または spam を登録/解除することができるバックエンドになります。spam-use-gmane がそのようなバックエンドです。

3. spam-install-checkonly-backend

この関数は、入ってくるメールが spam かどうかの検査だけを行なうことができるバックエンドになります。それはメッセージを登録または登録解除することができません。spam-use-blackholes と spam-use-hashcash がそのようなバックエンドです。

4. spam-install-statistical-checkonly-backend

この関数は、入ってくるメールの検査だけを行なうことができる、統計処理を行なうバックエンド（検査のためにメッセージの本文全体を必要とします）になります。spam-use-regex-body がそのような濾過器です。

5. spam-install-statistical-backend

この関数は、入ってくるメールの検査と登録/解除ルーチンを持つ、統計処理を行なうバックエンドになります。spam-use-bogofilter は、そのように仕立てられています。

6. spam-install-backend

これは最も普通のバックエンドになります。それは検査とメッセージの登録/解除を行なうことができ、統計処理の能力はありません。spam-use-BBDB がそのようなバックエンドです。

7. spam-install-mover-backend

移動させる (mover) バックエンドは spam.el の内部で、概略バッファを出るときにある明確なやり方で記事を移動させます。おそらくそのようなバックエンドを使うことは無いでしょう。

8.20.8 Spam 統計パッケージ

Paul Graham は統計を使った spam の濾過に関する優れたエッセイを書きました: A Plan for Spam (<http://www.paulgraham.com/spam.html>)。そこで彼は SpamAssassin によって使われているような規則ベースの濾過に固有な欠陥について述べています。例えば: 誰かが規則を書かな

けれどもならないし、他のすべての人はこれらの規則をインストールしなければなりません。あなたはいつも遅れをとってしまいます。それよりも、それが spam または非-spam に何となく似ているかどうかに基づいてメールを濾過する方が遥かに良いだろうと彼は主張しています。これを測定する一つの手段は単語の分布です。その後で彼は、新着メールがあなたの他の spam メールに似ているかどうかをチェックする方法を述べています。

基本的な考えはこうです：あなたのメールの二つの集合を作ります。一方は spam を、もう一方は spam ではないメールを集めたものです。両方の集合における各単語の出現頻度を数えて、集合のメールの総数で重み付けを行ない、この情報を辞書に格納します。新着メールのすべての単語について、spam か spam ではないメールに属する確率を判定します。15 の最も顕著な単語を使って、そのメールが spam であることの確率の総計を計算します。この確率がある閾値より高かったら、そのメールは spam であると見なされます。

Spam 統計パッケージは、この種の濾過のためのサポートを Gnus に追加します。Spam パッケージ (see Section 8.20 [Spam Package], page 274) のバックエンドの一つとして、またはそれ自体を使うことができます。

Spam 統計パッケージを使う前に、それを使うための準備を行なう必要があります。第一に、あなたのメールの二つの集合が要ります。一方は spam を、もう一方は spam ではないメールを集めたものです。そして、それらの二つの集合を使って辞書を作り、それをセーブして下さい。そして特に最後に、あなたの特級分割の規則でこの辞書を使って下さい。

8.20.8.1 spam-統計 (spam-stat) 辞書を作る

統計に基づいた spam 濾過を始めるには、前もって二つのメールの集合（一方は spam を、もう一方は spam ではないメールを集めたもの）に基づいた統計を作らなければなりません。そしてそれらの統計は、後で利用するために辞書に格納されます。それらの統計を意味のあるものにするために、両方の集合につき数百通のメールが必要です。

今のところ Gnus は nnml バックエンドでだけ辞書の自動生成をサポートします。nnml バックエンドは一通のメールにつき一つのファイルで、すべてのメールを一つのディレクトリーに格納します。以下を使って下さい：

`spam-stat-process-spam-directory` [Function]
このディレクトリーにあるすべてのファイルについて spam の統計を生成します。すべてのファイルは一つの spam メールとして扱われます。

`spam-stat-process-non-spam-directory` [Function]
このディレクトリーにあるすべてのファイルについて非-spam の統計を生成します。すべてのファイルは一つの spam ではないメールとして扱われます。

普通は ‘~/Mail/mail/spam’ のようなディレクトリー（通常 ‘nnml:mail.spam’ グループに対応）について `spam-stat-process-spam-directory` を呼ぶことになるでしょう。また ‘~/Mail/mail/misc’ のようなディレクトリー（通常 ‘nnml:mail.misc’ グループに対応）について `spam-stat-process-non-spam-directory` を呼びましょう。

IMAP を使っている場合はメールをローカルには持っていないので、それは働きません。一つの解決策は、Gnus エージェントで記事をキャッシュすることです。そうすれば `spam-stat-process-spam-directory` として ‘~/News/agent/nnimap/mail.yourisp.com/personal_spam’ のようなものを使うことができます。See Section 6.9.5 [Agent as Cache], page 219.

`spam-stat` [Variable]

この変数はすべての統計のハッシュテーブル—辞書と言っているもの—を保持します。このハッシュテーブルは、双方の集合のすべての単語について `spam` および `spam` ではないメールにおける出現頻度を表すベクトルを格納します。

統計を最初から作り直したいときは、辞書をリセットする必要があります。

`spam-stat-reset` [Function]

すべての統計を削除して `spam-stat` ハッシュテーブルをリセットします。

行なったら辞書をセーブしなければなりません。辞書はかなり大きくなるかもしれません。辞書を追加更新しない場合（言い換えると、例えば毎月一回作り直す場合）、頻繁に現れないか、または `spam` か `spam` ではないメールのどちらに属するかがはっきりしないすべての単語を削除することによって、辞書のサイズを小さくすることができます。

`spam-stat-reduce-size` [Function]

辞書のサイズを小さくします。これは辞書を追加更新したくない場合だけ使って下さい。

`spam-stat-save` [Function]

辞書をセーブします。

`spam-stat-file` [Variable]

辞書の格納に使うファイル名です。デフォルトは `~/ .spam-stat.el` です。

8.20.8.2 spam-統計 (`spam-stat`) を使ってメールを分割する

この章は Spam 統計パッケージを Spam パッケージ (see Section 8.20 [Spam Package], page 274) とは 独立して 使う方法について説明します。

最初に以下を `~/ .gnus.el` ファイルに追加して下さい:

```
(require 'spam-stat)
(spam-stat-load)
```

これは必要な Gnus のコードとあなたが作った辞書を読み込みます。

次に、特級分割の規則を改造する必要があります: どうやって `spam-stat` を使うかを決めて下さい。以下の例は `nnml` バックエンド用です。 `nnimap` バックエンドでもまったく同様に動作します。単に `nnmail-split-fancy` の代わりに `nnimap-split-fancy` を使って下さい。

`'mail.misc'` と `'mail.spam'` の二つのグループだけがある、最も単純な事例を想定しましょう。以下の式は、メールが `spam` でなかったら `'mail.misc'` に行くことを示します。もし `spam` だったら、`spam-stat-split-fancy` は `'mail.spam'` を返します。

```
(setq nnmail-split-fancy
      '(| (: spam-stat-split-fancy)
          "mail.misc"))
```

`spam-stat-split-fancy-spam-group` [Variable]

Spam 用のグループです。デフォルトは `'mail.spam'` です。

特定の表題を持つメールを他のグループに入れる濾過をも行ないたいならば、以下の式を使って下さい。正規表現に合致しないメールだけに `spam` の可能性があると考えます。


```
(setq nnmail-split-fancy
  '(| ("Subject" "\\bspam-stat\\b" "mail.emacs")
      (: spam-stat-split-fancy)
      "mail.misc"))
```

最初に spam の濾過をしたい場合、辞書を作るときに十分に注意しなければなりません。spam-stat-split-fancy は 'mail.emacs' と 'mail.misc' のどちらのメールも spam ではないと解釈しなければならないので、辞書を作るときに spam ではない集合に両方とも入っていなければならないことに注意して下さい。

```
(setq nnmail-split-fancy
  '(| (: spam-stat-split-fancy)
      ("Subject" "\\bspam-stat\\b" "mail.emacs")
      "mail.misc"))
```

これを伝統的な濾過と組み合わせることもできます。ここではすべての HTML だけのメールを 'mail.spam.filtered' グループに入れるものとしましょう。spam-stat-split-fancy はそれらのメールを見ないので、辞書を作るときに、'mail.spam.filtered' のメールが spam の集合または spam ではない集合のどちらにも入るべきではないことに注意して下さい!

```
(setq nnmail-split-fancy
  '(| ("Content-Type" "text/html" "mail.spam.filtered")
      (: spam-stat-split-fancy)
      ("Subject" "\\bspam-stat\\b" "mail.emacs")
      "mail.misc"))
```

8.20.8.3 spam-統計 (spam-stat) 辞書への低階層インターフェース

spam-stat を使うための主なインターフェースは以下の関数群です:

spam-stat-buffer-is-spam [Function]
Spam であると考えられる新着メールがあるバッファで呼ばれます。まだ処理されていない新着メールに対して使して下さい。

spam-stat-buffer-is-no-spam [Function]
Spam ではないと考えられる新着メールがあるバッファで呼ばれます。まだ処理されていない新着メールに対して使して下さい。

spam-stat-buffer-change-to-spam [Function]
それが spam ではなくて通常のものだとはもはや考えられないメールがあるバッファで呼ばれます。すでに spam ではないものとして処理されてしまったメールの地位の変更に使して下さい。

spam-stat-buffer-change-to-non-spam [Function]
それが通常のものでなくて spam だとはもはや考えられないメールがあるバッファで呼ばれます。すでに spam であるものとして処理されてしまったメールの地位の変更に使して下さい。

spam-stat-save [Function]
ハッシュテーブルをファイルにセーブします。変数 spam-stat-file で設定されたファイル名が使われます。

`spam-stat-load` [Function]
ハッシュテーブルをファイルから読み込みます。変数 `spam-stat-file` で設定されたファイル名が使われます。

`spam-stat-score-word` [Function]
単語の spam スコアを返します。

`spam-stat-score-buffer` [Function]
バッファの spam スコアを返します。

`spam-stat-split-fancy` [Function]
特級メール分割のためにこの関数を使って下さい。nnmail-split-fancy に規則 ‘(:spam-stat-split-fancy)’ を追加しましょう。

それを使う前に、必ず辞書が読み込まれているようにして下さい。これには ‘`~/gnus.el`’ ファイルに以下が必要です:

```
(require 'spam-stat)
(spam-stat-load)
```

以下は一般的なテストのための関数呼び出しです:

```
リセット: (setq spam-stat (make-hash-table :test 'equal))
Spam の学習: (spam-stat-process-spam-directory "~/Mail/mail/spam")
非-spam の学習: (spam-stat-process-non-spam-directory "~/Mail/mail/misc")
辞書をセーブ: (spam-stat-save)
辞書ファイルのサイズを確認: (nth 7 (file-attributes spam-stat-file))
単語数を確認: (hash-table-count spam-stat)
Spam の検査: (spam-stat-test-directory "~/Mail/mail/spam")
非-spam の検査: (spam-stat-test-directory "~/Mail/mail/misc")
辞書のサイズを小さくする: (spam-stat-reduce-size)
辞書をセーブ: (spam-stat-save)
辞書ファイルのサイズを確認: (nth 7 (file-attributes spam-stat-file))
単語数を確認: (hash-table-count spam-stat)
Spam の検査: (spam-stat-test-directory "~/Mail/mail/spam")
非-spam の検査: (spam-stat-test-directory "~/Mail/mail/misc")
```

以下は辞書を生成する方法です:

```
リセット: (setq spam-stat (make-hash-table :test 'equal))
Spam の学習: (spam-stat-process-spam-directory "~/Mail/mail/spam")
非-spam の学習: (spam-stat-process-non-spam-directory "~/Mail/mail/misc")
別の必要な非-spam グループに対して繰り返し...
辞書のサイズを小さくする: (spam-stat-reduce-size)
辞書をセーブ: (spam-stat-save)
```

8.21 他のモードとの相互作用

8.21.1 Dired

`gnus-dired-minor-mode` は dired バッファで使えるいくつかの便利な機能を提供します。これは次の式で有効になります:

```
(add-hook 'dired-mode-hook 'turn-on-gnus-dired-mode)
```

C-c C-m C-a

Dired で印を付けたものを添付ファイルとして送信します (gnus-dired-attach)。どの message バッファに添付するかを尋ねられます。

C-c C-m C-l

適切な mailcap 項目に従ってファイルを開きます (gnus-dired-find-file-mailcap)。接頭引数を付けると、ファイルを新しいバッファで (単に) 開きます。

C-c C-m C-p

mailcap 項目に従ってファイルを印刷します (gnus-dired-print)。印刷コマンドが無い場合は PostScript 画像として印刷します。

8.22 いろいろのいろいろ

gnus-home-directory

すべての Gnus のファイル名とディレクトリー名の変数は、これを基点にして初期値が決定されます。デフォルトは ‘~/’ です。

gnus-directory

Gnus がデータを格納するほとんどのファイルとディレクトリーの名前の変数は、これを基点にして初期値が決定されます。デフォルトは SAVEDIR 環境変数の値か、その変数が設定されていない場合は ‘~/News/’ です。

‘~/gnus.el’ ファイルが読まれたときは Gnus のほとんどはすでに読み込まれているということに注意して下さい。これはつまり、この変数を ‘~/gnus.el’ の中で設定しても、この変数によって初期化される他のディレクトリー変数は正しく設定されないだろうということです。この変数は代わりに ‘.emacs’ で設定して下さい。

gnus-default-directory

上記の変数にはまったく関係ありません—この変数はすべての Gnus バッファのデフォルトディレクトリーをどうすべきかを設定します。もし *C-x C-f* のような命令を実行すると、現在のバッファのデフォルトディレクトリーを起点にしたプロンプトが出てくるでしょう。この変数が nil (これがデフォルト) であれば、Gnus を起動したときにあなたがいたバッファのデフォルトディレクトリーがデフォルトディレクトリーになるでしょう。

gnus-verbose

この変数は 0 から 10 までの間の整数です。数値が大きいほどたくさんのメッセージが表示されます。この変数が 0 であれば Gnus は何のメッセージも見せません。これが 7 (デフォルト) であれば特に重要なメッセージが表示され、10 であれば Gnus は決してお喋りを止めず、たくさんのメッセージであなたにめまいを起こさせるでしょう。

gnus-verbose-backends

この変数は gnus-verbose と同様の効果をもたらしますが、Gnus 本体ではなく Gnus のバックエンドに対して適用されます。

nnheader-max-head-length

バックエンドが記事の連続したヘッダー部を読んでいるときは、できる限り少ない量だけを読もうとします。この変数 (デフォルト 8192) は、バックエンドがヘッダーと本文

の間の区切り行の検索を諦める前に読み込む絶対最大長を指定します。この変数が `nil` であれば、読み込み上限はありません。もし `t` であれば、バックエンドは記事を部分部分で読み込もうとはせず、記事全体を読み込みます。これは `ange-ftp` や `efs` のあるバージョンで意味を持ちます。

`nnheader-head-chop-length`

この変数 (デフォルト 2048) は、前記の操作を行なっているときに、どれくらいの大きさの単位で各記事を読み込むかを設定します。

`nnheader-file-name-translation-alist`

これはファイル名の文字をどのように変換するかを指定する連想リストです。例えば、もし `‘:’` があなたのシステムではファイル名の文字としては使えない場合 (あなたは OS/2 利用者ですね)、以下のようにすることができます。

```
(setq nnheader-file-name-translation-alist
      '((?: . ?_)))
```

実際には、これは OS/2 と MS Windows (ちえっ!) システム上でのこの変数のデフォルト値です。

`gnus-hidden-properties`

これは「不可視」テキストを隠すために使われる属性のリストです。ほとんどのシステムではデフォルトは (`invisible t intangible t`) で、これは不可視テキストを見えなくして触れないようにします。

`gnus-parse-headers-hook`

ヘッダーを解釈する前に呼び出されるフック。これは例えば、取得したヘッダーの統計情報を取るとか、あるいはある種のヘッダーを取り除くことに使うことができます。まあ、私は何でこんなものが欲しいかよくわかんないんだけどね。

`gnus-shell-command-separator`

二つのシェル命令を区切るのに使用される文字列。デフォルトは `‘;’` です。

`gnus-invalid-group-regexp`

利用者にグループ名を尋ねるときに使う、「無効な」グループ名に合致する正規表現です。デフォルト値は Gnus の内部動作をめちゃめちゃにしようかもしれない、いくつかの本当に使えないグループ名を引っかけます。(通常、選択方法とグループの境界に使っている `‘:’` などを許してしまうとまずい、ということです。)

IMAP の利用者はグループ名に `‘/’` を使いたいかもしれませんが。

9 終わり

はい、以上がマニュアルです—あなたはもう自分自身の人生を送ることができます。連絡をとって下さい。あなたの猫によろしく伝えて下さい。

おお、神よ—さよならを耐えることはできません。(すすり泣き。)

チャールズ・レズニコフはそれを非常によく表しているので、ここは彼のために譲ります:

賛美の歌 (テデウム)

勝利ゆえにぼくは
歌うのではない、
勝利などひとつもないから、
ありふれた日光のため、
そよ風のため、
春の気前よさのために歌う。

勝利のためではなく
僕としては精一杯やった
一日の仕事のために。
玉座のためではなく
みんなのテーブルの席で。

(新潮文庫「空腹の技法」著:ポールオースター、訳:柴田元幸、畔柳和代、ISBN:4102451080 より引用)

10 付録

10.1 XEmacs

XEmacs はパッケージの集合として配布されています。Gnus の XEmacs パッケージが必要とするものは何であれ、あなたはインストールすべきです。今のところ必要なのは ‘gnus’, ‘mail-lib’, ‘xemacs-base’, ‘eterm’, ‘sh-script’, ‘net-utils’, ‘os-utils’, ‘dired’, ‘mh-e’, ‘sieve’, ‘ps-print’, ‘W3’, ‘pgg’, ‘mailcrypt’, ‘ecrypto’ および ‘sasl’ です。

10.2 歴史

GNUS は梅田政信氏によって書かれました。1994 年の秋が忍び寄ってくるころ、退屈していた Lars・マッグヌ・イングブリグットソン (Lars Magne Ingebrigtsen) は Gnus を書き直そうと決心しました。

この非道な行為の責任者を調べてみたいのなら、あなたの (いまいましい!) ウェブブラウザを <http://quimby.gnus.org/> に向けることができます。これは新しくて粋な版の Gnus の第一配布場所で、Newsrsrc をぶっ壊して人々を激怒させるサイトとしても知られています。

最初のアルファ版の開発期間に、新しい Gnus は“ (ding) Gnus ”と呼ばれていました。(ding) はもちろん、*ding is not Gnus* の短縮形で、これはまったく完全な嘘ですが、だれがそんなことを気にするでしょうか? (ところで、この短縮形の“ Gnus ”はおそらく梅田さんの意図通り「ニュース」と発音されるべきで、そうするともっと適切な名前になります。そう思いませんか?)

どちらにせよ、すべてのエネルギーを新しい元気の良い名前を付けるのに使い果たした後で、その名前はあまりに元気が良すぎるということになり、それを“ Gnus ”と再び命名しました。でも、今回は大文字と小文字を混ぜています。“ Gnus ”と“ GNUS ”です。新しいものと古いもの。

10.2.1 Gnus Versions

最初の「正しい」Gnus 5 のリリースは 1995 年 11 月に Emacs 19.30 の配布に含まれたときになされました (132 の (ding) Gnus のリリース 足すことの Gnus 5.0 の 15 リリース)。

1996 年 3 月に次の世代の Gnus (別名“ September Gnus ”(99 リリースの後で)) が“ Gnus 5.2 ”という名前でリリースされました (40 リリース)。

1996 年の 7 月 28 日に Red Gnus の作業が始まり、それは 1997 年 1 月 25 日に (84 リリースの後で)“ Gnus 5.4 ”としてリリースされました (67 リリース)。

1997 年 9 月 13 日に、Quassia Gnus が開始され、37 リリース続きました。それは“ Gnus 5.6 ”として 1998 年 3 月 8 日にリリースされました (46 リリース)。

1998 年 8 月 29 日に Gnus 5.6 から Pterodactyl Gnus が生まれ、1999 年 12 月 3 日に (99 リリースと CVS リポジトリでの作業の後)“ Gnus 5.8 ”としてリリースされました。

2000 年 10 月 26 日に Oort Gnus が開始され、2003 年 5 月 1 日に Gnus 5.10 としてリリースされました (24 リリース)。

2004 年 1 月 4 日に No Gnus が始まりました。

接頭語を持った版の Gnus—“ (ding) Gnus ”, “ September Gnus ”, “ Red Gnus ”, “ Quassia Gnus ”, “ Pterodactyl Gnus ”, “ Oort Gnus ”, “ No Gnus ”—に出会っても、混乱しないで下さい。あなたが恐がっていることを知られてはいけません。後ろに下がりにさい。ゆっくりと。他に何をしても、走ってはいけません。それが届かなくなるまで、静かに歩き去りにさい。正しくリリースされた版の Gnus を見つけて、代わりにそれにすり寄りなさい。

10.2.2 他の Gnus のバージョン

Lars さんが調製してリリースした Gnus に加えて、日本では Semi-gnus の開発が行なわれています。これは SEMI という MIME の機能を実現するためのライブラリーを使うことを前提としています。

これらの gnus は、主に Gnus 5.6 と Pterodactyl Gnus を元にしています。それらは“ Semi-gnus ”と総称され、T-gnus, Nana-gnus および Chaos の異なった系統があります。これらは強力な MIME の機能と各国語対応の機能を提供するもので、特に日本人の利用者にとって大事なものです。

10.2.3 なぜ?

Gnus の目的は何ですか?

私は、あなたの考え付くことをすべてできる「いかす」「盛り上がる」「かっこいい」「しゃれた」ニュースリーダーを提供したいと思います。これは私の大元の動機だったのですが、Gnus の作業をしている間に、この世代のニュースリーダーは本当に石器時代に属していることが明らかになりました。ニュースリーダーは、インターネットの揺籃期からほとんど発展していませんでした。もし現在の増加率で流通量が増加しつづければ、すべての現在のニュースリーダーはまったく役に立たなくなるでしょう。毎日何千もの新しい記事がやってくるニュースグループを扱うにはどうすれば良いのでしょうか? 数百万の投稿者に遅れないように付いていくにはどうすれば良いのでしょうか?

Gnus はこれらの質問に真の解決を提供するわけではありませんが、私は Gnus がニュースを読み、取得するための新しい方法を実験する場として使われることを、是非とも見届けたいのです。ニュースリーダーをバックエンドから分離するという梅田さんの賢明な方針を拡張することによって、今や Gnus はメールを取得したり、異なる出所からニュースを取得するための新しいバックエンドを書きたい人のために、シンプルなインターフェースを提供しています。私は役に立ちそうなすべての場所に、カスタマイズのためのフックを加えました。それによって、探検し、発明したいすべての人を招いているのです。

おそらく Gnus は完成することはないのかもしれませんが。C-u 100 M-x all-hail-emacs と C-u 100 M-x all-hail-xemacs です。

10.2.4 互換性

Gnus は GNUS と完全に互換性があるように設計されています。ほとんどすべてのキーの割り当てはそのまま残っています。もちろん、多くのキーの割り当てが追加されましたが、一つか二つの目に付かないものを除いて、古い割り当てが変更されたことはありません。

私たちのモットーは:

鋼鉄の骨組みの空高く

です。(訳注: チャールズ・レズニコフの詩“ The Bridge ”の引用。この詩の本体はたった一行“ In a cloud bones of steel. ”だけ。直訳すれば「雲の中に鋼の骨たち」。)

すべての命令は名前が変わっていません。いくつかの内部関数は名前を変えました。

gnus-uu パッケージは劇的に変化しています。See Section 3.16 [Decoding Articles], page 77.

主要な互換性の問題の一つは、複数の概略バッファが存在することです。グループを読んでいるときに関連するすべての変数は、それが属する概略バッファでバッファローカルです。概略バッファでコマンドが実行されるときはいつでも、多くの重要な変数がそのグローバルな値にも複製されますが、あなたが注意していないと、その変更は正しくない値が使われることをもたらすかもしれません。

GNUS の内部の知識に依存したすべてのコードは実行できないでしょう。二つ例を挙げます: `gnus-news-alist` の並べ替え (もしくは、実際は何らかの方法でそれを変更すること) は厳しく禁止されています。Gnus はこの連想リストの項目を指し示すハッシュテーブルを維持しており (それは多くの関数の速度を上げます)、直接連想リストを変更することは異常な結果をもたらすでしょう。

古い `hilit19` のコードはまったく動作しません。実際のところ、おそらくすべての `hilit` コードをすべての Gnus のフック (`gnus-group-prepare-hook` および `gnus-summary-prepare-hook`) から取り除くべきです。Gnus はハイライトのためのいろいろな統合された関数を提供します。これらはもっと速くもっと正確です。すべての人の人生を楽にするために、Gnus はデフォルトですべての `hilit` フックからすべての `hilit` 呼び出しを取り除きます。きたないもの! 失せろ!

`expire-kill` のようなパッケージはもう動作しません。実際 Gnus を使い始めたときには、おそらくすべての古い GNUS パッケージ (と他のコード) を消去するべきでしょう。GNUS に実行させるために書いたコードは、Gnus がすでに実行しているということは良くあることです。(くすくす。)

ものごとを実行する古いやり方はまだ使うことができますが、新しいやり方だけがこのマニュアルに記載されています。もしこのマニュアルを読んでいる間に何かをする新しいやり方を発見しても、古いやり方を止めなければならないということではありません。

Gnus はすべての GNUS の起動ファイルを理解します。

全体として、GNUS の内部に依存したコードをほとんど書いていない普通の利用者は、問題に苦しむことはないでしょう。もし問題が起こったら、魔法の `M-x gnus-bug` 命令を実行することによって私に知らせて下さい。

非常に よくバグ報告を送る習慣があるのなら、あなたの役に立とうとする `help` バッファが、しばらくするとうるさく感じるかもしれません。そうならば、それが表示されないようにするために、`gnus-bug-create-help-buffer` を `nil` に設定して下さい。

10.2.5 標準への準拠

理由無き反抗などと申すものはございませんよ、奥様。私たちはすべての知られている標準に準拠しています。もちろん私たちが賛成できない標準と/もしくは習慣は除きますが。

RFC (2)822

この標準への知られている違反はありません。

RFC 1036 この標準も知られている違反はありません。

Son-of-RFC 1036

これにはいくつかの違反があります。

X-Newsreader

User-Agent

これらは「つまらないヘッダー」と見なされていますが、私は消費者の情報であると考えています。tin と Netscape から送られてくる非常に多くの酷い記事を見た後では、私は記事を投稿するためにはそれらを使わない方がよいということを知っています。もし X-Newsreader ヘッダーが無ければ、私はその情報を得ることはなかったでしょう。

USEFOR USEFOR は、Son-of-RFC 1036 に基づいて IETF の作業部会が RFC 1036 の後継として書いているものです。ニュース記事の様式に対して、いろいろな変更を提案した草稿を作成しました。その草稿が RFC として受け入れられたときに、Gnus タワーはその変更の実装を調べることになるでしょう。

MIME - RFC 2045-2049 etc

MIME 関連のすべての RFC がサポートされています。

Disposition Notifications - RFC 2298

Message Mode は受信者からの開封確認を要求することができます。

PGP - RFC 1991 and RFC 2440

RFC 1991 は最初の PGP メッセージの規格で、Informational RFC (訳注: 後述の標準化トラックではないが有用な情報) として発行されました。現在 Open PGP と呼ばれる後継の RFC 2440 が、標準化トラック (訳注: Standards Track—国際標準とすべき仕様) に乗せられました。どちらも非-MIME メッセージのための PGP の様式を定義します。Gnus はエンコード (署名および暗号化) とデコード (認証および暗号のデコード) の両方をサポートします。

PGP/MIME - RFC 2015/3156

RFC 2015 (RFC 1991 の代わりに RFC 2440 に基づいた 3156 で置き換えられました) は、RFC 1991/2440 を MIME で囲う様式について述べています。Gnus はエンコードとデコードの両方をサポートします。

S/MIME - RFC 2633

RFC 2633 は S/MIME の形式について述べています。

IMAP - RFC 1730/2060, RFC 2195, RFC 2086, RFC 2359, RFC 2595, RFC 1731

RFC 1730 は IMAP バージョン 4 で、RFC 2060 (IMAP 4 改定 1) で多少更新されています。RFC 2195 は IMAP の CRAM-MD5 認証について述べています。RFC 2086 は IMAP の使用制限一覧 (ACL) について述べています。RFC 2359 は IMAP のプロトコルの拡張について述べています。RFC 2595 は IMAP における適切な TLS の統合 (STARTTLS) について述べています。RFC 1731 は IMAP の GSSAPI/Kerberos4 の手法について述べています。

上に書かれている文章に関する事で、Gnus がそれを満たしていないような動作をしていることに気付いたら、ためらわずに Gnus タワーと私たちに知らせて下さい。

10.2.6 Emacsen

Gnus は以下のもので動作します:

- Emacs 21.1 とそれ以上。
- XEmacs 21.4 とそれ以上。

この Gnus の版はこれより古いどんな Emacsen でも完全に動作しないでしょう。少なくとも信頼できる動作はしないでしょう。古い版の Gnus は古い Emacs の版でも動作するでしょう。特に Gnus 5.10.8 は Emacs 20.7 と XEmacs 21.1 でも動くはずです。

いろいろなプラットフォームの Gnus の間にはいくつかの漠然とした違いがあります—XEmacs には画像機能 (ロゴとツールバー) の特徴があります—しかし、その他はすべての Emacsen でほとんど同じはずです。

10.2.7 Gnus の開発

Gnus は二つのサイクルで開発されています。最初のサイクルでは 'ding@gnus.org' でたくさんの議論を行ないます。そこでは人々が変更や新しい機能の提案をしたり、パッチや新しいバックエンドを投稿します。この段階は「アルファ」段階と呼ばれています。というのは、この段階でリリースされた Gnusae は「アルファリリース」もしくは (他の団体ではより良く使われる) 「スナップショッ

ト」と呼ばれるものだからです。この段階では Gnus は不安定で、一般の利用者によって使われるべきではないと考えられています。Gnus のアルファリリースは“ Red Gnus ”や“ Quassia Gnus ”のような名前になっています。

のらくらと 50-100 くらいのアルファリリースをした後で、Gnus は「凍結」されたと宣言され、バグ修正のみが適用されます。Gnus は接頭語を失い、その代わりに“ Gnus 5.6.32 ”のように呼ばれます。これらは普通の人を使うことができるものと考えられ、主に ‘gnu.emacs.gnus’ ニュースグループで議論されています。

アルファ Gnusae とリリースされた Gnusae では、変数のデフォルトが違うものがあります。特に mail-source-delete-incoming は、アルファ Gnusae では nil で、リリースされた Gnusae では t です。これはメールを扱っている際に、アルファリリースがしゃっくりをしてメールを失なうことを避けるためです。

ding メーリングリストと Gnus ニュースグループにおける議論は、純粹に公衆の関心によって分離されているわけではありません。アルファ Gnus リリースが（ときどき）するかもしれない恐ろしいことを公衆の場で書くのは、皆を恐れさせるというのも真実ですが、もっと重要なことは、導入された新しい実験的な機能について話すことが、一般の利用者を混乱させるかもしれないということです。新しい機能は頻繁に導入され、いじくられ、不十分であると判断され、そうすると捨てられるか、完全に書き換えられるかのどちらかです。メーリングリストを読んでいる人は普通はこの速い変更についていいますが、ニュースグループの人もそうであると見なすことはできません。

10.2.8 貢献者

新しい Gnus の版は (ding) メーリングリストのすべての人たちの助けが無ければできなかったでしょう。一年以上にわたって、私は毎日彼らから莫大な数の素敵なバグレポートを受け取り、そのそれぞれが私を喜びで満ちました。投げキッス。このリストの人たちは、私のリリース方針のために耐え難きを耐える試練に遭いました：「ああ、それはすばらしい考えだ <かしゃかしゃかしゃ...> よしっ、すぐにリリースだ <えいやっ> あれれっ、まったく動かないぞ <かしゃかしゃかしゃ...> よしっ、すぐに出そう <ほらよっ> おっと、待った、ぜんぜん動作しない...」。Micro\$oft—あっかんべーだ。アマチュアめ。私は もっと 悪い。(それとも「より悪い」? 「もっと悪い」? 「最悪」?)

私はこの機会に学会に感謝を... おおっと、違った。

- 梅田政信—元の GNUS を書いた人です。
- Shenghuo Zhu—uudecode.el, mm-uu.el, rfc1843.el, webmail.el, nnwarchive, それに一般的なバグ修正、新しい機能などはもとより MIME と他の形式のエンコード/デコードに関連するほんとうに多くのもの。
- Per Abrahamsen—custom、スコア、ハイライトと SOUP コード (他の多くのことと共に)。
- Luis Fernandes—デザインとグラフィック。
- Joe Reiss—スマイリーの顔の作者。
- Justin Sheehy—FAQ のメインテイナー。
- Erik Naggum—手助け、アイデア、支援、コード他。
- Wes Hardaker—‘gnus-picon.el’ とマニュアルの picon の章 (see Section 8.17.4 [Picons], page 268)。
- Kim-Minh Kaplan—picon コードにおける更なる作業。
- Brad Miller—‘gnus-gl.el’ とマニュアルの GroupLens の章。
- Sudish Joseph—数え切れないほどのバグの修正。
- Ilja Weis—‘gnus-topic.el’。

- Steven L. Baur—たくさんのたくさんのたくさんのバグの発見と修正。
- Vladimir Alexiev—refcard とリファレンスの小冊子。
- Felix Lee & Jamie Zawinski—私は Felix Lee と JWZ の XGnus 配布からいくつかの部分
を盗みました。
- Scott Byer—‘nnfolder.el’ の拡張と改訂。
- Peter Mutsaers—孤児記事のスコアコード。
- Ken Raebburn—POP メールサポート。
- Hallvard B Furuseth—いろいろな小さな物や部分、特に .newsrsrc ファイルを扱う部分。
- Brian Edmonds—‘gnus-bbdb.el’。
- David Moore—‘nnvirtual.el’ の改訂と多くの他のこと。
- Kevin Davidson—*ding* の名前を思い付けました。ですから、彼を責めて下さい。
- France, Agois Pinard—多くの、多くの興味深く完全なバグレポートと autoconf のサポート。

このマニュアル (Gnus 英語版) は Adrian Aichner と Ricardo Nassif, Mark Borges によって校正され、Jost Krieger によって一部分を校正されました。

以下の人々は多くのパッチと提案で貢献しました:

Christopher Davis, Andrew Eskilsson, Kai Grossjohann, Kevin Greiner, Jesper Harder, Paul Jarc, Simon Josefsson, David Koe, Aegedal, Richard Pieri, Fabrice Popineau, Daniel Quinlan, Michael Shields, Reiner Steib, Jason L. Tibbitts, III, Jack Vinson, 山岡 克美, and Teodor Zlatanov.

それと、以下の人にもパッチやその他のものを感謝します:

Jari Aalto, Adrian Aichner, Vladimir Alexiev, Russ Allbery, Peter Arius, Matt Armstrong, Marc Auslander, Miles Bader, Frank Bennett, Alexei V. Barantsev, Robert Bihlmeyer, Chris Bone, Mark Borges, Mark Boyns, Rob Browning, Lance A. Brown, Kees de Bruin, Martin Buchholz, Joe Buehler, Kevin Buhr, Alastair Burt, Joao Cachopo, Zlatko Calusic, Massimo Campostrini, Castor, David Charlap, Dan Christensen, Kevin Christian, Jae-you Chung, James H. Cloos, Jr., Laura Conrad, Michael R. Cook, Glenn Coombs, Andrew J. Cosgriff, Neil Crellin, Frank D. Cringle, Geoffrey T. Dairiki, Andre Deparade, Ulrik Dickow, Dave Disser, Rui-Tao Dong, Jov Dubach, Michael Welsh Duggan, Dave Edmondson, Paul Eggert, Mark W. Eichin, Karl Eichwalder, 榎並 嗣智, Michael Ernst, Luc Van Eycken, Sam Falkner, Nelson Jose dos Santos Ferreira, Sigbjorn Finne, Sven Fischer, Paul Fisher, Decklin Foster, Gary D. Foster, Paul Franklin, Guy Geens, Arne Georg Gleditsch, David S. Goldberg, Michelangelo Grigni, Dale Hagglund, D. Hall, Magnus Hammerin, 半田 剣一, Raja R. Harinath, 林 芳樹, P. E. Jareth Hein, ひさしげ けんじ, Scott Hofmann, Marc Horowitz, Gunnar Horrigmo, Richard Hoskins, Brad Howes, Miguel de Icaza, France, Agois Felix Ingrand, 市川 達哉, 石川 一郎, Lee Iverson, 岩室 元典, Rajappa Iyer, Andreas Jaeger, Adam P. Jenkins, Randell Jesup, Fred Johansen, Gareth Jones, Greg Klanderma, Karl Kleinpaste, Michael Klingbeil, Peter Skov Knudsen, 小林 修平, Petr Konecny, 小関 吉則, Thor Kristoffersen, Jens Lautenbacher, Martin Larose, Seokchan Lee, Joerg Lenneis, Carsten Leonhardt, James LewisMoss, Christian Limpach, Markus Linnala, Dave Love, Mike McEwan, Tonny Madsen, Shlomo Mahlab, Nat Makarevitch, Istvan Marko, David Martin, Jason R. Mastaler, Gordon Matzigkeit, Timo Metzmakers, Richard Mlynarik, Lantz Moore, 守岡 知彦, Erik Toubro Nielsen, Hrvoje Niksic, Andy Norman, Fred Oberhauser, C. R. Oldham, Alexandre Oliva, Ken Olstad, 大西 雅晴, 小野 秀貴, Ettore

Perazzoli, William Perry, Stephen Peters, Jens-Ulrik Holger Petersen, Ulrich Pfeifer, Matt Pharr, Andy Piper, John McClary Prevost, Bill Pringlemeir, Mike Pullen, Jim Radford, Colin Rafferty, Lasse Rasinen, Lars Balkar Rasmussen, Joe Reiss, Renaud Rioboo, Roland B. Roberts, Bart Robinson, Christian von Roques, Markus Rost, Jason Rumney, Wolfgang Rupprecht, Jay Sachs, Dewey M. Sasser, Conrad Sauerwald, Loren Schall, Dan Schmidt, Ralph Schleicher, Philippe Schnoebelen, Andreas Schwab, Randal L. Schwartz, Justin Sheehy, Danny Siu, Matt Simmons, Paul D. Smith, Jeff Sparkes, Toby Speight, Michael Sperber, Darren Stalder, Richard Stallman, Greg Stark, Sam Steingold, Paul Stevenson, Jonas Steverud, Paul Stodghill, 須藤 清一, Kurt Swanson, Samuel Tardieu, Teddy, 戸沢 晶彦, Chuck Thompson, Philippe Troin, James Troup, Trung Tran-Duc, Jack Twilley, Aaron M. Ucko, Aki Vehtari, Didier Verna, Vladimir Volovich, Jan Vroonhof, Stefan Waldherr, Pete Ware, Barry A. Warsaw, Christoph Wedler, Joe Wells, Lee Willis, and Lloyd Zusman.

Gnus のアルファ配布に含まれている ChangeLog は、それぞれの人たちが行ったことの完全な大要を伝える豊かな読み物です。(550KB といくらか)。(訳注: 非常に古い ChangeLog の記述が何度かばっさり捨てられましたが、それでも現在は非常に大きなサイズになっています。)

私が忘れてしまったすべての人に謝罪します。間違いなくたくさんの人を忘れてしまったことでしょう。

わぁ、こんなに人がいるとは思わなかった。これは本当に Gnus を使っている人がいるということなのでしょう。そんなことを誰が想像したのでしょうか!

10.2.9 新しい機能

これらのリストは、もちろん たいていの 重要な新しい機能に関する 短い 要約でしかありません。いいえ、実は違います。もっともっとたくさんのものがあるのです。そう、事実上私たちは十分に用の無いもの (feeping creaturism) を持っているのです。

10.2.9.1 (ding) Gnus

Gnus 5.0/5.1 の新しい機能:

- すべてのバッファの外観は、フォーマットのような変数 (see Section 2.1 [Group Buffer Format], page 13 and see Section 3.1 [Summary Buffer Format], page 43) によって設定を変えることができるようになりました。
- ローカルスプールと、いくつかの NNTP サーバーを同時に使うことができるようになりました (see Chapter 6 [Select Methods], page 133)。
- 複数のグループを仮想グループに合併できるようになりました (see Section 6.7.1 [Virtual Groups], page 203)。
- 多くの異なるメール様式を読むようになりました (see Section 6.3 [Getting Mail], page 145)。すべてのメールバックエンドは、便利なメール期限切れ消去機構を実装しています (see Section 6.3.9 [Expiring Mail], page 163)。
- Gnus は根っこ (root) を失ったスレッドを集めるためのいろいろな戦略 (それによってまばらな副スレッドを一つのスレッドにする) を使ったり、完全なスレッドを組み上げるのに十分なヘッダーをいったん戻って取得することができます (see Section 3.9.1 [Customizing Threading], page 63)。
- 切られたグループ (killed groups) はグループバッファに表示することができて、それらも読むことができます (see Section 2.11 [Listing Groups], page 29)。

- Gnus はグループを部分的に更新することができます—2,3 のグループの新しい記事を調べるために、アクティブファイル全体を取得する必要はありません (see Section 1.9 [The Active File], page 10)。
- Gnus はグループの段階的購読度を実装しました (see Section 2.6 [Group Levels], page 19)。
- 何種類もの基準に従って、記事にスコアを付けることができます (see Chapter 7 [Scoring], page 227)。どのように記事にスコアを付けるかを、Gnus に見つけさせることもできます (see Section 7.6 [Adaptive Scoring], page 237)。
- Gnus は普通の Emacs の方法で自動保存されるドリブルバッファを維持するので、あなたが何を讀んだかのデータをマシンが落ちたときでもあまり失わないでしょう (see Section 1.8 [Auto Save], page 9)。
- Gnus は ‘.emacs’ ファイルをぐちゃぐちゃにすることを避けるために、今では専用の起動ファイル (‘~/gnus.el’) を持つようになりました。
- グループと記事の両方にプロセス印を付けることができ、すべての印の付いた項目で処理を実行することができます (see Section 8.1 [Process/Prefix], page 249)。
- グループ群の一部を grep して、その結果から一つのグループを作ることができます (see Section 6.7.2 [Kibozed Groups], page 204)。
- グループの一覧を、えーと、どんな条件でも、表示することができます (see Section 2.11 [Listing Groups], page 29)。
- 外部サーバーを概観して、それらのサーバーのグループを購読することができます (see Section 2.14 [Browse Foreign Server], page 32)。
- Gnus はサーバーとの二つ目の接続で、記事を非同期的に取ってくることができます (see Section 3.11 [Asynchronous Fetching], page 70)。
- 記事をローカルにキャッシュすることができます (see Section 3.12 [Article Caching], page 71)。
- uudecode の関数が拡張され、一般化されました (see Section 3.16 [Decoding Articles], page 77)。
- uuencode された記事をまだ投稿することができます。これは過去に GNUS のあまり知られていない機能でした (see Section 3.16.5.3 [Uuencoding and Posting], page 80)。
- 親記事 (と他の記事) の取得は、今では調子が悪くなることも無く、実際に動作するようになりました (see Section 3.22 [Finding the Parent], page 99)。
- Gnus は FAQ とグループの説明を取得することができます (see Section 2.17.2 [Group Information], page 40)。
- まとめ送りされた記事 (および他のファイル) を、グループとして使えるようになりました (see Section 6.6.3 [Document Groups], page 196)。
- 記事をハイライトし、カスタマイズすることができます (see Section 4.3 [Customizing Articles], page 118)。
- URL と他の外部参照がボタンになるようになりました (see Section 3.17.6 [Article Buttons], page 89)。
- Gnus のウィンドウとフレームの設定でたくさんの変なことをできるようになりました (see Section 8.5 [Window Layout], page 254)。
- キーボードを使う代わりに、ボタンをクリックできるようになりました (see Section 8.10 [Buttons], page 260)。

10.2.9.2 September Gnus

Gnus 5.2/5.3 の新しい機能:

- 新しいメッセージ作成モードが使われます。mail-mode, rnews-reply-mode と gnus-msg のすべての古いカスタマイズ変数は今や旧式になりました。
- Gnus は「まばら」スレッドを作成することができるようになりました—スレッドの失われた記事があるところは、空の節で表現されるようになっていきます (see Section 3.9.1 [Customizing Threading], page 63)。

```
(setq gnus-build-sparse-threads 'some)
```

- 外に出ていく記事は、特別な保管サーバーに保存されるようになりました (see Section 5.5 [Archived Messages], page 126)。
- 記事が参照されたときに、スレッドの部分作成が行なわれるようになりました。
- Gnus は GroupLens の予測を利用することができるようになりました。
- Picons (personal icons) (個人アイコン) が XEmacs で表示できるようになりました (see Section 8.17.4 [Picons], page 268)。
- trn のような木バッファーを表示できるようになりました (see Section 3.24 [Tree Display], page 101)。

```
(setq gnus-use-trees t)
```

- ニュースリーダー nn のような、選んで読むマイナーモードを概略バッファで使うことができるようになりました (see Section 3.23.1 [Pick and Read], page 100)。

```
(add-hook 'gnus-summary-mode-hook 'gnus-pick-mode)
```

- バイナリーグループで特別なバイナリーマイナーモードを使うことができるようになりました (see Section 3.23.2 [Binary Groups], page 101)。
- グループ群を折り畳みトピック階層にグループ分けできるようになりました (see Section 2.16 [Group Topics], page 33)。

```
(add-hook 'gnus-group-mode-hook 'gnus-topic-mode)
```

- メールの再送と、弾かれたメールを送り直すことができるようになりました (see Section 3.5.1 [Summary Mail Commands], page 51)。
- グループがスコアを持つことができるようになり、訪れる回数に基づいた並べ替えが可能になりました (see Section 2.7 [Group Score], page 20)。

```
(add-hook 'gnus-summary-exit-hook 'gnus-summary-bubble-group)
```

- グループにプロセス印を付けられるようになり、グループのグループに対して命令を実行できるようになりました (see Section 2.8 [Marking Groups], page 21)。
- 仮想グループでキャッシュができるようになりました。
- nndoc はすべての種類のまとめ送り、メールボックス、rnews ニュースの一括配送、ClariNet の要約集、そしてその他のすべてを理解できるようになりました (see Section 6.6.3 [Document Groups], page 196)。
- Gnus は SOUP パケットを作成/読み込みをするための新しいバックエンド (nnsoup) を持っています。
- キャッシュがずっと速くなりました。
- グループを多くの基準に従って並べ替えることができるようになりました (see Section 2.12 [Sorting Groups], page 30)。

- メーリングリストのアドレスと期限切れ消去の時間を設定する、新しいグループパラメーターが導入されました (see Section 2.10 [Group Parameters], page 23)。
- すべてのフォーマット指定で、フェースを指定できるようになりました (see Section 8.4.5 [Formatting Fonts], page 252)。
- *M P* 副キーマップに、プロセス印の付いた記事の設定/削除/実行のための複数の命令が追加されました (see Section 3.7.6 [Setting Process Marks], page 60)。
- 広範囲の基準に基づいて、概略バッファが利用可能な記事の一部だけを表示するように制限できるようになりました。これらの命令は / 副マップのキーにバインドされています (see Section 3.8 [Limiting], page 61)。
- * 命令によって、記事を永続させることができるようになりました (see Section 3.13 [Persistent Articles], page 72)。
- 記事の要素を隠すすべての関数は、トグルになりました。
- 記事のヘッダーにボタンを付けることができるようになりました (see Section 3.17.4 [Article Washing], page 85)。
- すべてのメールバックエンドで、Message-ID による記事の取得をサポートするようになりました。
- 重複メールを適切に扱うことができるようになりました (see Section 6.3.11 [Duplicates], page 167)。
- すべての概略モード命令を、記事バッファから直接使用できるようになりました (see Section 4.4 [Article Keymap], page 120)。
- フレームが `gnus-buffer-configuration` の部分になることができるようになりました (see Section 8.5 [Window Layout], page 254)。
- デーモンのプロセスによって、新着メールを検査できるようになりました (see Section 8.11 [Daemons], page 261)。
- Gnus は spam を根絶やしにするために、NoCeM ファイルを利用できるようになりました (see Section 8.12 [NoCeM], page 262)。
(`setq gnus-use-nocem t`)
- グループを常に見えるようにしておくことが (訳注: そのグループに未読記事が無くても)、できるようになりました (see Section 2.11 [Listing Groups], page 29)。
(`setq gnus-permanently-visible-groups "^nnml:"`)
- カスタマイズを楽にするために、多くの新しいフックが導入されました。
- Gnus は Mail-Copies-To ヘッダーを尊重するようになりました。
- References ヘッダーを調べることによって、スレッドを集めることができるようになりました (see Section 3.9.1 [Customizing Threading], page 63)。
(`setq gnus-summary-thread-gathering-function
 'gnus-gather-threads-by-references`)
- 再取得を避けるために、既読記事を特別なバックログ・バッファに貯めることができるようになりました (see Section 3.14 [Article Backlog], page 73)。
(`setq gnus-keep-backlog 50`)
- 簡単にトリートメントを行なうことができるようにするために、現在の記事の完全な複製がいつも別バッファに置かれるようになりました。

- Gnus がどこに記事を保存するかを提案できるようになりました (see Section 3.15 [Saving Articles], page 73)。
- 記事を保存するときに、多くを入力しなくても良いようになりました (see Section 3.15 [Saving Articles], page 73)。
(setq gnus-prompt-before-saving t)
- gnus-uu は記事を取得している間に、非同期でデコードされたファイルを表示できるようになりました (see Section 3.16.5.2 [Other Decode Variables], page 79)。
(setq gnus-uu-grabbed-file-functions 'gnus-uu-grab-view)
- 記事バッファで、引用されたテキストの折り返しが適切に動作するようになりました (see Section 3.17.4 [Article Washing], page 85)。
- 引用されたテキストを表示するか隠すかを切り替えるためのボタンが追加されました。また、どのくらいの引用文を隠すかをカスタマイズできるようになりました (see Section 3.17.3 [Article Hiding], page 83)。
(setq gnus-cited-lines-visible 2)
- 興味の無いヘッダーを隠すことができます (see Section 3.17.3 [Article Hiding], page 83)。
- スコアのデフォルト値をメニューバーから設定できるようになりました。
- 送信される記事の更なる構文チェックが追加されました。

10.2.9.3 Red Gnus

Gnus 5.4/5.5 の新しい機能:

- 'nntp.el' は非同期に動作するやり方で、完全に改訂されました。
- 記事の先行取得を行なう機能が Gnus に編入されました (see Section 3.11 [Asynchronous Fetching], page 70)。
- スコア付けは and, or, not のような論理演算子と、親記事にさかのぼってリダイレクトすることで実行できるようになりました (see Section 7.15 [Advanced Scoring], page 244)。
- 記事の洗濯状態の記事のモード行に表示できるようになりました (see Section 4.5 [Misc Article], page 121)。
- 'gnus.el' が多くの小さいファイルに分割されました。
- Message-ID に基づいて、記事の重複を抑制することができるようになりました (see Section 3.29 [Duplicate Suppression], page 110)。
(setq gnus-suppress-duplicates t)
- どのスコアと適応ファイルが、ホームスコアと適応ファイルであるかを指定する (see Section 7.7 [Home Score File], page 239) 新しい変数が加えられました。
- nndoc がより簡単に拡張できるように改訂されました (see Section 6.6.3.1 [Document Server Internals], page 197)。
- グループは親のトピックからグループパラメーターを継承できるようになりました (see Section 2.16.5 [Topic Parameters], page 37)。
- 記事を編集するための機能が継ぎはぎされて、実際に使用可能になりました。
- 署名がもっと利口なやり方で認識されるようになりました (see Section 3.17.10 [Article Signature], page 93)。
- 概略ピックモードがもっと (ニュースリーダー) nn らしくなりました。行番号が表示され、記事を選ぶために . 命令を使うことができるようになりました (Pick and Read)。

- あるサーバーから別のサーバーへ ‘.newsrsrc.eld’ を移動する命令が加えられました (see Section 1.6 [Changing Servers], page 8)。
- 今では、バッファの行を作成するときに、抑制される「興味の無い」部分を指定する方法があります (see Section 8.4.3 [Advanced Formatting], page 251)。
- グループバッファの複数の命令について、それらで行なったことを *C-M-_* で元に戻すことができるようになりました (see Section 8.13 [Undo], page 264)。
- 新しいスコア型 *w* を使うことによって、単語でスコア付けをすることが可能になりました (see Section 7.4 [Score File Format], page 232)。
- 表題の一語一語を基にして、適応スコアをすることができるようになりました (see Section 7.6 [Adaptive Scoring], page 237)。

```
(setq gnus-use-adaptive-scoring '(word))
```

- スコアを減衰させることができるようになりました (see Section 7.16 [Score Decays], page 247)。

```
(setq gnus-decay-scores t)
```

- 正規表現を使って、日付のヘッダーでスコア付けを実行できるようになりました。日付は最初にコンパクトな ISO 8601 様式で正規化されます (see Section 7.4 [Score File Format], page 232)。
- 記事に関するすべてのデータを、基本のサーバーから取り除く命令が加えられました (see Section 1.6 [Changing Servers], page 8)。
- 文書を寄せ集めたものを読むための新しい命令 (*nn doc* グループのてっぺんで *nnvirtual* を使います) が加えられました—*C-M-d* (see Section 3.26.4 [Really Various Summary Commands], page 108)。
- プロセス印の設定を *push* と *pop* でスタックに出し入れできるようになりました (see Section 3.7.6 [Setting Process Marks], page 60)。
- NNTP サーバーが投稿を許可していない場合でも、新しい *mail-to-news* バックエンドが、投稿することを可能にしました (see Section 6.6.5 [Mail-To-News Gateways], page 202)。
- ウェブ検索エンジン (*DejaNews*, *Alta Vista*, *InReference*) からの検索結果を読むための、新しいバックエンドが加えられました (see Section 6.4.2 [Web Searches], page 180)。
- 標準の並べ替え関数を使って、トピックの中にあるグループを並び代えることができるようになりました。また、それぞれのトピックを独立して並べ替えることができるようになりました (see Section 2.16.3 [Topic Sorting], page 36)。
- グループ群の一部を、独立して並べ替えることができるようになりました (Sorting)。
- キャッシュされた記事を、グループに引き込むことができるようになりました (see Section 3.26.3 [Summary Generation Commands], page 107)。
- スコアファイルがもっと信頼できる順番で適用されるようになりました (see Section 7.3 [Score Variables], page 230)。
- メールメッセージが分割されてどこに行くかの報告を、作成することができるようになりました (see Section 6.3.3 [Splitting Mail], page 147)。
- 入って来たメールを保存する前にがらくたを取り除くフックと関数が、もっと追加されました (see Section 6.3.10 [Washing Mail], page 165)。
- 強調表示することを指定されたテキストが、適切に表示されるようになりました。

10.2.9.4 Quassia Gnus

Gnus 5.6 の新しい機能:

- Gnus をオフラインニュースリーダーとして使う新機能が加えられました。過剰なほどの新しい命令とモードが追加されました。全貌については Section 6.9 [Gnus Unplugged], page 210 を見て下さい。
- nndraft バックエンドが戻ってきました。でも、依然とは違う動作をします。すべてのメッセージバッファは、今では自動的に作成される nndraft グループの記事でもあります。
- gnus-alter-header-function を、ヘッダーの値を変えるために使うことができました。
- gnus-summary-goto-article が Message-ID を受け付けるようになりました。
- メッセージの本文において、指定したリージョン以外のテキストを消去するための新しいメッセージ命令があります: `C-c C-v`。
- `C-u C-c C-c` によって nnvirtual グループを構成しているグループに投稿できるようになりました。
- nntp-rlogin-program—カスタマイズを簡単にするための新しい変数です。
- gnus-article-edit-mode における `C-u C-c C-c` 命令は、記事バッファの再ハイライトを抑制するようになりました。
- gnus-boring-article-headers に、long-to という新しい要素があります。
- `M-i` シンボル接頭引数命令があります。詳細は Section 8.3 [Symbolic Prefixes], page 250 を見て下さい。
- 概略バッファにおける `L` と `I` は、`'all.SCORE'` ファイルにスコア規則を加えるためのシンボル接頭引数 `a` を受け付けるようになりました。
- 変数 gnus-simplify-subject-functions によって、表題の単純化を強力に制御できるようになりました。
- `A T`—現在のスレッドを取得するための新しい命令です。
- `/ T`—現在のスレッドを制限に含めるための新しい命令です。
- `M-RET` は、引用文の途中に割って入るための新しいメッセージ命令です。
- `'\1'` のような表現が nnmail-split-methods で有効になりました。
- 関数 custom-face-lookup が取り除かれました。あなたの初期化ファイルでこの関数を使っていたのなら、代わりに face-spec-set を使うように書き直さなければなりません。
- 投稿のキャンセルに、現在の選択方法を使うようになりました。シンボル接頭引数 `a` で、普通の投稿方法を強制することができます。
- `マ ソ sm*rtq**t*s` を適切なテキストに翻案する新しい命令があります—`W d`。
- nntp のデバッグを楽にするために、nntp-record-commands を nil ではない値に設定することができます。
- nntp は `'~/ .authinfo'` を使うようになりました。これは `' .netrc'` のようなファイルで、どこの NNTP サーバーにはどのように AUTHINFO を送るかを制御するためのものです。
- 概略バッファのグループパラメーターを編集するための命令が加えられました。
- メールがどこに分割されたかの履歴を利用できるようになりました。
- 記事の日付を表示するための新しい命令が加えられました—`article-date-iso8601`。
- gnus-score-thread-simplify を設定することによって、スレッドを作成するときの表題を単純化できるようになりました。

- メッセージで引用をするための新しい関数が加えられました— `message-cite-original-without-signature`。
- `article-strip-all-blank-lines`—新しい記事命令です。
- 記事の終わりまでを切り取る (kill する) 新しいメッセージ命令が加えられました。
- 変数 `gnus-adaptive-word-minimum` を使うことによって、最小限度の適応スコアを指定することができます。
- `gnus-start-date-timer` 命令によって「記事が投稿されたときからの経過時間」ヘッダーが継続的に更新されるようになりました。
- ウェブで提供されているメーリングリストのアーカイブを、`nnlistserv` バックエンドによって読むことができるようになりました。
- 古い `dejaneWS` アーカイブを `nnweb` で読むことができるようになりました。

10.2.9.5 Pterodactyl Gnus

Gnus 5.8 の新しい機能:

- メールを取り込む機能が変わりました。たくさんの詳細についてはマニュアルを見て下さい。特に `procmail` で取り込むためのすべての変数が無くなっています。

以下のような `procmail` の使い方は

```
(setq nnmail-use-procmail t)
(setq nnmail-spool-file 'procmail)
(setq nnmail-procmail-directory "~/mail/incoming/")
(setq nnmail-procmail-suffix "\\..in")
```

現在では次のように変わっています。

```
(setq mail-sources
      '((directory :path "~/mail/incoming/"
                  :suffix ".in")))
```

See Section 6.3.4.1 [Mail Source Specifiers], page 148.

- Gnus は MIME に対応したリーダーになりました。これは Gnus の多くの部分に影響していて、たくさんの新しいコマンドが追加されています。詳細はマニュアルを参照して下さい。
- しかも Gnus は各国語対応になりました。ここでは要約できないくらいに Gnus の多くの部分に影響していて、新しいたくさんの変数が追加されています。
- `gnus-auto-select-first` が、ポイントを置く場所を決定するための関数であってもよくなりました。
- 概略バッファと NOV ファイルに含める追加のヘッダーを、利用者が決めることができるようになりました。
- `gnus-article-display-hook` が削除されました。代わりに `gnus-treat-` で始まるたくさんの変数が追加されました。
- Gnus posting styles が再び作り直されました。現在は微妙に違うやり方で動作します。
- 新しいウェブに基づいたバックエンドが追加されました。`nnslashdot`, `nnwarchive` および `nnultimate` です。`nnweb` は常に変化する構成をとり続けるために、再び作り直されました。
- Gnus は `nnimap` によって IMAP のメールを読むことができます。

10.2.9.6 Oort Gnus

Gnus 5.10 の新しい機能:

- インストールに関する変更

- Oort を使ったことがある場合の、以前の (安定な) 版からのグレードアップ。

Oort (このリリースに先立つ安定ではない Gnus の枝) を使ってみたものの、安定版に戻ってしまったならば、この版にグレードアップするときに注意して下さい。特に、おそらくすべての `‘.marks’` (nnml) と `‘.mrk’` (nnfolder) ファイルを消去する必要があるでしょう。その目的は、この版 (の Gnus) がフラグを格納する `‘.marks’/‘.mrk’` ファイルではなくて `‘.newsrc.eld’` からフラグが読まれるようにするためです (訳注: 言い換えると、古い様式の `‘.marks’/‘.mrk’` ファイルを新しい Gnus が読んではいけないということです。それらは新たに作成されます)。後述の項目で、印 (marks) に関するより多くの情報を読んで下さい。グレードを下げて一般には助けにならないことに注意して下さい。

- Lisp ファイルがデフォルトで `‘.../site-lisp/gnus/’` にインストールされるようになりました。以前は `‘.../site-lisp/’` がデフォルトでした。加えて、新しいインストーラーは他にインストールされている、新しい Gnus より優先される Gnus を検出して警告を発します。それらを手動で取り除いても良いし、`make remove-installed-shadows` を使って削除することもできます。

- MS ウィンドウズで Gnus をコンパイルしてインストールするための、新しい `‘make.bat’`。

MS ウィンドウズで Gnus をインストールするには `‘make.bat’` を使って下さい。このバッチ・プログラムの第一引数はディレクトリーです。そこで `‘xemacs.exe’` と `‘emacs.exe’` が順に捜し出されます。コンパイルしてから Gnus をインストールしたいときは、`‘make.bat’` の第二引数に `/copy` を与えて下さい。

`‘make.bat’` はゼロから書き直されました。XEmacs と GNU Emacs を自動認識し、`‘gnus-load.el’` を生成し、コンパイル中と info ファイルの生成中にエラーが起きたら構築処理の最後に報告します。makeinfo が利用可能であればそれを使い、さもなければ `‘infohack.el’` に頼ります。今や `‘make.bat’` は Gnus を動作させるために必要なすべてのファイルをインストールするはずで、大体において Unix システムにおける `configure`; `make`; `make install` サイクルの完全な置き換えになりました。

新しい `‘make.bat’` によって `‘make-x.bat’` と `‘xemacs.mak’` が不要になったので、それらは削除されました。

- `‘~/News/overview/’` は不要。

以下の変更の結果、もはや `‘~/News/overview/’` ディレクトリーは要りません。すべての階層を安全に削除することができます。

- (require 'gnus-load)

単独で配布されている Gnus を使う場合には、load-path に Gnus の lisp ディレクトリーを追加してから、`‘~/emacsen/’` に (require 'gnus-load) を加えるのが良いです。

`‘gnus-load.el’` ファイルは、そのうちのいくつかは Emacsen の配布に入っていないかもしれない自動読み込み (autoload) コマンド、関数および変数を含んでいます。

- Gnus に内蔵された新しいパッケージとライブラリー

- 改定された Gnus FAQ がマニュアルに含まれています。See Section 10.10 [Frequently Asked Questions], page 357.

- TLS ラッパーが Gnus に同梱されました。

TLS/SSL が、‘`tls.el`’ と GNUTLS を介して IMAP と NNTP でサポートされるようになりました。(サードパーティーの) ‘`ssl.el`’ と OpenSSL による古い TLS/SSL は、まだ働きます。

- 改良された spam 対抗機能。

Gnus は非常に変化に富んだプログラムと濾過の規則を使って、メールやニュースの奔流から spam を抜き取ってしまうことができるようになりました。対応している方式は、RBL blocklists、bogofilter それにホワイト/ブラックリストです。また SpamAssassin や Hashcash のような外部パッケージを容易に使うための hook も新しくなりました。Section 8.19 [Thwarting Email Spam], page 269 および Section 8.20 [Spam Package], page 274.

- Gnus は Sieve を使ったサーバー側でのメールの濾過をサポートします。

Sieve の規則はグループパラメーターとして加えることができ、グループバッファで `Dg` を使うと完全な Sieve スクリプトが生成されます。そうしたら、生成された Sieve バッファで `C-c C-l` を使って、サーバーにアップロードして下さい。Section 2.17.5 [Sieve Commands], page 42、それに新しい Sieve のマニュアル (see section “Top” in *Emacs Sieve*) を参照して下さい。

- グループモードの変更

- `gnus-group-read-ephemeral-group` を `GM` キーで対話的に呼ぶことができます。
- 憲章とコントロールメッセージの取得。

二つの新しいコマンドで、ニュースグループの憲章を取り込む (`H c`) ことと、コントロールメッセージを取得する (`H C`) ことができます。

- 新しい変数 `gnus-parameters` を、グループパラメーターを設定するために使うことができます。

これは初期には、パラメーターを ‘`~/newsrsrc.eld`’ に格納する `Gp` (または `Gc`) でしか行なうことができませんでしたが、この変数によってカスタマイズの威力を堪能することができます。また、その変数は ‘`~/newsrsrc.eld`’ ではなくて ‘`~/gnus.el`’ で設定するので、バックアップが簡単になります。その変数は、グループ名に合致する正規表現を、以下のような流儀でグループパラメーターに割り当てます:

```
(setq gnus-parameters
      '(("mail\\.*"
         (gnus-show-threads nil)
         (gnus-use-scoring nil))
        ("~nnimap:\\(foo.bar\\)"
         (to-group . "\\1"))))
```

- `nnimap` のグループにおける未読の数が正確になりました。

グループバッファで表示される `nnimap` グループの未読記事の数の見積りが正確になったはずで。これは `gnus-setup-news-hook` (起動時に呼ばれる) と `gnus-after-getting-new-news-hook` (新しいメールを取得した直後に呼ばれる) から `nnimap-fixup-unread-after-getting-new-news` を呼ぶことによって成し遂げられます。これらの変数をデフォルトではない値に変えている場合は、重ねて `nnimap-fixup-unread-after-getting-new-news` を追加する必要があるかもしれません。見積りに満足していて、新しいメールを取得するときにいくらかの (わずかな) 時間を節約したいのであれば、その関数を外して下さい。

- グループ名は、デフォルトで UTF-8 であるものとして取り扱われます。
これは USEFOR が移行しようとしていると想定されるものです。カスタマイズするには `gnus-group-name-charset-group-alist` および `gnus-group-name-charset-method-alist` を参照して下さい。
- `gnus-group-charset-alist` と `gnus-group-ignored-charsets-alist`
これらの変数に設定された正規表現は、完全な (full) グループ名と比較されます。Gnus 5.8 では実際の (real) グループ名が比較の対象でした。したがって、これらの変数をカスタマイズしている利用者は、正規表現を変更しなければなりません。例です：

```
("^han\\>" euc-kr) -> ("\\(^\\|:\\|)han\\>" euc-kr)
```
- 概略モードと記事モードの変更
 - 領域が活性化されている場合に、*F* キー (`gnus-article-followup-with-original`) および *R* キー (`gnus-article-reply-with-original`) は、その領域にあるテキストだけを `yank` します。
 - ドラフト・グループで *e* キーが `gnus-draft-edit-message` コマンドに割り当てられました。`gnus-summary-edit-article` コマンドには、代わりに *B w* キーを使って下さい。
 - 記事のボタン。
URL、メールアドレス、Message-ID、Info へのリンク、man ページと Emacs または Gnus に関連した参考文献のための、より多くのボタンが追加されました。See Section 3.17.6 [Article Buttons], page 89. すべての記事のボタンの見栄えを制御するために `gnus-button-*-level` 変数を使うことができます。See Section 3.17.7 [Article Button Levels], page 91.
 - 単一の `yenc` でエンコードされた添付パートをデコードすることができます。
 - Picons
Picon のコードが、GNU Emacs で動作させるために再実装されました。以前のいくつかのオプションが、削除または改名されています。
Picon は、利用者、ドメイン、それにニュースグループを表現するための「個人的なアイコン (personal icons)」で、記事バッファーに表示することができます。See Section 8.17.4 [Picons], page 268.
 - 新しいオプション `gnus-treat-body-boundary` を非-nil にすると、ヘッダーのおしまいに境界線が描かれます。
 - 署名された記事のヘッダー (X-PGP-Sig) を、*W p* で認証することができます。
 - 概略バッファーは `fringe` の中の矢印で現在の記事を示します。これを無効にするには (`setq gnus-summary-display-arrow nil`) を使って下さい。
 - ニュースにメールで返信しようとしたら警告します。
間違っってニュースにメールで返信しようとしてしまうことが、しょっちゅうありませんか？ そんなあなたに新オプション `gnus-confirm-mail-reply-to-news`。
 - 新しいオプション `gnus-summary-display-while-building` を非-nil にすると、概略バッファーが作られていく様子が表示されます。
 - 新しい recent 印 ‘.’ で、新規に届いたメッセージを (未読だけれども古い記事とは区別して) 表示します。
 - Gnus は RFC 2369 のメーリングリストのヘッダーをサポートします。また、メーリングリストのグループ用に数々のコマンドを用意しました。See Section 3.31 [Mailing List], page 112.

- 日付ヘッダーを、英語で発音できる形式で表示することができます。See Section 3.17.8 [Article Date], page 91.
- `mm-u-diff-groups-regexp` に合致するグループでは、差分 (diffs) が自動的にハイライトされます。
- マイクロソフト引用様式のより良い取り扱い。
いくつかのマイクロソフトのメイラーが、メッセージの残りの部分が引用であることを示すために使う台無しにされたヘッダーブロックを、たとえそれが引用符で囲まれていなくても、Gnus はとにかく認識しようとします。変数 `gnus-cite-unsightly-citation-regexp` は、それらの引用の先頭に合致します。
新しい `W Y f` コマンド (`gnus-article-outlook-deuglify-article`) で、醜く壊れた Outlook (Express) の記事を整形し直すことができます。
- `gnus-article-skip-boring`
`gnus-article-skip-boring` を `t` に設定すると、Gnus はうんざりする文しか含んでいないページを見せるために、下方にスクロールしません。`gnus-article-boring-faces` を使って、何を読み飛ばしてしまっても良いかをカスタマイズすることができます。
てっぺんに少しだけある新規な内容に、長くて刈り込まれていない引用が続いているたくさんの記事を読む場合に、これは特に役に立ちます。
- スマイリー (‘:-)’、‘;-)’ など) が Emacs でもアイコン化されるようになりました。
これを働かないようにするには、(`setq gnus-treat-display-smileys nil`) を ‘`~/gnus.el`’ に置いて下さい。
- Face ヘッダーを扱えるようになりました。See Section 8.17.2 [Face], page 267.
- 概略バッファで、新しいコマンド `/N` は新着メッセージを挿入し、`/o` は古いメッセージを挿入します。
- `W m` を押すと、Gnus はモールスでエンコードされたメッセージをデコードします。
- `gnus-summary-line-format`
デフォルト値が ‘`%U%R%z%I%([%4L: %-23,23f%]) %s\n`’ に変わりました。さらに、受信者の名前が NNTP グループに投稿したグループ名で利用者を置き換えるために、`gnus-extra-headers`、`nnmail-extra-headers` および `gnus-ignored-from-addresses` のデフォルト値が変わりました。
- 添付ファイルの消去。
`gnus-mime-save-part-and-strip` コマンド (MIME ボタン上で `C-o` に割り当てられている) は、パートをセーブしてから外部のそれと置き換えます。`gnus-mime-delete-part` (MIME ボタン上で `d` に割り当てられている) は、パートを削除します。これは編集をサポートしているバックエンドでだけ動作します。
- `gnus-default-charset`
デフォルト値は `iso-8859-1` に代わって `current-language-environment` 変数によって決定される値になります。また、`gnus-group-charset-alist` にあった ‘`.*`’ の項目は削除されました。
- 印刷の性能が向上しました。
Gnus はそれ自身が、概略と記事バッファにおける `OP` で Muttprint をサポートします。さらに MIME ボタン上で `p` を使うことによって、個々の MIME パートのそれぞれを印刷することができます。

- 拡張された書法仕様 (format specs)。

書法仕様 ‘`%&user-date;`’ が `gnus-summary-line-format-alist` に追加されました。それに、利用者定義による拡張されたフォーマットの仕様もサポートされています。拡張された書法仕様は ‘`%u&foo;`’ のようなもので、関数 `gnus-user-format-function-foo` を起動します。‘`&`’ がエスケープ文字に使われているので、古い利用者定義書法である ‘`%u&`’ は今ではサポートされていません。
- `/ *` (`gnus-summary-limit-include-cached`) が書き直されました。

これは `Y c` (`gnus-summary-insert-cached-articles`) の別名でした (訳注: 以前は)。新しい関数は他の記事を濾過して除去します。
- いくつかの制限命令は `C-u` 接頭引数で合致の否定を扱うことができます。

`C-u` を `subject`、`author` または `extra` ヘッダー、すなわち `/ s`、`/ a` および `/ x` (`gnus-summary-limit-to-{subject,author,extra}`) で使うと、結果としてその表現に合致しないすべての記事が表示されます。
- Gnus は外部パート (`message/external`) をインライン表示します。
- Message モードの変更と関連する Gnus の機能
 - 遅延記事。

Message バッファにおける `C-c C-j` で、メッセージの送信を遅らせることができます。メッセージは指定された時刻に配送されます。これはあなた自信のための忘備録として役に立つでしょう。See Section 3.6 [Delayed Articles], page 55.
 - `auto-compression-mode` が有効になっていると、添付ファイルを見るときに自動で圧縮が解かれます。
 - 新しいオプション `gnus-gcc-mark-as-read` は、Gcc の記事に自動的に既読の印を付けます。
 - 添付ファイルの切り離し (`externalizing`)。

`gnus-gcc-externalize-attachments` または `message-fcc-externalize-attachments` が非-`nil` になっていると、ローカルファイルを外部パートとして添付します。
 - Sendmail を使うときのエンベロープ送信者 (`envelope sender`) のアドレスが、カスタマイズできるようになりました。See section “メール変数” in *The Message Manual*.
 - Gnus は今では `Sender`: ヘッダーを自動では生成しません。

初期のころ、それは利用者が設定できる `email` アドレスが、Gnus が想定した利用者のデフォルトのアドレスと違っていただけに限り生成されました。今日ではその想定アルゴリズムが正しいことはまれで、`Sender`: ヘッダーの唯一の (議論的になる) 用途は、ニュースを `cancel/supersede` する資格があるかどうかを検査すること (これは代わりに、他の章で述べられる `Cancel Locks` によって解決されました) なので、そのヘッダーの生成はデフォルトで抑制されています。変数 `message-required-headers`、`message-required-news-headers` および `message-required-mail-headers` を参照して下さい。
 - サードパーティーによる ‘`message-utils.el`’ の機能が ‘`message.el`’ に加えられました。

Message は表題の行から ‘(was: <old subject>)’ を削除するかどうかを尋ねるようになりました (`message-subject-trailing-was-query` 参照)。`C-c M-m` と `C-c M-f` は挿入されたテキストを示す印を挿入します。`C-c C-f a` は `X-No-Archive`: ヘッダーを付け加えます。`C-c C-f x` は、適切なヘッダーと、クロスポストとフォロー先についての注意書きを本文に挿入します (`message-cross-post-*` 変数群を見て下さい)。

- 今や `message-generate-headers-first` が `nil` だったら、メッセージの作成を始めるときに `References` と `X-Draft-From` ヘッダーは生成されません。
- `X-Faces` ヘッダーの挿入が簡単になりました。See Section 8.17.1 [X-Face], page 265.
- グループカーボンコピー (GCC) を引用符で囲む。

空白や他の変な文字を含むグループを扱えるようにするために、グループは `Gcc: header` に置かれる前に引用符で囲まれます。これは、空白を含むグループが使えるようにするために、もはや `gnus-message-archive-group` のような変数に引用文字を含めるべきではないことを意味します。さらに、文字列 `'nnml:foo, nnml:bar'` (二つのグループに `Gcc` を格納することを示す) を使っているならば、`("nnml:foo" "nnml:bar")` というリストを返すように変更しなければなりません。さもないと、`Gcc:` 行は間違った囲まれ方をされてしまうでしょう。初期のころに文字列 `'nnml:foo, nnml:bar'` を返すようにしたことが間違いだったことに着目して下さい。それは直接挿入されたので、まったく問題を生じませんでした。

- `message-insinuate-rmail`

`(message-insinuate-rmail)` と `(setq mail-user-agent 'gnus-user-agent)` を `~/ .emacs` に加えることによって、`message-mode` でメッセージの作成、返信および転送を行なうように `Rmail` を説得することができます。そこでは `MML` の威力を堪能することができます。

- `message-minibuffer-local-map`

この下の行は、メッセージを再送するときに `BBDB` を使えるようにします:

```
(define-key message-minibuffer-local-map [(tab)]
  'bbdb-complete-name)
```

- `gnus-posting-styles`

このような合致の様式が加わりました。

```
((header "to" "larsi.*org")
 (Organization "Somewhere, Inc."))
```

下記のような古い様式は時代遅れになりましたが、まだ受け入れられます。

```
(header "to" "larsi.*org"
 (Organization "Somewhere, Inc.))
```

- `message-ignored-news-headers` と `message-ignored-mail-headers`

`'X-Draft-From'` と `'X-Gnus-Agent-Meta-Information'` が、これら二つの変数に加えられています。それらをカスタマイズする場合に、もしかするとそれら二つのヘッダーも加える必要があります。

- `Gnus` は `"format=flowed"` (RFC 2646) パラメーターをサポートします。メッセージを作成するときに、それは `use-hard-newlines` で活性化されます。`format=flowed` のデコードは以前からできましたが、初期の版では説明の文書がありませんでした。
- `mm-fill-flowed` オプションで `"format=flowed"` メッセージを流動テキストとして処理することをやめさせることができます。また、PGP 署名が埋め込まれたメッセージを送信するとき、流動テキストの処理は行われません。See section “流動テキスト” in *The Emacs MIME Manual*. (`Gnus` 5.10.7 の新機能)
- `Gnus` は RFC 2298 の開封確認要求の生成をサポートします。
これはメッセージモードの `C-c M-n` キーで呼び出されます。

- Message は Importance: ヘッダー (RFC 2156) をサポートするようになりました。
メッセージバッファで `C-c C-f C-i` か `C-c C-u` を使うと、可能な値が循環します。
- Gnus はニュースの Cancel Locks をサポートします。
投稿するニュース記事に ‘Cancel-Lock’ ヘッダーが挿入されることです。これは、記事をあなたが書いたのかどうかを確かめるために使います (キャンセルと置き換えのとき)。最初に記事を投稿するときに、Gnus はランダムなパスワード文字列を生成し、カスタムの機構を使って ‘~/ .emacs’ にセーブします。その変数は `canlock-password` と呼ばれますが、機密を気にするデータではありません。ウェブ上で `canlock` を公開しても、以前から彼女にできなかった何かを、誰かができるようにするものではありません。 `message-insert-canlock` をカスタマイズすることによって、振る舞いを変更することができます。
- Gnus は PGP (RFC 1991/2440)、PGP/MIME (RFC 2015/3156) および S/MIME (RFC 2630-2633) をサポートします。
これには S/MIME と OpenPGP が実装されている必要があります。でも追加の Lisp ライブラリーは要りません。メッセージの作成時に、いくつかのメニューと `C-c RET` キーの割り当てが Attachments メニューに追加されます。これはまた、 `gnus-article-hide-gpg-hook` を時代遅れにしました。
- MML (Mime 作成) コマンドの接頭キーが、 `M-m` から `C-c C-m` に変わりました。
この変更によって、標準キー割り当ての `back-to-indentation` との衝突が回避されました。このコマンドもまた、メッセージモードでは役に立つのです。
- `message-forward-show-mml` のデフォルトが `best` というシンボルに変わりました。
値 `best` の振る舞いは、それがふさわしい場合は MML を表示する (すなわち MIME に変換する) ことです。変換がデジタル署名を無効にしてしまうので、署名された、または暗号化されたメッセージを転送するときは MML は使われません。
- `auto-compression-mode` が有効になっていると、添付ファイルを見るときに自動で圧縮が解かれます。
- 非-ASCII ドメイン名のサポート。
Message は From:, To: および Cc: にある非-ASCII ドメイン名をサポートし、メッセージの送信をしようとしたときにエンコードするかどうかを尋ねます。 `message-use-idna` 変数でこれを制御します。Gnus もまた、メッセージを見るときに From:, To: および Cc: にある非-ASCII ドメイン名をデコードします。これを制御するのは `gnus-use-idna` 変数です。
- Message バッファに添付ファイルをドラッグ&ドロップすることができます。
`mml-dnd-protocol-alist` と `mml-dnd-attach-options` を参照して下さい。 See section “MIME” in *The Message Manual*.
- バックエンドの変更
 - Gnus は RSS のニュース配送を、ニュースグループとして表示します。 See Section 6.4.6 [RSS], page 183.
 - `nndoc` バックエンドは、`mailman` のまとめ送りと `exim` が弾いたメッセージをサポートするようになりました。
 - Gnus は Maildir グループをサポートします。
Gnus は新バックエンドである ‘`nnmaildir.el`’ を含んでいます。 See Section 6.3.13.5 [Maildir], page 171.

- nnml と nnfolder バックエンドは、グループ毎に印 (marks) を格納するようになりました。
これは nnml/nnfolder サーバー/グループ を ‘~/newsrsrc.eld’ と切り離して、しかし印は守りつつ、バックアップすることを可能にします。さらに、例えば研究室や職場などの組織内で、(‘~/newsrsrc.eld’ ファイルを共有すること無しに) 利用者間で記事と印を共有することをも可能にします。これは、‘~/newsrsrc.eld’ に格納される印を、グループ毎の ‘.marks’ ファイル (nnml 用) と ‘groupname.mrk’ ファイル (groupname の名前を持つ nnfolder 用) に格納することによって動作します。nnml/nnfolder を他のマシンに引っ越しても、Gnus は ‘~/newsrsrc.eld’ にある情報の代わりに、自動的に ‘.marks’ か ‘.mrk’ ファイルを使います。新しいサーバー変数である nnml-marks-is-evil と nnfolder-marks-is-evil が、この機能を抑制するために使うことができます。
- 外見に関すること
 - グループと概略バッファのメニューバー項目の名前“ Misc ”は“ Gnus ”に改名されました。
 - Message mode で“ MML ”と名付けられたメニュー項目は“ Attachments ”に改名されました。このメニューは、署名と暗号化 (see section “セキュリティー” in *The Message Manual*) のような、セキュリティーに関連したものも含んでいることに着目して下さい。
 - ツールバーがグループ、概略および Message モードで GNOME のアイコンを使うように更新されました。ツールバーはカスタマイズ可能です。これは Gnus 5.10.9 の新機能です。(Emacs だけです。XEmacs は未対応。)
 - The tool bar icons are now (de)activated correctly グループバッファで変数 gnus-group-update-tool-bar を参照して下さい。そのデフォルト値は Emacs のバージョンに依存しています。これは Gnus 5.10.9 の新機能です。
- その他の変更
 - gnus-agent
Gnus エージェントは大規模な更新を経て、今やデフォルトで有効になります。そして gnus-select-method と gnus-secondary-select-method で指定されるすべての nntp と nnimap のサーバーが、デフォルトでエージェント化されます。初期においては gnus-select-method のサーバーだけがデフォルトでエージェント化され、エージェントはデフォルトでは有効にされませんでした。エージェントが有効にされると、今では可能ならばバックエンドに代わってエージェントのキャッシュからヘッダーが取り寄せられます。初期には、これはオフライン (unplugged) の状態でのみ行なわれていました。サーバーバッファで *J a* と *J r* を使うことによって、サーバーの登録と削除を行なうことができます。グループバッファから *J u* か *J s* を使って命令しない限り、Gnus は記事をエージェントのキャッシュにダウンロードしません。(setq gnus-agent nil) を設定することによって、エージェントが有効にされていなかった昔の振る舞いに戻すことができます。もはや ‘~/gnus.el’ に (gnus-agentize) を置いておく必要が無いことに注意して下さい。
 - Gnus は plugged のときに、エージェントに NOV と記事を読み込みます。
Plugged のときに記事を読む場合に、その記事がすでにエージェントにあるならば、もう一度ダウンロードすることはありません。(setq gnus-agent-cache nil) は旧式の動作に戻します。
 - Dired の統合。
gnus-dired-minor-mode (Section 8.21 [Other modes], page 296 参照) は dired のバッファで、添付ファイルを送信する、mailcap の適切な項目を使ってファイルを開く、それに mailcap の項目を使ってファイルを印刷するためのキーを割り当てます。

- ポイントの位置決めのための書法仕様 (format spec) である %C は、%* に変更されました。
- gnus-slave-unplugged
オフラインの Gnus をスレーブモードで起動する新しいコマンドです。

10.2.9.7 No Gnus

No Gnus の新しい機能:

- インストールに関する変更
 - No Gnus を使ったことがあるが、以前の (安定した) 版に戻ってしまった人たちへの注意。
No Gnus (このリリースにつながる不安定な Gnus の枝) を試してみたものの、安定版に戻してしまっている場合、このバージョンへアップグレードするときには注意して下さい。特に、`~/News/marks` ディレクトリーの内容を (もしかしたら注意深く選んで) 削除する必要があるでしょう。(訳注: 削除しないと、新しい No Gnus を初めて使ったときに、安定版を使っていなかった時期に更新されなかった marks ファイルが読み込まれて、`~/newsrsrc.eld` ファイルの内容を上書きしてしまいます。) 削除することによって、このリリースで nntp のフラグを保存している marks ファイルからではなく `~/newsrsrc.eld` からフラグを読むようになります (訳注: そして新しい marks ファイルが作られます)。nntp marks については、次の項目でさらに詳しい情報を得ることができます。一般にダウングレードすることは安全ではありません。
 - Lisp ファイルがデフォルトで `.../site-lisp/gnus/` にインストールされるようになりました。以前は `.../site-lisp/` がデフォルトでした。加えて、新しいインストーラーは他にインストールされている、新しい Gnus より優先される Gnus を検出して警告を発します。それらを手動で取り除いても良いし、`make remove-installed-shadows` を使って削除することもできます。
- Gnus に含まれる新しいパッケージとライブラリー
 - Gnus は Emacs Lisp SASL ライブラリーを含むようになりました。
これによって、Emacs の中から SASL の機構を利用するために、すっきりした API を使うことができます。利用者の目に見える利点は、以前は無かった DIGEST-MD5 と NTLM がサポートされるようになったことです。See section “Emacs SASL” in *Emacs SASL*.
 - ManageSieve の接続に、デフォルトで SASL ライブラリーを使うようになりました。
これによる主な変更点は、サーバーがサポートしている場合に DIGEST-MD5 と NTLM をサポートするようになったことです。
 - Gnus は password.el にパスワードをキャッシュする機構を含めました。
パスワードキャッシュはデフォルトで有効です (password-cache を参照)。タイムアウトは 16 秒と短いです (password-cache-expiry を参照)。PGG を PGP のバックエンドとして使う場合に、PGP のパスフレーズはこの機構で管理されます。ManageSieve 接続のパスワードは、利用者にそうするかどうかを尋ねてから、この機構が管理します。
- 概略モードと記事モードの変更
 - 国際化ホスト名 (IDNA) を、`Wi` (gnus-summary-idna-message) を使うことによって、記事の本文中でデコードできるようになりました。この機能を使うには GNU Libidn (<http://www.gnu.org/software/libidn/>) をインストールしておく必要があります。
 - Gnus は dns-mode を使って text/dns として送信された DNS マスターファイルを表示します。

- Gnus は概略バッファで新しい制限コマンド `/r` (`gnus-summary-limit-to-replied`) と `/R` (`gnus-summary-limit-to-recipient`) をサポートします。See Section 3.8 [Limiting], page 61.
- `Y t` (`gnus-summary-insert-ticked-articles`) を使って、サーバーからすべての可視記事を取り寄せることができるようになりました。See Section 3.26.3 [Summary Generation Commands], page 107.
- Gnus は概略バッファで新しい並べ替えコマンド `C-c C-s C-t` (`gnus-summary-sort-by-recipient`) をサポートします。See Section 3.21 [Summary Sorting], page 98.
- S/MIME が LDAP の利用者証明書の検索に使えるようになりました。`smime-ldap-host-list` でサーバーを設定する必要があります。
- OpenPGP ヘッダーにある URL をクリックすると、ヘッダーがダウンロードされてあなたの PGP の鍵束に取り込まれます。
- Picon はテキストの対象物の右側に表示できるようになりました。`gnus-picon-style` を見て下さい。See Section 8.17.4 [Picons], page 268.
- ANSI SGR 制御シーケンスを `W A` で変換することができます。
中国語のニュース階層のグループにおいて、記事をハイライト表示するために ANSI シーケンスが使われます (`gnus-article-treat-ansi-sequences`)。
- Gnus は記事に“ MIME-Version ”ヘッダーがなくても記事を MIME デコードします。このために `gnus-article-loose-mime` のデフォルト値が変更されました。
- `gnus-decay-scores` をスコアファイルに合致する正規表現にできます。例えば `'\\.ADAPT\\.'` に設定すると、適応スコアファイルだけが減衰されるようになります。See Section 7.16 [Score Decays], page 247.
- `gnus-ignored-from-addresses` を使う場合に、概略行において To と Newsgroup ヘッダーに相当する場所の最初に表示する文字列を、`gnus-summary-to-prefix` および `gnus-summary-newsgroup-prefix` でカスタマイズすることができます。See Section 3.1.2 [To From Newsgroups], page 46.
- MIME パートを外部にある本体で置き換えることができます。`gnus-mime-replace-part` と `gnus-article-replace-part` を見て下さい。See Section 3.18 [MIME Commands], page 94, Section 4.2 [Using MIME], page 116.
- `mm-fill-flowed` オプションで `format=flowed` なメッセージの取り扱いを無効にすることができます。また、PGP の署名が埋め込まれたメッセージを送信するときに、`flowed text` は無効にされます。See section “流動テキスト” in *The Emacs MIME Manual*. (Gnus 5.10.7 の新機能)
- Message モードの変更
 - Gnus は“ `hashcash` ”client puzzle anti-spam の機構をサポートします。(`setq message-generate-hashcash t`) で有効になります。See Section 8.19.4 [Hashcash], page 273.
 - メッセージバッファに添付ファイルをドラッグ&ドロップできます。`mml-dnd-protocol-alist` と `mml-dnd-attach-options` を見て下さい。See section “MIME” in *Message Manual*.
 - `message-yank-empty-prefix` オプションで、引用文の空行にどんな引用符を付けるかを制御することができます。See section “挿入するための変数” in *Message Manual*.
 - Gnus はメッセージバッファでヘッダーを隠すために、それら以外の部分だけが見えるようにバッファを狭めます。References はデフォルトで表示されません。すべての

ヘッダーが見えるようにするには (setq message-hidden-headers nil) として下さい。See section “メッセージヘッダー” in *Message Manual*.

- You can highlight different levels of citations like in the article buffer. See `gnus-message-highlight-citation`.
 - 記事バッファでできるのと同様に、引用された文のレベルの違いに応じたハイライトを行うことができます。 `gnus-message-highlight-citation` を参照して下さい。
 - Message モードでは `auto-fill-mode` がデフォルトで ON になります。 `message-fill-column` を参照して下さい。 See section “メッセージヘッダー” in *Message Manual*.
 - 署名ファイルを `message-signature-directory` 変数で指定するディレクトリーに置くことができます。
- バックエンドの変更
- nntp バックエンドは記事の印を ‘~/News/marks’ に保管します
そのディレクトリーは `nntp-marks-directory` という (カスタマイズ可能な) 変数で変更することができます。また、nntp で印を使うことを `nntp-marks-is-evil` という変数 (バックエンド変数) で無効にすることができます。印を使うことの利点は、‘~/News/marks’ を別のホストにインストールされた Gnus にも (rsync, scp などを使って) コピーすることによって、どの記事を読んでどの記事に印を付けたかの情報を、そこでも維持できることです (訳注: 同じ nntp サーバーに接続する場合に限ります)。(‘~/News/marks’ のデータは ‘~/newsrsrc.eld’ にある同じデータより優先されます。
 - RSS の購読情報を OPML のファイルから取り込み、または書き出すことができます。 See Section 6.4.6 [RSS], page 183.
 - IMAP の identity (RFC 2971) をサポートします。
デフォルトでは Gnus はそれ自身に関する情報を送信しませんが、`nnimap-id` 変数を使ってそれをカスタマイズすることができます。
 - nnrss バックエンドは多言語テキストをサポートします。nnrss グループでは非-ASCII 文字列を使ったグループ名もサポートされます。 See Section 6.4.6 [RSS], page 183.
 - POP3 によるメールの取得において、SSL/TLS と StartTLS をサポートするようになりました。
 - nnml バックエンドではメッセージを圧縮するために ‘gzip’ 以外のプログラムも使うことができます。 See Section 6.3.13.3 [Mail Spool], page 169.
 - nnml バックエンドではグループを圧縮することができます。
関数 `gnus-group-compact-group` (グループバッファの `Gz` キー) および `gnus-server-compact-server` (サーバーバッファの `z` キー) で呼び出すこの機能は、グループのすべての記事の番号を 1 から順に振り直して、すきまを取り除きます。その結果として、正しい全記事数を得ることができます (再びメッセージが削除されるまでは)。
- 外見
- ツールバーが GNOME のアイコンを使うように更新されました。ツールバーをカスタマイズすることもできます。まだマニュアルで文書化されていませんが、`M-x customize-apropos RET -tool-bar$` で始めることができます (Emacs だけです。XEmacs は未対応。)
 - ツールバーのアイコンがグループバッファで正しく活性化 (または不活性化) されるようになりました。変数 `gnus-group-update-tool-bar` を参照して下さい。そのデフォルト値は Emacs の版に依存します。

- Gnus のバッファーにおける XEmacs の toolbar の位置を変更できるようになりました。
`gnus-use-toolbar` と `message-use-toolbar` を見て下さい。
- その他の変更
 - サーバーバッファーで外部グループのための `select-method` を変更すると、すぐにそのサーバーを使うグループの講読に反映されるようになりました。例えば `nntp-via-address` を `'foo.example.com'` から `'bar.example.com'` に変更すると、Gnus は次回から中間ホスト `'bar.example.com'` を経由してニュースサーバーに接続するようになります。
 - `W e` で `'all.SCORE'` ファイルをグループバッファーから編集することができます。

10.3 マニュアル

このマニュアルは TeXinfo ファイルから作成され、それから texi2dvi を通して、あなたの手元にあるものになりました。

以下の習慣が用いられました:

1. これは ‘文字列’ です。
2. これは キー打鍵 です。
3. これは ‘ファイル’ です。
4. これは シンボル です。

ですから、私が「flargnoze を ‘yes’ に設定する」と言ったときは、次のような意味です:

```
(setq flargnoze "yes")
```

もし、私が「flargnoze を yes に設定する」と言ったときは、次のような意味です:

```
(setq flumphel 'yes)
```

‘yes’ と yes は二つの まったく 違ったものです—絶対に混同しないで下さい。

10.4 マニュアルを書く

おそらく、たいていのマニュアルは事後に書かれていると思います。つまり、すでにあるプログラムを文書化しているということです。このマニュアルはそういう方法で書かれていません。何かを実装するときは、何かをそのままマニュアルの一節に書きます。それから機能の説明が難しいことを発見して、それがどのようにあるべきであるかを書き、次には実装を変更します。文書とコードを書くことは協調して行なわれていきます。

もちろん、これはこのマニュアルには流れ構造がほとんど無いが、あっても少しだということを意味します。Gnus の完全にすべてのことが説明されていますが、あなたが探している場所ではないということがよくあります。これはリファレンスマニュアルであって、Gnus を始めるための手引きではありません。

それはこのリファレンスマニュアルを元にして書かれた、まったく違った本になるでしょう。とても違ったものになるはずで。

10.5 用語

ニュース (news)

これは、あなたがそれを読むために使うことになっているもの、つまり、それというのがニュースです。ニュースは一般的には近くの NNTP サーバーから取得され、一般的にはすべての人が公に利用することができます。もしニュースを投稿すると、あなたがまさに書いたものを全世界の人たちが読むことになるでしょう。そして、みんながいたずらっぽくクスクス笑うでしょう。あなたの知らないところで。

メール (mail)

あなたに個人的に配送されるすべてのものがメールです。いくつかのニュース/ メールリーダー (Gnus のような) はメールとニュースの区別を曖昧にしますが、違いがあります。メールは私的です。ニュースは公的です。メールを送信することは投稿ではなく、返信はフォローアップではありません。

返信 (reply)

あなたが読んでいるものを書いた人にメールを送ることです。

フォローアップ (follow up)

あなたが読んでいる記事に応答して、現在のニュースグループに記事を投稿することです。

バックエンド (back end)

Gnus はメールとニュースがほとんど同じだとみなします。本当に。違いは実際の記事にどのようにアクセスするかだけです。メールメッセージはローカルディスクのファイルから読めるのに対して、ニュース記事は一般に NNTP プロトコルで取得します。Gnus の内部構造は、それらのために「フロントエンド」と数々の「バックエンド」から成り立っています。内部的には、あなたがグループに入る (そう、RET をたたく) と、それによって Gnus のフロントエンドの機能呼び出します。そうするとフロントエンドは、バックエンドに「foo グループの記事のリストをくれ」とか「4711 番の記事を見せてよ」と「話す」のです。

そういうわけで、バックエンドは主にプロトコルか、ファイルの形式とディレクトリーの配置のどちらかを定義します。前者は nntp バックエンドが NNTP でニュースにアクセスしたり、nnimap バックエンドが IMAP でメールにアクセスすることを指します。また、後者は nnsPOOL バックエンドが共通の「スプールディレクトリー」形式にアクセスしたり、それととてもよく似たファイルの形式とディレクトリーの配置を介して nnml バックエンドがメールにアクセスすることを指します。

Gnus は基礎的なメディアを扱いません。言わばこれは、すべてバックエンドによって行なわれるということです。バックエンドは記事にアクセスするための機能の集成です。しかし、「バックエンド」という用語は「サーバー」と言った方がふさわしい場面ときどき使われます。そして同じことを指すことができる「選択方法」(select method) という用語があります。かように Gnus の用語はとても混乱しています。

基本 (native)

Gnus はいつも一つの方法 (とバックエンド) を、ニュースを得るための「基本」もしくはデフォルトの手段として使います。

外部 (foreign)

同時に任意の数の外部グループにアクセスできる状態にすることもできます。これらはニュースを取得するための、基本ではなく、二次のでもないバックエンドを使うグループです。

二次の (secondary)

二次のバックエンドは、基本と外部の間くらいに位置するバックエンドですが、ほとんど基本と同じように動作するものです。

記事 (article)

ニュースとして投稿されたメッセージです。

メールメッセージ (mail message)

メールで送られたメッセージです。

メッセージ メールメッセージもしくはニュース記事です。

ヘッド (head)

メッセージの最上部で、管理情報 (等) が入れられているところです。

本文 (body)

記事の残りの部分です。ヘッドに無いものはすべて本文です。

ヘッダー (header)

記事のヘッドの行です。

ヘッダー群 (headers)

そのような行の集合もしくは、ヘッドの集合です。もしくは、NOV 行の集合です。

NOV

グループに入ると、Gnus はグループのすべての未読記事のヘッダーをバックエンドに要求します。ほとんどのサーバーは News OverView 様式をサポートしています。それは標準の HEAD 様式よりコンパクトで、とても速く、読んで解析することができます。

レベル (level)

それぞれのグループは何らかの「レベル」(1-9) で購読されています。低いレベルのものは高いレベルのものより「より」購読されています。実際のところ、レベル 1-5 のグループは「購読」；6-7 は「未購読」；8 は「ゾンビ」；9 は「切られた」(killed) と見なされます。グループの一覧を表示したり、新しい記事を走査する命令は、すべて数値接頭引数を「動作レベル」として使います。

切られたグループ (killed groups)

切られたグループの情報は保存されたり更新されたりしないので、切られたグループを扱うのは購読されているグループよりも簡単です。

ゾンビグループ (zombie groups)

ほとんど切られたグループと同じで、それより少し死んでいるだけです。

アクティブファイル (active file)

ニュースサーバーは、どの記事を持っているかどのグループが存在するかを記録しておかなければなりません。アクティブファイルに格納されるすべてのこの情報は、あなたが推測するように比較的大きいです。

偽グループ (bogus groups)

‘.newsrc’ ファイルに存在するけれどもサーバーが知らないグループ (すなわち、それはアクティブファイルにありません) は 偽グループ です。おそらくそのグループは (もはや) 存在していないでしょう。

活性化 (activating)

サーバーにグループの情報を尋ねて未読記事の数を演算する行為は「グループを活性化 (activate) する」と呼ばれています。活性化されていないグループは、グループバッファに ‘*’ とともに一覧表示されます。

スプール (spool)

ニュースサーバーは何らかのやり方で記事をローカルに保存します。ある古い流儀の保存方法は、単に記事毎に一つのファイルを持つことです。それは伝統的なスプール (traditional spool) と呼ばれます。

サーバー (server)

接続して、ニュース (もしくはメール) を取得することができるマシンです。

選択方法 (select method)

バックエンドと、サーバーおよび仮想サーバーの設定を指定する構造です。

仮想サーバー (virtual server)

名前が付けられていて、その名前で指定することができる選択方法です。選択方法は (物理的な) サーバーに関するすべてを定義するので、ものごとを全体として捉えるのは仮想サーバーになります。

洗濯 (washing)

バッファを持ってきて、何らかの種類のフィルターにかけることです。結果は (たいてい) 元のものよりもよりきれいで喜ばしいものになるでしょう。

一時グループ (ephemeral groups)

たいていのグループはどの記事を読んだかのデータを保存します。「一時」グループはデータが溜められないグループです—グループを出ると、それは天空のかなたに消え去ります。

固定グループ (solid groups)

これは一時グループの反対です。グループバッファに一覧表示されているすべてのグループは固定グループです。

まばら記事 (sparse articles)

gnus-build-sparse-threads が有効にされているときに、それらは概略バッファに表示される (存在しない) 記事のための場所取りです。

スレッド化 (threading)

応答の記事を、それが応答した元記事の直後に置くことです—階層的な流儀で。

根 (root) スレッドの最初の記事が根です。それはスレッドのすべての記事の祖先です。

親 (parent)

応答が得られた記事です。

子 (child) それとは別の記事、すなわち親に応答する記事です。

まとめ送り (digest)

複数のメッセージを一つのファイルに集めたものです。最も一般的なまとめ送りの様式は、RFC1153 で規定されています。

分割 (splitting)

ある規則によってメールを区分けする行為です。ときどき間違ってメールの濾過 (filtering) と呼ばれます。

10.6 カスタマイズ

すべての変数は、このマニュアルのどこか他のところで適切に説明されています。この章は、非常に良くある状況でどのように Gnus をカスタマイズすれば良いかを調べるための、総合的な案内になるように作られています。

10.6.1 遅くて高価な NNTP 接続

Emacs をローカルのマシンで実行していて、非常に細いひもの向こうのマシンからニュースを取り寄せているとしたら、Gnus が NNTP サーバーから取って来なければならないデータの総量を減らしたくなるでしょうね。

`gnus-read-active-file`

これを `nil` にして下さい。これは Gnus がサーバーにアクティブファイル全体を要求することを禁止します。このファイルはしばしば非常に大きいです。さらに、Gnus が不意にアクティブファイルをとにかく取り寄せようと決意しないように、`gnus-check-new-newsgroups` および `gnus-check-bogus-newsgroups` も `nil` に設定する必要があります。

`gnus-nov-is-evil`

これも `nil` にしていなければなりません。そうしておかないと、記事のヘッダーを NNTP サーバーから掘み取ってくるのが、あまり速くなりません。もっとも、すべての NNTP サーバーが XOVER をサポートしているわけではありません。そのことは Gnus が自分で検査します。

10.6.2 遅いターミナル接続

Emacs と Gnus を実行しているシステムに、家のコンピューターをダイヤルアップで接続しているとしましょう。モデムが遅い場合は、電線を伝って送られているデータの総量を（可能な限り）減らしたくなるでしょう。

`gnus-auto-center-summary`

Gnus が概略バッファをリセンターする（訳注：現在の記事が真ん中に表示されるようにする）するために、これを `nil` に設定して下さい。これが `vertical` だったら、垂直方向のリセンターだけをします。`nil` でも `vertical` でも無ければ、水平方向と垂直方向の両方でリセンターを行ないます。

`gnus-visible-headers`

記事に含まれるヘッダーを最小限に減らします。実際のところ、それらが無くてもすべて間に合わせることができます—たいていの役に立つデータは、とにかく概略バッファにありますから。この変数を ‘`NEVVVVER`’ や ‘`From:`’ や、何でも必要になりそうなものに設定して下さい。

利用できるすべての「隠す」機能を有効にするために、以下を使って下さい:

```
(setq gnus-treat-hide-headers 'head
      gnus-treat-hide-signature t
      gnus-treat-hide-citation t)
```

`gnus-use-full-window`

これを `nil` に設定することによって、すべてのウィンドウを小さくすることができます。これは総じてそんなに減らしませんが、この記事は何が何でも読みたくなかったんだと決心する前に、それを少ししか見ないで済みます。

gnus-thread-hide-subtree

これを `nil` ではない値にしておくと、すべての概略バッファースレッド（の親以外）は、初めは隠されているようになります。

gnus-updated-mode-lines

これを `nil` にすると、Gnus はバッファースレッドのモード行に情報を表示しないので、いくらか時間を節約できるでしょう。

10.6.3 少ないディスク容量

起動ファイルはやや大きくなり得るので、空き容量が少なくなっているときは、そのサイズを少し小さくする必要があります。

gnus-save-newsrc-file

これを `nil` にすると、Gnus は決して `‘.newsrc’` を保存しません—`‘.newsrc.eld’` だけを保存します。これは Gnus 以外のニュースリーダーが使いなくなることを意味します。この変数はデフォルトで `t` です。

gnus-read-newsrc-file

これが `nil` であれば、Gnus は `‘.newsrc’` を決して読みません—`‘.newsrc.eld’` だけを読みます。これは Gnus 以外のニュースリーダーが使いなくなることを意味します。この変数はデフォルトでは `t` です。

gnus-save-killed-list

これが `nil` であると、Gnus は死んだグループのリストを保存しません。この変数を `nil` に設定したときは、`gnus-check-new-newsgroups` を `ask-server` に、`gnus-check-bogus-newsgroups` を `nil` に設定するべきでしょう。この変数はデフォルトで `t` です。

10.6.4 遅いマシン

遅いマシンを持っているか、または本当は単に忍耐力が無いだけでも、Gnus を速く走らせるためにできることが少しあります。

起動を速くするために `gnus-check-new-newsgroups` および `gnus-check-bogus-newsgroups` を `nil` に設定して下さい。概略バッファースレッドに入るものと抜けることを速くするために、`gnus-show-threads` と `gnus-use-cross-reference`、それに `gnus-nov-is-evil` を `nil` に設定して下さい。

10.7 問題解決

Gnus は箱から出してすぐに 非常に よく動作します—どんな問題が起きることも想像できません、本当に。

オッホン。

1. コンピューターの電源が入っていることを確かめて下さい。
2. 現在通用している版の Gnus を本当に読み込んでいることを確認して下さい。今まで GNUS を実行してきたのであれば、Gnus が動作するように一度 Emacs を終了して再起動する必要があります。
3. *M-x gnus-version* を試して下さい。もし ‘Gnus v5.10.6’ のようなものが出てきたなら、正しいファイルが読み込まれています。そうならないのは古い ‘.el’ ファイルが散らかっているせいでしょう。それらを消して下さい。
4. FAQ と入門書を読むために、ヘルプグループ (グループバッファで *G h*) を読んで下さい。
5. Gnus は多くの再帰構造で動作しているので、何か極端な (そして非常に希な) 場合には、Gnus は再帰を「あまりに深く」降りすぎてしまい、Emacs があなたにピープ音を鳴らすことがあります。もしこれが起こったなら、*max-lisp-eval-depth* を 500 かそこいらの値に設定して下さい。

もし他のすべてが失敗したなら、バグとして問題を報告して下さい。

もし Gnus のバグを見つけたなら、*M-x gnus-bug* 命令で報告することができます。*M-x set-variable RET debug-on-error RET t RET* とタイプして、私にバクトレースを送って下さい。私はバグを修正しようと思いますが、あなたがバグを再現させる方法を正確に書いてくれないと、それを修正することができません。

バグ報告では、詳細すぎることは決してありません。バグ報告をするときは、いつも *M-x gnus-bug* 命令を使って下さい。それを使うたびに 10KB のメールができてしまっても、そしてあなたの環境のことを以前私に 500 回送ったことがあったとしてもです。

私がどんなたくい記憶も持っていないことを、覚えておくことも重要です。もしあなたがバグ報告を送ると、私は返答を送ります。その後で、あなたが「いや、そうじゃない! このうすのろめっ!」とだけ送り返してきても、私はあなたが何について私を侮辱しているかがわかりません。常に、すべてを説明し過ぎて下さい。それは私たちすべてにとって、もっとやり易くなります—もし私が必要なすべての情報を得られなかったら、私はあなたにメールを送ってさらなる情報を求め、その結果すべてがより多くの時間を費やすことになります。

もしあなたの直面している問題が非常に視覚的で、それをうまく説明できない場合は、Emacs のウィンドウをファイルにコピーして (例えば、*xwd* で)、それをどこか手の届くどこかにおいて、その画像の URL をバグ報告に含めて下さい。

もしあなたがバグの修正や改善のためのパッチを寄稿して下さるのでしたら、すみませんがそのパッチは ‘*diff -u*’ で作って下さい。

問題を報告する前にもっとデバッグしたければ、あなた自身で問題を解決してパッチを送るために *edebug* を使うことができるでしょう。Lisp コードのデバッグについては *ELisp マニュアル* (see section “Debugging Lisp Programs” in *The GNU Emacs Lisp Reference Manual*) に書かれています。*edebug* を始めるには、もし *c* を押したときにある変な振舞いが発見されるならば、第一歩は *C-h k c* をタイプし、ドキュメンテーション・バッファ中でハイパーリンクをクリック (Emacs のみ) して、その関数定義を参照することです。そしてその関数名の場所で *M-x edebug-defun RET* をタイプして Gnus に戻り、そのコードを呼び出すために *c* を押し下さい。Lisp バッファでは、*SPC* でシングルステップ動作、*M-:* で式を評価、*C-h v* で変数を検査、*q* で実行を中断、あるいは *c* が *g* で実行を再開することができます。

ときどき、直接に `elisp` のエラーを起こさないものの、Gnus が非常に遅くなるために明らかになる問題があります。そんな場合には `M-x toggle-debug-on-quit` を使って、遅くなったときに `C-g` を押し、しかる後にバクトレースを解析して下さい (その手続きを繰り返すことは、真の問題領域を分離するのに役立ちます)。

より上等なやり方は `elisp` プロファイラー (訳注: プログラムの実行時の動きを分析する道具) `ELP` を使うことです。プロファイラーについてはどこか他の場所で完全に文書化されているはずですが、それを始めるために必要な手順を少々書いておきましょう。第一に、プロファイルしてみたい Gnus の部分を計測するための設定を、例えば `M-x elp-instrument-package RET gnus` や `M-x elp-instrument-package RET message` で行なって下さい。そして、遅い動作を行なわせてから `M-x elp-results` を押しましょう。すると、どの動作が時間を食っているかを見て、それらをさらにまたデバッグすることができます。動作全体が、プロファイラーの出力の中で最も遅い関数で費やされた時間よりはるかに長くなるのは、たぶん Gnus の間違っている部分をプロファイルしたせいでしょう。プロファイルの統計をリセットするには `M-x elp-reset-all` を使って下さい。`M-x elp-restore-all` はプロファイルする動作を取り除くことになっていますが、Gnus によって複雑にされかつ動的なコード生成の影響を受けるため、それは必ずしも完全には動作しないかもしれません。

もし手助けが欲しいだけであれば、`'gnu.emacs.gnus'` で尋ねるのが良いでしょう。私はあまり役に立ちません。また、ding メーリングリスト—`ding@gnus.org` で尋ねることもできます。購読するには `ding-request@gnus.org` にメールを送って下さい。

10.8 Gnus リファレンスガイド

誰かが Gnus でできる何か粋なものに知恵を働かせて、その粋なものを書いてもくれることが私の願いです。それを促進するためには、Gnus の内部動作を説明するのが良いだろうと思いました。それに、さほど内部ではない動作をいくつかと、私が今やっていることも。

プログラムの内部の仕様が変更されることはない、などと思っはいいませんが、Gnus とそのバックエンドの間のインターフェース（これは完全に記述されています）や、スコアファイルの形式（同じく）、データ構造（これは他のものほどには変更されないでしょう）、それに一般的な操作のメソッドを、（細部にわたって）定義していきます。

10.8.1 Gnus の有用な関数

フックなどから実行される小さな関数を書くときは、Gnus の内部関数や変数にアクセスすることが絶対に必要です。以下が最もよく使われるものの一覧です。

`gnus-newsgroup-name`

この変数は現在のニュースグループの名前を持っています。

`gnus-find-method-for-group`

group の選択方法を返す関数です。

`gnus-group-real-name`

正規の（接頭語付きの）Gnus グループ名を受け取って、接頭語が無い名前を返します。

`gnus-group-prefixed-name`

接頭語が無いグループ名と選択方法を受け取って、正規の（接頭語付きの）Gnus グループ名を返します。

`gnus-get-info`

group のグループ情報のリストを返します。

`gnus-group-unread`

group の未読記事の数か、それが分からない場合は `t` を返します。

`gnus-active`

group に関するアクティブファイルの項目を返します。

`gnus-set-active`

group に関するアクティブファイルの項目を設定します。

`gnus-add-current-to-buffer-list`

Gnus を終了するときに消去するバッファのリストに、現在のバッファを追加します。

`gnus-continuum-version`

引数として Gnus のバージョン文字列を受け取って、浮動小数点の数値を返します。古いバージョンは必ず新しいバージョンよりも小さい数になります。

`gnus-group-read-only-p`

group が読み出し専用かどうかを示します。

`gnus-news-group-p`

group がニュースバックエンドかどうかを示します。

`gnus-ephemeral-group-p`

group が一時ニュースグループかどうかを示します。

`gnus-server-to-method`

server に対応している選択方法を返します。

`gnus-server-equal`

二つの仮想サーバーが同一かどうかを示します。

`gnus-group-native-p`

group が基本グループかどうかを示します。

`gnus-group-secondary-p`

group が二次グループかどうかを示します。

`gnus-group-foreign-p`

group が外部グループかどうかを示します。

`gnus-group-find-parameter`

group のパラメーターのリストを返します。二つ目の引数を与えると、*group* 用のそのパラメーターの値を返します。

`gnus-group-set-parameter`

三つの引数 *group*, *parameter*, *value* を与えて、パラメーターとして設定します。

`gnus-narrow-to-body`

現在のバッファを、記事の本文に狭めます。

`gnus-check-backend-function`

二つの引数 *function* と *group* を取ります。*group* のバックエンドが *function* をサポートしているなら、`nil` ではない値を返します。

```
(gnus-check-backend-function "request-scan" "nnml:misc")
⇒ t
```

`gnus-read-method`

利用者に選択方法を入力してもらう関数です。

10.8.2 バックエンドインターフェース

Gnus は NNTP やスプール、メール、仮想グループについては何も知りません。ただ仮想サーバー *virtual servers* と対話する方法を知っているだけです。仮想サーバーはバックエンド *back end* といくつかのバックエンド変数 *back end variables* からなります。前者の例としては `nntp`, `nnsPOOL`, `nnmbox` などがあります。後者の例としては `nntp-port-number` や `nnmbox-directory` があります。

Gnus がバックエンド—例えば `nntp`—に何かの情報を要求するとき、通常は関数の引数として仮想サーバー名を含めます。(無い場合は、バックエンドは「現在の」仮想サーバーを使うべきです。) 例えば `nntp-request-list` は、その唯一の(省略可能な)引数として仮想サーバーを使います。もしこの仮想サーバーとの接続が開かれていないと、この関数の実行は失敗するはずです。

仮想サーバー名は、物理的なサーバー名とは何の関係も無いことに注意して下さい。例を挙げましょう:

```
(nntp "odd-one"
  (nntp-address "ifi.uio.no")
  (nntp-port-number 4324))
```

ここで物理サーバー名は `'ifi.uio.no'` であるのに対して、仮想サーバー名は `'odd-one'` です。

バックエンドは複数の仮想サーバーを切り替えることができればなりません。標準のバックエンドは、必要なときに仮想サーバーの環境を取り出し・押し込みを行なう連想リストを保持することによって、これを実現しています。

インターフェース関数には二つのグループがあります。必ず存在しなければならない必須関数 *required functions* と、呼び出す前にそれが存在するかどうかを常に Gnus が確認する任意関数 *optional functions* です。

これらすべての関数は、その戻り値のデータを `nntp-server-buffer` (`*nntpd*`) バッファに返すことが求められます。これはちょっと不運な名前付けですが、これで我慢しなければなりません。私が結果のデータ *resulting data* と言ったときは、そのバッファの中のデータを指しています。戻り値 *return value* と言ったときは、関数呼び出しによって返される関数の値のことを言っています。関数が失敗したときは、戻り値として `nil` を返さなくてはなりません。

バックエンドにはサーバー型 *server-forming* のバックエンドと呼ばれるものがあり、またそう呼ばれないものもあります。後者は一般には、同時には一つのグループだけしか操作しないバックエンドで、「サーバー」の概念がありません。このサーバーとは、グループを持ち、そのグループの情報を配送するもので、それ以上のものではありません。

Gnus はグループ名と記事番号によって、それぞれのメッセージを特定します。それら記事番号に関するちょっとした説明をすることは有益かもしれませんが。まず第一に、その数値は正の整数です。第二に Gnus を混乱させることなく古い記事番号を、後で「再使用」することは普通はできません。すなわち、もしあるグループにかつて 42 番の記事があったとしたら、別の記事がその番号を持つことができないか、または Gnus が激しく混乱してしまうということです。¹ 第三に、記事番号はそのグループでの到着順になっていなければならないことです。メッセージの日付も、必ず到着順になっているわけではありませんが。

すでに前の節で、記事番号は一回使われただけで役目を終わらなければならない「厳しい」制限について説明しました。しかし、記事番号の並びに抜けがあると Gnus はとても混乱してしまうので、連続した 通し 番号を付けることが有用なのかもしれません。ただし「再使用不可」の制限があるので、完全に番号の抜けを回避できるとは限りません。また、可能な限り記事番号を 1 から始めることは、番号を使い切ってしまうことを避けるために役立ちます。

慣例として、バックエンドは `nn` なんたら と名付けられますが、Gnus には `'nnheader.el'`、`'nnmail.el'` および `'nnoo.el'` のように、いくつかのバックエンドではない `nn` かんたら があることに注意して下さい。

ここでの例と定義では、想像上のバックエンド `nnchoke` を引き合いに出すことにします。

10.8.2.1 必須バックエンド関数

(`nnchoke-retrieve-headers` `ARTICLES` &optional `GROUP` `SERVER` `FETCH-OLD`)

`articles` は記事番号の範囲か、Message-ID のリストのどちらかです。現在のバックエンドは、どちらも完全にサポートしているわけではありません—記事番号のひと続き (リスト) だけで、多くのバックエンドは Message-ID による取得をサポートしていません。でも、それらは両方サポートすることに努めるべきです。

結果のデータは HEADs か NOV 行のいずれかであるべきで、戻り値はこれを反映した `headers` か `nov` のどちらかでなければなりません。これは今後、HEADs と NOV 行が混在する `various` に拡張されるかもしれませんが、現在の Gnus ではサポートされていません。

¹ `nnchoke-request-update-info` 関数の説明を見て下さい。See Section 10.8.2.2 [Optional Back End Functions], page 340.

fetch-old が *nil* ではなかったら、ある意味での「余分なヘッダー」を取得しようとし
ます。これは通常、*articles* の中の最小番号の記事よりも小さい番号を持っている (最
大で) *fetch-old* 個の記事と、*articles* の中で欠番になっている記事の、余分なヘッ
ダーを取得します。もしバックエンドがこの要求に従うことを煩わしいと思った場合に
は、このパラメーターの存在は無視されることもあります。この値が *nil* でも数値でも
なかったら、最大限の取得を行いません。

これが HEAD の例です:

```
221 1056 Article retrieved.
Path: ifi.uio.no!sturles
From: sturles@ifi.uio.no (Sturle Sunde)
Newsgroups: ifi.discussion
Subject: Re: Something very droll
Date: 27 Oct 1994 14:02:57 +0100
Organization: Dept. of Informatics, University of Oslo, Norway
Lines: 26
Message-ID: <38o8e1$a0o@holmenkollen.ifi.uio.no>
References: <38jdmq$4qu@visbur.ifi.uio.no>
NNTP-Posting-Host: holmenkollen.ifi.uio.no
.
```

そういうわけで、*headers* という戻り値は、データバッファーにその要素数と同じ個数
のヘッダーがあることを暗示します。

これがそういうバッファーの BNF 定義です:

```
headers      = *head
head         = error / valid-head
error-message = [ "4" / "5" ] 2number " " <error message> eol
valid-head   = valid-message *header "." eol
valid-message = "221 " <number> " Article retrieved." eol
header       = <text> eol
```

(ここで使った BNF の版は RFC822 で使われているものです。)

戻り値が *nov* だった場合は、データバッファーには *network overview database* 行が
含まれていなければなりません。これは基本的には複数の欄をタブで区切ったものです。

```
nov-buffer = *nov-line
nov-line   = field 7*8[ <TAB> field ] eol
field      = <text except TAB>
```

これらの欄に何が含まれるべきかをきちんと調べたいのならば、Section 10.8.4 [Head-
ers], page 349 を参照して下さい。

(nnchoke-open-server SERVER &optional DEFINITIONS)

ここでの *server* は仮想サーバー名です。*definitions* はこの仮想サーバーを定義する
(VARIABLE VALUE) の組のリストです。

サーバーと接続できなかった場合でも、エラーをシグナルして処理を中断してはいけま
せん。バックエンドは、これ以後さらにこのサーバーに接続しようとする試みを、拒否
することを選ぶことができます。実際、そうすべきです。

すでにそのサーバーと接続されていた場合には、この関数は *nil* ではない値を返さなけ
ればなりません。このとき、返される結果のデータはありません。

(nnchoke-close-server &optional SERVER)

server との接続を閉じて、これに関連するすべてのリソースを開放します。もし何らかの理由でサーバーを閉じることができない場合は、`nil` を返します。

返される結果のデータはありません。

(nnchoke-request-close)

すべてのサーバーとの接続を閉じて、バックエンドが保有していたすべてのリソースを開放します。このバックエンドによって作られたすべてのバッファを削除しなければなりません。(もっとも `nntp-server-buffer` は削除されませんが。) 普通この関数は Gnus が終了するときのみ呼び出されます。

返される結果のデータはありません。

(nnchoke-server-opened &optional SERVER)

server が現在の仮想サーバーで、かつその物理サーバーへの接続が生きている場合、この関数は `nil` ではない値を返さなければなりません。どんな状況でも、この関数は接続が失われたサーバーへの再接続を試みてはいけません。

返される結果のデータはありません。

(nnchoke-status-message &optional SERVER)

この関数は *server* からの最後のエラーメッセージを返します。

返される結果のデータはありません。

(nnchoke-request-article ARTICLE &optional GROUP SERVER TO-BUFFER)

この関数の結果のデータは、*article* で指定された記事でなければなりません。Message-ID か番号のいずれかを指定することができます。Message-ID による記事の取得を実装するかどうかは任意ですが、それが可能になっている方が良いでしょう。

to-buffer が `nil` でなかったら、結果のデータは通常のデータバッファの代わりに、このバッファに返さなければなりません。Gnus は主に、記事バッファに直接記事を挿入するように要求しますが、これによって、多量のデータのあるバッファから別のバッファにコピーするのを避けることが可能になります。

もし少しでも可能なら、この関数は `cons` セルを返すべきです。その `car` は取得した記事があるグループ名で、`cdr` は記事の番号です。これによって、Message-ID で記事を取得したときに、Gnus が本当のグループと記事番号を知ることができるようになります。これが不可能な場合は、記事の取得に成功したときに `t` を返さなければなりません。

(nnchoke-request-group GROUP &optional SERVER FAST)

group のデータを取得します。この関数には *group* を現在のグループにするという副作用もあります。

fast が設定されたなら、有用なデータを返す面倒を行わずに、単に *group* を現在のグループにします。

これが結果のデータの例と、定義それ自体です:

```
211 56 1000 1059 ifi.discussion
```

最初の数値は状態で、これは 211 でなくてはなりません。次はそのグループにある記事の総数、最小の記事番号、最大の記事番号、そして最後がグループ名です。しかし、いくつかの記事はキャンセルされているかもしれないので、記事の総数は、記事の最大・最小番号から単純に考えられる数よりも小さいかもしれないことに注意して下さい。Gnus

は総数を単に捨ててしまうので、(それが問題であるときに) 正しい値を生成する面倒を負うべきかどうかは、読者への課題として残してあります。もしそのグループに記事が無かったら、最小記事番号は 1、最大は 0 として報告しなければなりません。

```
group-status = [ error / info ] eol
error        = [ "4" / "5" ] 2<number> " " <Error message>
info         = "211 " 3* [ <number> " " ] <string>
```

(nnchoke-close-group GROUP &optional SERVER)

group を閉じて、それに関連するすべてのリソースを開放します。ほとんどのバックエンドは何もすることが無いでしょう。

返される結果のデータはありません。

(nnchoke-request-list &optional SERVER)

server 上で利用可能なすべてのグループのリストを返します。本当に 全部 という意味です。

これは、たった二つしかグループを持っていないサーバーの場合の例です:

```
ifi.test 0000002200 0000002000 y
ifi.discussion 3324 3300 n
```

各行にはグループ名、そのグループ内の最大の記事番号、最小の記事番号、そして最後にフラグがあります。もしそのグループに記事が無かったら、最小記事番号は 1、最大は 0 として報告しなければなりません。

```
active-file = *active-line
active-line = name " " <number> " " <number> " " flags eol
name        = <string>
flags       = "n" / "y" / "m" / "x" / "j" / "=" name
```

フラグは、そのグループが読み出し専用 ('n') か、司会者付き ('m') なのか、死んでいる ('x') か、どこか他のグループの別名 ('=other-group') なのか、それらのどれでもない ('y') のかを示します。

(nnchoke-request-post &optional SERVER)

この関数は、現在のバッファを投稿しなければなりません。投稿が成功したかどうかを返しても構いませんが、必須ではありません。例えば、投稿が非同期に行なわれる場合は、この関数が終了した時点では、投稿は普通完了していません。その場合この関数は、投稿を完了させることができないときに、それをはっきりと利用者に知らせる見張り (sentinel) のようなものを設定するべきでしょう。

この関数から返される結果のデータはありません。

10.8.2.2 任意バックエンド関数

(nnchoke-retrieve-groups GROUPS &optional SERVER)

groups はグループのリストです。また、この関数はそれら全部のグループのデータを要求しなければなりません。どうやってそれを行なうかは Gnus の知ったことではありませんが、これをできるだけ迅速な方法で行なうことに挑まなければなりません。

この関数の戻り値は *active* か *group* のどちらでも良く、それが結果のデータの形式が何であるかを示します。前者は *nnchoke-request-list* によるデータと同じ形式です。一方後者は *nnchoke-request-group* が返すものと同じ形式の、バッファを埋める行です。

```
group-buffer = *active-line / *group-status
```

(nnchoke-request-update-info GROUP INFO &optional SERVER)

Gnus のグループ情報 (see Section 10.8.6 [Group Info], page 350) が、バックエンドのそれを改変するために渡されます。これはバックエンドが (仮想グループや imap グループの場合のように)、本当にすべての情報を持っている場合に役に立ちます。この関数は、その要求に適合させる情報を破壊的に置き換えて、nil ではない値を返さなければなりません (例外的に nntp-request-update-info は、ネットワーク資源を浪費しないように常に nil を返します)。

この関数が返す結果のデータはありません。

(nnchoke-request-type GROUP &optional ARTICLE)

利用者が「ニュースを送信する」命令 (例えば、概略バッファで F) を実行したときに、Gnus は利用者がフォローアップしようとしている記事がニュースなのかメールなのかを知っている必要があります。この関数は *group* の中の *article* がニュースであれば *news* を、メールであれば *mail* を、その種別を判定できない場合は *unknown* を返さなければなりません。(*article* 引数は、メールグループとニュースグループがごちゃまぜになっているかもしれない nnvirtual において必要です。) *group* と *article* は両方とも nil であるかもしれません。

この関数が返す結果のデータはありません。

(nnchoke-request-set-mark GROUP ACTION &optional SERVER)

記事の印を設定/消去/追加します。通常 Gnus は記事の印 (既読、可視、期限切れ消去など) を内部で扱い、`~/newsrsrc.eld` に保存します。しかし、いくつかのサーバー (例えば IMAP) は記事のすべての情報をサーバーで持っているので、Gnus が印の情報をサーバーに伝搬させる必要があります。

action は印を設定する要求のリストで、以下の様式を持ちます:

```
(RANGE ACTION MARK)
```

range は印を付けたい記事の範囲です。*action* は *add* または *del* で、印を追加したり消すために使われます (言及されていないすべての印は保存します)。*mark* は印のリストです。それぞれの印はシンボルです。現在使われている印は *read*, *tick*, *reply*, *expire*, *killed*, *dormant*, *save*, *download*, *unsubscribe*, *forward* および *recent* ですが、あなたのバックエンドは、可能ならこれらを制限をするべきではありません。

矛盾する動作が指定された場合は、リストの最後の動作が効力を持つものになるべきです。すなわち、*action* が記事 1 に 可視 印を追加する要求を含んでいて、リストのおしまいの方で、同じ記事から印を消去することを要求していたならば、印は実際には消去されるべきです。

action リストの例です:

```
(( (5 12 30) 'del '(tick))
  ((10 . 90) 'add '(read expire))
  ((92 94) 'del '(read)))
```

関数は印を設定できなかった記事の範囲を返さなければなりません (現在はどんな目的のためにも使われていません)。

この関数が返す結果のデータはありません。

(nnchoke-request-update-mark GROUP ARTICLE MARK)

バックエンドが嫌う印を利用者が設定しようとしたら、この関数がその印を変更することができます。この関数が返したどんなものでも、Gnus は *article* への印として元の

mark の代わりに使います。バックエンドがそれでも構わない場合には、元の *mark* を返さなければなりません。nil やその他のゴミを返してはいけません。

私が知っているこれのこの利用法は、nnvirtual が行なっていることだけです—その仮想グループで既読の印を付けると、もし構成要素のグループが自動期限切れ消去可能ならば、結果としてその記事に期限切れ消去の印が付けられます。

この関数が返す結果のデータはありません。

(nnchoke-request-scan &optional GROUP SERVER)

バックエンドが新着記事を検査する要求を (Gnus か他の何かによって) 行なうときはいつでも、あれやこれやとこの関数が呼び出されるでしょう。この関数が起動されると、一般にメールバックエンドはスプールファイルを読むか POP サーバーに問い合わせます。group に留意する必要はありません—バックエンドが、一つのグループだけを走査するのが大変すぎると判断した場合には、すべてのグループを総がかりで走査しても構いません。ですが、その方が実用的ならば、局所に限定するのが良いでしょう。

この関数が返す結果のデータはありません。

(nnchoke-request-group-description GROUP &optional SERVER)

この関数が返す結果のデータは、group の説明でなければなりません。

```
description-line = name <TAB> description eol
name              = <string>
description       = <text>
```

(nnchoke-request-list-newsgroups &optional SERVER)

この関数が返す結果のデータは、サーバー上で利用できるすべてのグループの説明でなければなりません。

```
description-buffer = *description-line
```

(nnchoke-request-newgroups DATE &optional SERVER)

この関数から返される結果のデータは、‘date’ 以降に作成されたすべてのグループでなければなりません。‘date’ は人間が読める普通の日付の形式 (すなわち、メールやニュースのヘッダーで使われる形式で、デフォルトは関数 message-make-date が返すもの) です。データは active バッファの形式でなければなりません。

この関数が「多すぎる」グループを返すのはオケです。いくつかのバックエンドでは、新しいグループだけではなく、すべてのグループのリストを返す方が安上がりに済むことを見出すかもしれません。しかし、たくさんのグループがあるバックエンドで、これをしてはいけません。普通、利用者が自分で作ったグループならば多すぎることはないでしょうから、nnml とそれに類するものはたぶん心配ありません。しかし nntp のようなバックエンドでは、グループはサーバーによって作られているので、いかにもたくさんのグループがありそうです。

(nnchoke-request-create-group GROUP &optional SERVER)

この関数は group という名前の空のグループを作成しなければなりません。

返されるデータはありません。

(nnchoke-request-expire-articles ARTICLES &optional GROUP SERVER FORCE)

この関数は、articles の範囲のすべての記事に対して期限切れ消去の処理を行ないます (現在 articles は記事番号の単純なリストです)。記事がどれだけ古いかを、この関数で消去される前に判定することは、バックエンドに任されています。force が nil ではな

い値だったら、それらがどんなに新しくても、すべての *articles* を消去しなければなりません。

この関数は削除しなかった、あるいは削除することができなかった記事のリストを返さなければなりません。

返される結果のデータはありません。

(nnchoke-request-move-article ARTICLE GROUP SERVER ACCEPT-FORM &optional LAST)

この関数は *group* にある記事 *article* (番号) を、*accept-form* を呼び出すことによって移動しなければなりません。

この関数は、当の記事を移動させるための準備として、それが記事に付加したどんなヘッダー行をも削除して、記事を大体において「きれい」にしておく必要があります。そして「きれい」な記事があるバッファで、*accept-form* を eval しなければなりません。これは実際に複製を行いません。もしこの eval が nil 以外の値を返したら、その記事を削除しなければなりません。

もし *last* が nil だったら、それはこの直後にさらに要求が発行される見込みが高いことを意味し、これによっていくらか最適化ができるようになります (訳注: 例えば nil だったらサーバーとの接続を閉じないでおくとか)。

この関数は、移動先のグループ名が *car* で、移動先の記事番号が *cdr* である cons セルを返さなければなりません。

返されるデータはありません。

訳注: 移動先のグループは *accept-form* の中で指定します。そこで使われるのが、次の nnchoke-request-accept-article です。

(nnchoke-request-accept-article GROUP &optional SERVER LAST)

この関数は、現在のバッファの中身を *group* に挿入します。*last* が nil だったら、この関数へのさらなる呼び出しが直ちに行なわれるだろうという意味です。

この関数はグループ名が *car* で、移動先の記事番号が *cdr* である cons セルを返さなければなりません。

そのグループは、記事を受け入れてもらうことをバックエンドが要求する前に存在しなければなりません。

返されるデータはありません。

(nnchoke-request-replace-article ARTICLE GROUP BUFFER)

この関数は *group* から記事 *article* (番号) を削除して、代わりに *buffer* の中身をそこに挿入しなければなりません。

返されるデータはありません。

(nnchoke-request-delete-group GROUP FORCE &optional SERVER)

この関数は *group* を消去しなければなりません。もし *force* が設定されていたら、そのグループ内のすべての記事を本当に消去して、そしてそのグループ自身を消去しなければなりません。(もし「グループ自身」というものがあれば。)

返されるデータはありません。

(nnchoke-request-rename-group GROUP NEW-NAME &optional SERVER)

この関数はグループ名を *group* から *new-name* に変更しなければなりません。*group* 内にあるすべての記事は、*new-name* に移動しなければなりません。

返されるデータはありません。

10.8.2.3 エラーメッセージの発行

バックエンドはエラーの状況の報告に `nnheader-report` を使わなければなりません—要求を実行できないときにエラーを生起させてはいけません。この関数の最初の引数はバックエンド名のシンボルで、残りは、複数の引数があれば `format` への引数として解釈され、一つであればただの文字列です。この関数は常に `nil` を返さなければなりません。

```
(nnheader-report 'nnchoke "You did something totally bogus")
```

```
(nnheader-report 'nnchoke "Could not request group %s" group)
```

一方 Gnus は、サーバーから `nil` を返されたときに `nnheader-get-report` を呼び出します。するとこの関数が、当のバックエンドに対して最後に報告されたメッセージを返します。この関数は一つの引数—サーバーのシンボルを取ります。

内部的には、これらの関数は `back-end-status-string` にアクセスします、したがって `nnchoke` バックエンドはそのエラーメッセージを `nnchoke-status-string` に格納します。

10.8.2.4 新しいバックエンドを書く

多くのバックエンドはよく似通っています。`nnml` は `nnspool` と瓜二つですが、サーバー上の記事を編集することができます。`nnmh` はまるで `nnml` のようですが、アクティブファイルを使わないし、概要データベースも保持しません。`nndir` は `nnml` にとても似ていますが、これには「グループ」の概念は無く、記事の編集はできません。

新しいバックエンドを書くときに他のバックエンドから関数を「継承」できたらなあ、と思うのは理に合っています。そしてまさに、あなたがそうしたければ、それができるのです。(あなたがそうしなくてはならなくても良いですよ、もちろん。)

すべてのバックエンドは、公共変数と公共関数を `nnoo` というパッケージを使って宣言します。

他のバックエンドから関数を継承するには (そして現在のバックエンドから他のバックエンドに関数を継承できるようにするには)、以下のマクロを使用しなければなりません:

`nnoo-declare`

このマクロは、最初の引数を、その後に続く引数の子供であることを宣言します。例えば:

```
(nnoo-declare nndir
  nnml nnmh)
```

ここで `nndir` は、`nnml` と `nnmh` の両方から関数を継承するつもりであることを宣言しています。

`defvoo`

このマクロは `defvar` と等価ですが、その変数を公共サーバー変数として登録します。ほとんどの状態志向型の変数は、`defvar` ではなく `defvoo` によって宣言するべきです。通常の `defvar` の引数に加えて、このマクロは親バックエンドにおける変数のリストを取ります。それらの親バックエンドで定義されている関数を子のバックエンドで実行するときに、その関数の中でアクセスされる親の変数を、子の変数で置き換えます。

```
(defvoo nndir-directory nil
  "Where nndir will look for groups."
  nnml-current-directory nnmh-current-directory)
```

これは `nndir` のために `nnml` の関数が呼び出されたときに、`nnml-current-directory` は `nndir-directory` に設定されるという意味です。(`nnmh` も同様です。)

nnoo-define-basics

このマクロは、ほとんどすべてのバックエンドが持つべき共通関数をいくつか定義します。

```
(nnoo-define-basics nndir)
```

deffoo

このマクロはまさに defun のようなもので、同一の引数を取ります。通常の defun の処理に加えて、このマクロは他のバックエンドがそれを継承できるように、その関数が公共物になっているものとして登録します。

nnoo-map-functions

このマクロは、現在のバックエンドの関数から親バックエンドの関数への、置き換えができるようにします。

```
(nnoo-map-functions nndir
  (nnml-retrieve-headers 0 nndir-current-group 0 0)
  (nnmh-request-article 0 nndir-current-group 0 0))
```

これは nndir-retrieve-headers が呼び出されたときに、一番目、三番目、および四番目の引数が nnml-retrieve-headers に渡され、一方、二番目の引数は nndir-current-group の値として設定されるという意味です。

nnoo-import

このマクロは他のバックエンドから関数を輸入します。これは単にまだ定義されていない関数を定義するだけなので、ソースファイルの最後に書かなければなりません。

```
(nnoo-import nndir
  (nnmh
    nnmh-request-list
    nnmh-request-newgroups)
  (nnml))
```

これは、nndir-request-list への呼び出しは単に nnmh-request-list に引き渡されなければならない、一方 nnml の公共関数でまだ nndir で定義されていないものをここで定義するということです。

以下は nndir バックエンドのちょっと短縮した版です。

```
;;; nndir.el — 単一のディレクトリーをニュースグループにする
;;; Copyright (C) 1995,96 Free Software Foundation, Inc.
```

```
;;; Code:
```

```
(require 'nnheader)
(require 'nnmh)
(require 'nnml)
(require 'nnoo)
(eval-when-compile (require 'cl))
```

```
(nnoo-declare nndir
  nnml nnmh)
```

```
(defvoo nndir-directory nil
  "nndir がグループを探す場所。"
  nnml-current-directory nnmh-current-directory)
```

```

(defvoo nndir-nov-is-evil nil
  "*これが nil でなかったら NOV ヘッダーを取得しません。"
  nnml-nov-is-evil)

(defvoo nndir-current-group ""
  nil
  nnml-current-group nnmh-current-group)
(defvoo nndir-top-directory nil nil nnml-directory nnmh-directory)
(defvoo nndir-get-new-mail nil nil nnml-get-new-mail nnmh-get-new-mail)

(defvoo nndir-status-string "" nil nnmh-status-string)
(defconst nndir-version "nndir 1.0")

;;; インターフェイス用の関数。

(nnoo-define-basics nndir)

(deffoo nndir-open-server (server &optional defs)
  (setq nndir-directory
    (or (cadr (assq 'nndir-directory defs))
        server))
  (unless (assq 'nndir-directory defs)
    (push '(nndir-directory ,server) defs))
  (push '(nndir-current-group
    ,(file-name-nondirectory
      (directory-file-name nndir-directory)))
    defs)
  (push '(nndir-top-directory
    ,(file-name-directory (directory-file-name nndir-directory)))
    defs)
  (nnoo-change-server 'nndir server defs))

(nnoo-map-functions nndir
  (nnml-retrieve-headers 0 nndir-current-group 0 0)
  (nnmh-request-article 0 nndir-current-group 0 0)
  (nnmh-request-group nndir-current-group 0 0)
  (nnmh-close-group nndir-current-group 0))

(nnoo-import nndir
  (nnmh
    nnmh-status-message
    nnmh-request-list
    nnmh-request-newgroups))

(provide 'nndir)

```

10.8.2.5 新しいバックエンドを Gnus に繋げる

あなたの新しいバックエンドを Gnus で使い始めるのはとても簡単です—単に `gnus-declare-backend` 関数で宣言するだけです。これはバックエンドを `gnus-valid-select-methods` 変数に追加します。

`gnus-declare-backend` は二つの引数を取ります—バックエンドの名前と任意の数の能力 *abilities* です。

これが例です。

```
(gnus-declare-backend "nnchoke" 'mail 'respool 'address)
```

そして上記の行が `'nnchoke.el` ファイルに入ります。

能力には以下のものがあります:

<code>mail</code>	これはメール風バックエンドです—フォローアップは (たいていは) メールで送られるはずです。
<code>post</code>	これはニュース風バックエンドです—フォローアップは (たいていは) ニュースで送られるはずです。
<code>post-mail</code>	このバックエンドはメールとニュースの両方をサポートします。
<code>none</code>	これはニュースでもメールでもないバックエンドです—まったく違った何かです。
<code>respool</code>	これは再スプールをサポートします—というか、その元の記事とグループを書き換えることができます。
<code>address</code>	サーバーの名前が仮想サーバー名に含まれていなければなりません。これはほとんど全部のバックエンドに当てはまります。
<code>prompt-address</code>	グループバッファで <code>B</code> などの命令を実行したときに、利用者はアドレスの入力を求められるはずで、例えばこれは <code>nntp</code> のようなバックエンドに当てはまりますが、 <code>nnmbox</code> はそうではありません。

10.8.2.6 メール風バックエンド

メールバックエンドがその他のバックエンドに対して一線を画しているのは、ほとんどのメールバックエンドが `'nnmail.el` で定義されている共通の関数に強く依存しているという点です。例えばこれは `nnml-request-scan` の定義です:

```
(deffoo nnml-request-scan (&optional group server)
  (setq nnml-article-file-alist nil)
  (nnmail-get-new-mail 'nnml 'nnml-save-nov nnml-directory group))
```

単に `nnmail-get-new-mail` にいくつか引数を与えて呼び出すだけで、`nnmail` がメールの移動や分割のすべての面倒を見してくれます。

この関数は四つの引数を取ります。

method これは、どのバックエンドがこの呼び出しの責任を負うかを指示するシンボルでなければなりません。

exit-function

この関数は、分割が実行された後で呼び出されるものでなければなりません。

temp-directory

一時ファイルを格納する場所です。

group

この引数は省略可能です。分割が一つのグループだけに対して行なわれる場合は、この引数でグループ名を指定しなければなりません。

`nnmail-get-new-mail` は、それぞれの記事を保存するために `back-end-save-mail` を呼び出します。`back-end-active-number` は、この記事に割り当てられた記事番号を調べるために呼び出されます。

この関数は次の変数も使用します: `back-end-get-new-mail` (このバックエンドの新着メールを受け取るかどうか)、新しいアクティブファイルを生成するための `back-end-group-alist` および `back-end-active-file` です。`back-end-group-alist` は、以下のようなグループとアクティブの連想リストです:

```
((("a-group" (1 . 10))
  ("some-group" (34 . 39)))
```

10.8.3 スコアファイルの構文

スコアファイルは簡単に分析できるだけでなく、極めて柔軟な対応ができるようになっています。それには Emacs Lisp のリストとして読み込むことができるような構文がふさわしいだろうと判断されました。

これは良くあるスコアファイルです:

```
((("summary"
  ("win95" -10000 nil s)
  ("Gnus"))
  ("from"
    ("Lars" -1000))
  (mark -100))
```

スコアファイルの BNF 定義です。

```
score-file      = "" / "(" *element ")"
element         = rule / atom
rule            = string-rule / number-rule / date-rule
string-rule     = "(" quote string-header quote space *string-match ")"
number-rule     = "(" quote number-header quote space *number-match ")"
date-rule       = "(" quote date-header quote space *date-match ")"
quote           = <ascii 34>
string-header   = "subject" / "from" / "references" / "message-id" /
                  "xref" / "body" / "head" / "all" / "followup"
number-header   = "lines" / "chars"
date-header     = "date"
string-match    = "(" quote <string> quote [ "" / [ space score [ "" /
                  space date [ "" / [ space string-match-t ] ] ] ] ] ")"
score           = "nil" / <integer>
date            = "nil" / <natural number>
string-match-t  = "nil" / "s" / "substring" / "S" / "Substring" /
                  "r" / "regex" / "R" / "Regex" /
                  "e" / "exact" / "E" / "Exact" /
```

```

        "f" / "fuzzy" / "F" / "Fuzzy"
number-match    = "(" <integer> [ " " / [ space score [ " " /
        space date [ " " / [ space number-match-t ] ] ] ] ")"
number-match-t  = "nil" / "=" / "<" / ">" / ">=" / "<="
date-match      = "(" quote <string> quote [ " " / [ space score [ " " /
        space date [ " " / [ space date-match-t ] ] ] ] ")"
date-match-t    = "nil" / "at" / "before" / "after"
atom            = "(" [ required-atom / optional-atom ] ")"
required-atom    = mark / expunge / mark-and-expunge / files /
        exclude-files / read-only / touched
optional-atom    = adapt / local / eval
mark            = "mark" space nil-or-number
nil-or-number    = "nil" / <integer>
expunge          = "expunge" space nil-or-number
mark-and-expunge = "mark-and-expunge" space nil-or-number
files            = "files" *[ space <string> ]
exclude-files    = "exclude-files" *[ space <string> ]
read-only        = "read-only" [ space "nil" / space "t" ]
adapt            = "adapt" [ space "ignore" / space "t" / space adapt-rule ]
adapt-rule       = "(" *[ <string> *[ "(" <string> <integer> ")" ] ] ")"
local           = "local" *[ space "(" <string> space <form> ")" ]
eval             = "eval" space <form>
space            = *[ " " / <TAB> / <NEWLINE> ]

```

認識不可能なスコアファイルの要素は無視されるべきですが、捨ててしまっても構いません。

ご覧のように空白が必要ですが、空白の量と型は重要ではありません。つまり、スコアファイルの様式はプログラマーに任されています—すべてを一つの長ーい行に吐き出す方がより簡単なのであれば、それでも構いません。

いろいろなアトムの意味は、このマニュアルのどこかで説明されています (see Section 7.4 [Score File Format], page 232)。

10.8.4 ヘッダー

Gnus は記事のヘッダーを溜めておくために、内部的には NOV の規格を怪しげなやり方で踏襲する様式を使っています。NOV の仕様を見た作者が、恥知らずにもすべてを盗んだ と思うかもしれませんが、それは正しいです。

「ヘッダー」はひどく荷の重い用語です。「ヘッダー」は RFC1036 では記事の頭の行 (例えば、From) について話すのに用いられています。それは多くの人が「ヘッド」—「ヘッダーと本文」の同義語として使っています。(私に言わせれば、これは避けるべきです。) そして Gnus は、私がここで話す「ヘッダー」と言う様式を内部的に使っています。これは基本的には九つの要素からなるベクトルで、それぞれのヘッダー (あ痛っ) が一つの場所を占めます。

これらの場所は、順番に number, subject, from, date, id, chars, lines, xref, および extra です。これらの場所を読み出したり設定するマクロがあります—それらはすべて、それぞれ mail-header- と mail-header-set- という予想しやすい名前を持っています。

extra のための場所がヘッダーと値の対の連想リストであることを除いて、これらすべての場所には文字列が入ります (see Section 3.1.2 [To From Newsgroups], page 46)。

10.8.5 範囲

GNUS は非常に有用な概念を導入してくれました。私はそれをたくさん使い、かなり入念に仕上げました。

設問は単純です：何か番号で呼ぶことができる大量のもの（粗雑な 例としては、例えば記事）を持っていて、それらが「含まれている」ことを表現したいとしましょう。それらを順番に並べるのは、あまり便利ではありません。（20,000 個を順番に並べたものは、ちょっと長たらしいですね。）

解決策は設問と同じくらい単純です。単にその並びを折りたためれば良いのです。

```
(1 2 3 4 5 6 10 11 12)
```

は次のように変形されます。

```
((1 . 6) (10 . 12))
```

単独のものを表すために ‘(13 . 13)’ のようなやっかいな要素を持たなくても良いように、‘13’ は有効な要素になっています。例えば：

```
((1 . 6) 7 (10 . 12))
```

以下のような二つの範囲を比較して、それらが等しいがどうかを調べるのは、少し手のこんだことになります：

```
((1 . 5) 7 8 (10 . 12))
```

と

```
((1 . 5) (7 . 8) (10 . 12))
```

は等しいです。実際のところ、下り順で並んでいないリストは範囲です：

```
(1 2 3 4 5)
```

これはかなり長たらしいものですが、完璧に有効な範囲です。以下も有効です：

```
(1 . 5)
```

そして、これはその前の範囲と等しいものです。

これは範囲の BNF 定義です。もちろん、数値の並びが下り順であってはならないことを覚えておかねばなりません。（同じ数値を任意の回数にわたって繰り返すことができますが、範囲の扱いにおいて消え去る傾向があります。）

```
range          = simple-range / normal-range
simple-range    = "(" number " . " number ")"
normal-range   = "(" start-contents ")"
contents       = "/" simple-range *[" " contents ] /
               number *[" " contents ]
```

現在 Gnus は既読記事と記事の印を維持するために範囲を使っています。当局が私にそれをさせたがっているのなら、私は数の範囲の操作を C で実装しようと思っています。（私はまだ尋ねていません。と言うのは、普通の連続体に変換し直さずに、世の中を完全に範囲に基づいたものにするためには何が必要かを、私はもっと考えなければならないからです。）

10.8.6 グループ情報

Gnus はグループのすべての永続的な情報を *group info* リストに格納します。このリストは 3 から 6（またはそれ以上）の長さの要素で、徹底的にグループを記述します。

ここにあるのはグループ情報 (group info) の二つの例です。一つは非常に単純なグループですが、二つ目はもっと複雑なものです：


```

("no.group" 5 ((1 . 54324)))

("nnml:my.mail" 3 ((1 . 5) 9 (20 . 55))
  ((tick (15 . 19)) (replied 3 6 (19 . 3)))
  (nnml ""))
  ((auto-expire . t) (to-address . "ding@gnus.org")))

```

最初の要素は「グループ名」——とにかく Gnus が知っているグループです。二番目の要素は「購読度」で、普通は小さな整数です。(それは「階級」(rank) になることもできます。car がレベルで cdr がスコアのコンスセルです。) 三番目の要素は既読記事の範囲のリストです。四番目の要素はいろいろな種類の記事の印のリストのリストです。五番目の要素は選択方法です (もしくは、そう言いたければ仮想サーバーです)。六番目の要素は「グループパラメーター」のリストで、この章はそのためにあります (訳注: ほんとうに?)。

最後の三つの要素はどれでも、必要が無ければ存在しないこともあります。実際、グループの非常に大部分は最初の三つの要素だけを持ち、それは (最後の三要素が省略できることは) 非常に多くのコンスセルを節約します。

これはグループ情報様式の BNF 定義です:

```

info          = "(" group space ralevel space read
               [ "" / [ space marks-list [ "" / [ space method [ "" /
               space parameters ] ] ] ] ] ")"
group         = quote <string> quote
ralevel       = rank / level
level         = <integer in the range of 1 to inf>
rank          = "(" level "." score ")"
score         = <integer in the range of 1 to inf>
read          = range
marks-lists   = nil / "(" *marks ")"
marks         = "(" <string> range ")"
method        = "(" <string> *elisp-forms ")"
parameters    = "(" *elisp-forms ")"

```

実は 'marks' の規則はごまかしです。'marks' は 'range' とともに cons を構成する '<string>' ですが、疑似 BNF でそれを現すのは難しいのです。

情報の要素群にアクセスして、それらの値を取得または設定するために、Gnus は一連のマクロを提供しています。

```

gnus-info-group
gnus-info-set-group
      グループ名を取得/設定 (get/set) します。

```

```

gnus-info-rank
gnus-info-set-rank
      グループの階級 (rank) を取得/設定します (see Section 2.7 [Group Score], page 20)。

```

```

gnus-info-level
gnus-info-set-level
      グループのレベルを取得/設定します。

```

```
gnus-info-score
gnus-info-set-score
    グループのスコアを取得/設定します (see Section 2.7 [Group Score], page 20).
```

```
gnus-info-read
gnus-info-set-read
    既読記事の範囲を取得/設定します。
```

```
gnus-info-marks
gnus-info-set-marks
    印が付いている記事の範囲のリストを取得/設定します。
```

```
gnus-info-method
gnus-info-set-method
    グループの選択方法を取得/設定します。
```

```
gnus-info-params
gnus-info-set-params
    グループパラメーターを取得/設定します。
```

取得するためのすべての関数は一つの引数を取ります—情報のリストです。設定するための関数は二つの引数を取ります—情報のリストと新しい値です。

グループ情報の最後の三つの要素は必須ではないので、要素を設定する前にグループ情報を拡張する必要があるでしょう。それが必要な場合、最後の三つの設定するための関数の第三引数に `nil` ではない値を指定すれば、自動的に拡張させることができます。(訳注: 例えば三つの要素しかない情報に四つ目の要素を加える処理を第三引数を使わずに行なうと、`(setcar (nthcdr 3 INFO) VALUE)` というコードが実行される結果、エラーになってしまいます。)

10.8.7 対話形式の拡張

Gnus は Emacs 標準の `interactive` の仕様を、シンボル接頭引数を簡単に使うことができるようにするために、少し拡張しています (see Section 8.3 [Symbolic Prefixes], page 250)。これはその使い方の例です:

```
(defun gnus-summary-increase-score (&optional score symp)
  (interactive (gnus-interactive "P\ny"))
  ...
)
```

最上のものは `interactive` の式を返すマクロとして `gnus-interactive` を実装することでしょうが、Emacs は関数が対話的かどうかを調べるために、ラムダ式に対して単純に `assq` を行なうので、これは不可能です。そこで、文字列を受け取って `interactive` で使うことができる値を返す `gnus-interactive` 関数を、代わりに持つことにしました。

この関数は (ほとんど) すべての `interactive` の指定を受け付けますが、もう少し加えることにします。

‘y’ 現在のシンボル接頭引数—変数 `gnus-current-prefix-symbol` です。

‘Y’ 現在のシンボル接頭引数のリスト—変数 `gnus-current-prefix-symbol` です。

‘A’ 現在の記事番号—関数 `gnus-summary-article-number` です。

‘H’ 現在の記事ヘッダー —関数 `gnus-summary-article-header` です。

‘g’ 現在のグループ名—関数 `gnus-group-group-name` です。

10.8.8 Emacs/XEmacs コード

Gnus は Emacs, XEmacs と Mule で動作するので、そのうちの一つを主環境とすることに決めました。私は Emacs を選びました。私が XEmacs や Mule を好きではないということではなく、それがアルファベット順で最初に来たからです。(訳注: 現在 Gnus がサポートしている (X)Emacs については Section 10.2.6 [Emacsen], page 304 を参照して下さい。)

これは、Gnus は Emacs で少しの警告も無くバイトコンパイルできるのに対して、XEmacs はバイトコンパイルをしている間にギガバイトくらいの警告を出すということでもあります(訳注: 現在はそんなことはありません)。私は些細な失敗を見つけ出すことに役立てるためにバイトコンパイルの警告を使っているので、それはとても助けになります。

さらに、私は首尾一貫して Emacs の関数のインターフェースを使ってきましたが、それらの関数のために Gnus の別名 (aliases) を使ってきました。例を出しましょう: Emacs が関数 `run-at-time` を定義している一方で、XEmacs は関数 `start-itimer` を定義しています。そこで、私は Emacs の `run-at-time` と同じ引数を受け取る `gnus-run-at-time` という関数を定義しました(訳注: 現在は `run-at-time` に統一されています)。Gnus を Emacs で実行しているときは、`gnus-run-at-time` は単に `run-at-time` の別名になっています。しかし XEmacs で実行したときは、`gnus-run-at-time` は次の関数の別名となっています(訳注: 現在こういうものはありません):

```
(defun gnus-xmas-run-at-time (time repeat function &rest args)
  (start-itimer
   "gnus-run-at-time"
   '(lambda ()
       (,function ,@args))
   time repeat))
```

この種のことが多くの関数のために行なわれています。Gnus は XEmacs で実行しているときに、元からある Emacs の関数を再定義しません—代わりにそれは、Gnus の等価なものに `defalias` をすることによって行ないます。その方が、よりきれいです。

XEmacs の関数のインターフェースの方が明らかにきれいな場合は、私は代わりにそれを使います。例えば、`gnus-region-active-p` は XEmacs では `region-active-p` の別名であるのに対して、Emacs では関数です。

もちろん XEmacs を私の基本プラットフォームに選んで、関数の割り当てを逆にすることもできましたが、私はそうしませんでした。XEmacs で Gnus を実行するときの、これらの遠回しな割り当てが強い性能への打撃は、僅かなはずです。

10.8.9 いろいろなファイル様式

10.8.9.1 アクティブファイルの様式

アクティブファイルは、対象になっているサーバーのすべての使用可能なグループの目録を保持します。そこには、それぞれのグループの最高と最低の記事番号の目録もあります。

これは普通のアクティブファイルからの抜粋です:

```
soc.motss 296030 293865 y
alt.binaries.pictures.fractals 3922 3913 n
comp.sources.unix 1605 1593 m
comp.binaries.ibm.pc 5097 5089 y
no.general 1000 900 y
```

これはこのファイルの疑似 BNF 定義です:

```
active      = *group-line
group-line  = group spc high-number spc low-number spc flag <NEWLINE>
group       = <non-white-space string>
spc         = " "
high-number = <non-negative integer>
low-number  = <positive integer>
flag        = "y" / "n" / "m" / "j" / "x" / "=" group
```

このファイルの完全な説明は、‘innd’のマニュアルページ、特に‘active(5)’を見て下さい。

10.8.9.2 ニュースグループファイルの様式

ニュースグループファイルは、グループの目録をそれらの説明とともに保持します。サーバーにあるすべてのグループがある必要は無いし、そのファイルにあるすべてのグループがサーバーに存在しなくても構いません。このファイルは純粋に利用者の情報のためにあります。

様式はとても単純です: グループ名、タブ、そして説明です。これが定義です:

```
newsgroups  = *line
line        = group tab description <NEWLINE>
group       = <non-white-space string>
tab         = <TAB>
description = <string>
```

10.9 異教徒への Emacs

信じるかどうかはともかく、Gnus Love Boat の旅に搭乗する前にあまり Emacs を使ったことが無いという Gnus の利用者たちがいます。“C-M-a”や「リージョンを kill する」、それに「gnus-flargblossen を連想リストに設定して下さい。そのキーはグループ名に合致するために使われる正規表現です。」といったことが、あなたにとって少しかまったく意味の無い魔法の言葉ならば、この付録はあなたのためにあります。もしあなたがすでに Emacs に親しんでいるのであれば、これを無視してあなたの猫を可愛がりに行って下さい。

10.9.1 打鍵

- Q: 経験のある Emacs の利用者とは何ですか?
- A: 端末にペダルがあったらなぁと願う人のことです。

はい、Emacs を使うとコントロールキー、シフトキー、メタキーをたくさん使うようになるでしょう。これは一部の人々（主に vi 利用者）には非常に煩わしいものですが、私たちはその地獄を愛します。諦めてそれを甘受して下さい。Emacs は本当は“Escape-Meta-Alt-Control-Shift”の略で、あなたがいがわしい (Emacs の作者のような) 出どころから聞いているかもしれない“Editing Macros”ではありません。

シフトキーは普通は両手の小指の近くにあって、普通は大文字などを打つために使われています。あなたは絶え間なくそれを使いますよね。コントロールキーには普通“CTRL”のような印が付いています。メタキーは奇妙なことにどのキーボードでもそういう印が付いていません。それは普通はキーボードの左手側にあって、最下段にあるのが一般的です。

さて、私たち Emacs 人は、それがひどく不便なので「meta-control-m キーを押す」とは言いません。私たちが使うのは「C-M-m を押す」です。M- は「メタ」を現す接頭語で、“C-”は「コントロール」を意味する接頭語です。ですから「C-k を押す」は、「コントロールキーを押しながら次に k を押す」ということです。「C-M-k を押す」は「メタキーとコントロールキーを押しながら次に k を押す」ということです。簡単です、よね?

このことは、すべてのキーボードがメタキーを持っているわけではないという事実によりって、多少ややこしくなっています。そういう場合には「エスケープ」キーを使えばよいでしょう。ただしメタキーを持っているときより作業が増えるので、メタキーのあるキーボードを手に入れていただくことを謹んでお勧め申し上げます。それ無しでは生きて行けないでしょう。

10.9.2 Emacs Lisp

Emacs はエディターの王様です。なぜなら、それが真の Lisp インタープリターだからです。あなたが叩くすべてのキーは、何らかの Emacs Lisp コードの小片を実行します。Emacs Lisp はインタープリターで実行される言語なので、どのキーが何のコードを実行するかを任意に設定することができます。あなたは、ただそうすれば良いのです。

Gnus は Emacs Lisp によって書かれていて、インタープリターで実行されるたくさんの関数によって動作します。（これらは速度のためにバイトコンパイルされていますが、インタープリターで実行されることに変わりはありません。）もし Gnus のある動作が好みではないと感じたら、それを違うやり方で実行させるのは取るに足らないことです。（えーと、少なくとも Lisp コードの書き方を知っていれば。）でもそれはこのマニュアルの範疇ではないので、Gnus をカスタマイズするために‘~/gnus.el’ファイルで普段使われるいくつかの一般的な構文のことだけを話すことにしましょう。（‘~/emacs’ファイルを使うこともできますが、Gnus に関する設定には‘~/gnus.el’ファイルを使う方がはるかに良いです。）

もし変数 gnus-florgbnize を四 (4) に設定したいのであれば、以下のものを書きましょう:

```
(setq gnus-florgbnize 4)
```

この関数 (本当は「特殊形式」(special form)) `setq` は、変数を何かの値に設定することができるものです。これがあなたが本当に知っていなければならないことのすべてです。これからは Gnus の動作を変更するために、たくさんのこういうもので ‘~/gnus.el’ ファイルを埋め尽くすことができます。

そういうものを ‘~/gnus.el’ ファイルに入れておくと、それらは次回に Gnus を起動したときに読み込まれ、`eval` (それは「実行」の Lisp 語です) されます。もし変数をすぐに変更したいのであれば、閉じ括弧の後ろで `C-x C-e` とタイプすれば良いのです。それは前にある「式」(ここでは簡単な `setq` 文) を `eval` します。

さあ、やってみましょう—あなたが Emacs の前にいるのなら試してみてください。 `C-x C-e` をタイプすると、エコーエリア (訳注: 一般には Emacs の画面の一番下) に ‘4’ が現われるのが見えるでしょう。それはあなたが `eval` した式の戻り値です。

いくつかの落とし穴:

もしマニュアルが「gnus-read-active-file を some に設定しなさい」と言ったなら、それは

```
(setq gnus-read-active-file 'some)
```

ということです。

一方、マニュアルが「gnus-nntp-server を ‘nntp.ifi.uio.no’ に設定しなさい」と言ったなら、それは

```
(setq gnus-nntp-server "nntp.ifi.uio.no")
```

ということです。

ですから、文字列 (後者) をシンボル (前者) と混同しないように注意して下さい。マニュアルは明確に区別していますが、混乱しやすいかもしれません。

10.10 よく尋ねられる質問

要約

これは新しい Gnus のよく尋ねられる質問のリストです。Web ブラウザーがあれば <http://my.gnus.org/FAQ/> にある公式なハイパーテキスト版を読むことができます。Docbook のソースは <http://sourceforge.net> (<http://sourceforge.net/projects/gnus/>) から手に入ることができます。

機能に関することや提案は FAQ discussion メーリングリスト (faq-discuss@my.gnus.org) に送って下さい。このメーリングリストは qconfirm (<http://smarden.org/qconfirm/index.html>) によって、ごみメールから守られています。参加者からの投稿は自動的に通過するでしょう。さらに空のメールを faq-discuss-subscribe@my.gnus.org に送るか、ここ (<http://mail1.kens.com/cgi-bin/ezmlm-browse?command=monthbythread%26list=faq-discuss>) を閲覧することによっても、このメーリングリストを講読することができます (残念ながら現在は壊れています)。

変更履歴

- Gnus 5.10 のリリースと No Gnus の開発の開始を反映して FAQ を更新しました。

序

これは Gnus のよく尋ねられる質問のリストです。

Gnus は Emacs の要素として実装された Usenet ニュースリーダーおよび電子メールのユーザーエージェントです。それはほぼこの十年間何らかの形で存在しており、その期間の多くにおいて Emacs の標準要素として配布されてきました。Gnus 5 は最新の (そして最も偉大な) 作品です。オリジナルの版は GNUS と言い、梅田政信さんが書きました。1994 年の秋が忍び寄る頃、退屈していたラルス・マッグヌ・イングブリグットスン (Lars Magne Ingebrigtsen) は Gnus を書き直そうと決心しました。

その最大の強みは、極めてカスタマイズに適しているという事実にあります。このことを始めて目にするると引いてしまうかもしれませんが、それを利用する準備ができるまでは、複雑なもののほとんどは無視することができます。そこそこの量の (いろんなメーリングリストに配信される) 電子メールがやって来るのならば、流通量が多いメーリングリストを読みたいけれども遅れずについていくことができないのならば、流通量が多いニュースグループを読んでいるのならば、あるいは単に退屈しているのならば、Gnus はあなたが望むものです。

この FAQ は 2002 年 3 月まで Justin Sheehy によって維持されていました。彼は、それ以前に素晴らしい仕事をしてくれた Steve Baur と Per Abrahamsen に感謝を表明しています。私たちも同じことをしましょう - ありがとう Justin!

Web ブラウザーがあれば <http://my.gnus.org/FAQ/> にある公式なハイパーテキスト版を読むことができます。この版は、ユトレヒト大学、オックスフォード、Smart Pages、オハイオ州立大学に保存されている非公式なハイパーテキスト版や他の FAQ のアーカイブに比べて、はるかに良いものです。別のフォーマットでそれを手に入れるための情報が欲しいなら、以下にある質問集を見て下さい。

ここにある情報は、Gnus 開発メーリングリストの援助でコンパイルされました。どんなエラーあるいはミスプリントも my.gnus.org チームが犯した誤りです。すみません。

訳注: そしてどんな誤訳の責任も gnus-doc-ja チームにあります。

10.10.1 インストールに関する FAQ

質問 1.1

Gnus の最新版は何ですか？

回答

ジャーン: Gnus 5.10 がリリースされました。熱いうちに召し上がれ！バージョン番号の歩みがいささか小さいのに反して、Gnus 5.10 には見逃せない何トンもの新しい機能があります。現在のリリース (5.10.8) は、少なくとも 5.8 系のリリースの最新版と同じくらい安定なはずです。

質問 1.2

5.10 では何が新しいですか？

回答

第一に Gnus tarball の先頭のディレクトリーにある GNUS-NEWS ファイルに目を通すべきです。そこでは最も重要な変更が羅列されています。ここでは特に重要/興味深いものの短いリストを挙げるに留めます:

- Gnus エージェントの大幅な書き直し。デフォルトで有効になっています。
- 見苦しく形成された記事を洗濯するための多くの機能。
- Spam 除け機能。
- Message-utils が Gnus に含まれました。
(訳注: Holger Schauer さんが書いた message-utils.el の諸機能が message.el に移入されました。)
- 概略行のための新しい書法仕様。例えば %B は複雑な trn 様式のスレッド木 (tree) を指定する書法仕様です。

質問 1.3

Gnus はどこで、どうやって取得することができますか？

回答

Gnus は Emacs や XEmacs のリリースとは別に、独自にリリースされます。そのため Emacs に同梱されている版や XEmacs パッケージにある版は、最新ではないかもしれませんが (例えば Emacs 20 に同梱されている Gnus 5.9 は、使用期限が切れています)。リリースされた最新版の Gnus は、<http://www.gnus.org/dist/gnus.tar.gz> から、または匿名 ftp で <ftp://ftp.gnus.org/pub/gnus/gnus.tar.gz> から手に入れることができます。

質問 1.4

tarball で何をすれば良いですか？

回答

‘tar xvzf gnus.tar.gz’ でアーカイブを展開して、ありふれた
‘./configure; make; make install’ の手順を実行して下さい。

MS-Windows では <http://www.cygwin.com> から Cygwin の環境も取得して下さい。それによって上述のことを行なうこと、またはある梱包器 (packer) (例えば

<http://www.winace.com> にある Winace) で tarball を開梱すること、そして tarball に含まれている Gnus をインストールするためのパッチファイル 'make.bat' を使うことができるようになります。

Gnus をシステム領域にインストールしたくない (またはその権限が与えられていない) ならば、ホームディレクトリーにインストールすることもできます。その場合は '~/.xemacs/init.el' ファイルか '~/.emacs' ファイルに以下の行を加えて下さい。

```
(add-to-list 'load-path "/path/to/gnus/lisp")
(if (featurep 'xemacs)
    (add-to-list 'Info-directory-list "/path/to/gnus/texi/")
    (add-to-list 'Info-default-directory-list "/path/to/gnus/texi/"))
```

この行より前に、どんな Gnus に関係するものも確実に無いようにして下さい。MS Windows では "C:/path/to/lisp" のように書いて下さい (そう、"/" です)。

質問 1.5

ときどき目にする No Gnus と Oort Gnus って何ですか？

回答

Oort Gnus は Gnus の開発版の名前で、2003 年の秋に Gnus 5.10 になりました。No Gnus は現行の開発版の名前で、Gnus 5.12 か Gnus 6 になるでしょう。(なぜ 5.11 ではないのかが不思議ですか？ 奇数のバージョン番号は通常 Emacs に同梱される Gnus の版に使われるのです。)

質問 1.6

Emacs のどの版が必要ですか？

回答

Gnus 5.10 は Emacs 20.7 以上、または XEmacs 21.1 以上を必要とします。開発版の Gnus (No Gnus として知られているもの) は Emacs 21 か XEmacs 21.4 を必要とします。

質問 1.7

Gnus を Emacs と XEmacs の両方で走らせるには、どうすれば良いですか？

回答

バイトコンパイルされた Gnus の同じコピーを両者で使うことはできません。Emacs では Emacs でコンパイルしたものを、XEmacs では XEmacs でコンパイルしたものを使って下さい。

10.10.2 起動 / グループバッファー

質問 2.1

いつも Gnus を起動すると "Gnus auto-save file exists. Do you want to read it?" というメッセージを受け取るのですが、これは何を意味しているのですか？ また、どうやったら回避できますか？

回答

このメッセージが意味するのは、最後に Gnus を使ったときに適切に終了させなかったために、ディスクにその情報 (例えばどのメッセージを読んだかの) を書き込むことができなかったので、今、

それらの情報を自動保存 (auto-save) ファイルから復活させるかどうかを尋ねられているということです。

このメッセージが発せられるのを回避するには、Gnus を終了させるときに単に Emacs を kill するのではなく、グループバッファで `q` を使うようにして下さい。

質問 2.2

Gnus は私がどのグループを講読するかを覚えてくれませんか。どうしてですか？

回答

Gnus を起動したときに、上記の質問と回答 (see [[2.1]], page 359) で述べられているメッセージを受け取りませんでしたか？ これは同じ問題の別の症状なので、上記の回答を読んで下さい。

質問 2.3

グループバッファの各行の形式を変更するには、どうしたら良いですか？

回答

変数 `gnus-group-line-format` の値を調整しなければなりません。そのやり方についてはマニュアル (see section “グループ行の仕様” in *The Gnus Manual*) を見て下さい。このための例です (質問者の ‘`~/gnus.el`’ ファイルからの推測です :-):

```
(setq gnus-group-line-format "%P%M%S[%5t]%5y : %(%g%)\n")
```

質問 2.4

私のグループバッファはちょっと混んでいるのですが、それらを楽に巡回できるように、カテゴリごとにまとまるように並べ変える方法はありますか？

回答

Gnus はトピックモードを提供します。それによってグループをその中へ入れて並べ変えることができるようになります。えーとトピックを使うというのは、例えば Linux を扱うすべてのグループは `linux` というトピックに収め、音楽を扱うすべては `music` というトピックに収め、スコットランド音楽を扱うすべては `scottish` という `music` のサブトピックに収める、といったことです。

トピックモードに入るには、グループバッファで単に `t` を叩いて下さい。すると `T n` を使って現在位置でトピックを作ったり、あるグループを `T m` を使って指定したトピックに移すことができます。さらなるコマンドについてはマニュアルかメニューを見て下さい。グループ行をより良く行下げ (indent) させるためには、`gnus-group-line-format` 変数の先頭に `%P` 書法仕様を含ませる必要があるでしょう。

質問 2.5

グループバッファを手で並べ変えるにはどうしたら良いですか？ トピック内のグループを並べ変えるにはどうしたら良いですか？

回答

移動させたいグループの上にポイントを移動して `C-k` を叩き、次にそのグループを移動させる目的の場所にポイントを移動して `C-y` を叩いて下さい。

10.10.3 メッセージの取得

質問 3.1

今まさに Gnus をインストールして *M-x gnus* で起動したのですが、`"nntp (news) open error"` としか言ってくれません。どうしたら良いですか？

回答

どこからニュースを取得すべきかを Gnus に教えてあげなければなりません。やり方については文献を読んで下さい。初めて起動するのであれば、次のようなものを `~/ .gnus.el` ファイルに書き込んで下さい:

```
(setq gnus-select-method '(nntp "news.yourprovider.net"))
(setq user-mail-address "you@yourprovider.net")
(setq user-full-name "Your Name")
```

質問 3.2

Windows を使っているのですが `~/ .gnus.el` の意味がわかりません。

回答

`~/` とは Gnus と Emacs が設定ファイルを探す場所であるホームディレクトリーのことです。でも、実はその意味を知らなくても構わないのです。Emacs がわかっていれば十分ですから。:-) *C-x C-f ~/ .gnus.el RET* とタイプすれば (そう、Windows でもスラッシュでいいのです) Emacs は正しいファイルを開いてくれるでしょう (それが新規ファイルであるために中身が空っぽであることは、おおいにあり得るでしょう)。しかしちょっと待って下さい。Emacs が選ぶディレクトリーは、きっとあなたの希望通りにはならないので、正しいやり方でそれを行ないましょう。第一に、例えば `c:\myhome` のような適当なディレクトリーを (ディレクトリー名に空白を含めないで) 作って下さい。そして、このディレクトリーを環境変数 HOME に設定しましょう。これを Win9x か Me で行なうには、`autoexec.bat` ファイルに以下の行を追加して、再起動して下さい。

```
SET HOME=C:\myhome
```

NT, 2000 および XP では、システム・オプションを入力するために *Winkey + Pause/Break* を叩いて (もしそれが使えなかったら「コントロールパネル -> システム」を辿って) 下さい。そこで環境変数を設定できるでしょうから、HOME という名前で値が `C:\myhome` のものを作して下さい。再起動は不要です。

では、Emacs に *C-x C-f ~/ .gnus.el RET C-x C-s* を指示して、`~/ .gnus.el` ファイルを作りましょう。

質問 3.3

ニュースサーバーが認証を要求します。ディスクにユーザー名とパスワードを保存しておくには、どうすれば良いですか？

回答

次のように、それぞれのサーバーに関する行を含んだ `~/ .authinfo` ファイルを作して下さい。

```
machine news.yourprovider.net login YourUserName password YourPassword
```

OS が対応しているならば、そのファイルを他人が読めないようにしておきましょう。Unix ではシェル上で次のコマンドを実行して下さい。

```
chmod 600 ~/ .authinfo
```

質問 3.4

Gnus はうまく起動したようなのですが、グループを講読する方法が見つかりません。

回答

そのグループの名前がわかっているのなら、グループバッファで `U name.of.group RET` を使って下さい (タブ補完を使え、ルーク (訳注: オビ=ワン・ケノービの声で))。あるいはグループバッファで `^` を使いましょう。これはあなたをサーバーバッファへ連れて行きます。その場合は、目的のグループがあるサーバーの上にポイント (カーソル) を置いて `RET` を叩き、読みたいグループにポイントを移動してから `u` でそのグループを講読しましょう。

質問 3.5

Gnus がすべてのグループを表示してくれません / このサーバーへの投稿が許可されません。どうしてですか?

回答

プロバイダーのいくつかは匿名での接続を制限していて、認証してからでないと完全なアクセスを許しません。Gnus に認証のための情報 (authinfo) を送出させるには、`~/ .authinfo` ファイルの該当するサーバーの行の最後に

```
force yes
```

を追加して下さい。

質問 3.6

複数のサーバーからニュースを取り込みたいのですが、それは可能ですか?

回答

もちろん。変数 `gnus-secondary-select-methods` に、もっと多くの記事の供給元を設定することができます。以下のようなものを `~/ .gnus.el` ファイルに追加して下さい:

```
(add-to-list 'gnus-secondary-select-methods
              '(nntp "news.yourSecondProvider.net"))
(add-to-list 'gnus-secondary-select-methods
              '(nntp "news.yourThirdProvider.net"))
```

質問 3.7

それから、ローカル・スプール・ファイルからニュースを取り込むことは可能ですか?

回答

問題ありません。それは `nnsPOOL` というもう一つの選択方法です。こんなふうに設定して下さい:

```
(add-to-list 'gnus-secondary-select-methods '(nnsPOOL ""))
```

あるいは NNTP を第一ニュースソースとして使う必要が無いのであれば、こうして下さい:

```
(setq gnus-select-method '(nnsPOOL ""))
```

これで Gnus は `/usr/spool/news` にあるスプール・ファイルを探します。何か違うことをやりたいのならば、上記の行を次のようなものに変更して下さい:

```
(add-to-list 'gnus-secondary-select-methods
              '(nnsPOOL ""))
```

```
(nnsPOOL-directory "/usr/local/mysPOOLddir"))
```

これは、このサーバーだけのために、スプールが存在するディレクトリーを設定します。さらに記事を投稿するために使うプログラムなどを設定する必要があるかもしれません。そういう場合にどうしたら良いかについては、Gnus のマニュアルを参照して下さい。

質問 3.8

ニュースを読むのうまくいきましたが、Gnus でメールも読めるようにしたいのです。どうすれば良いですか？

回答

それは少しばかり難しいです。使うことができるメールソースはいっぱいあるし、メールを格納する方法はたくさんあるし、送信するための方法も様々なので。最もありふれているのは、次の二つの事例のようなものでしょう：

1. POP3 サーバーからメールを読み、SMTP サーバーに直接メールを送信します。
2. fetchmail のようなプログラムでメールを取り込んで、Gnus が読むことになっているディレクトリーに格納します。外に行くメールは Sendmail, Postfix または他の MTA によって送出されます。

ときには、これらを併用する必要さえあります。

しかし最初に行なうことは、どの方法でメールを格納するか、Gnus の用語で言うと、どのバックエンドを使うかを Gnus に指示することです。Gnus は多くの異なるバックエンドをサポートしますが、最も一般的に使われているのは nnml です。それは一通のメールを一つのファイルに格納し、そのため極めて高速です。でも、あなたが使っているファイルシステムがたくさんの小さなファイルを扱う上で問題があるのならば、一つのグループのすべてのメールを一つのファイルに収める方法を使う必要があるかもしれません。おそらくそういう場合の選択肢が nnfolder バックエンドです。

nnml を使うには、以下を ‘~/ .gnus.el’ ファイルに加えて下さい：

```
(add-to-list 'gnus-secondary-select-methods '(nnml ""))
```

nnfolder を使いたいのであれば、あなたの想像した通り、こうすれば良いでしょう：

```
(add-to-list 'gnus-secondary-select-methods '(nnfolder ""))
```

次に、メールをどこから取得するかを Gnus に伝えなければなりません。それが POP3 サーバーであるのなら、このように設定して下さい：

```
(add-to-list 'mail-sources '(pop :server "pop.YourProvider.net"
                                :user "yourUserName"
                                :password "yourPassword"))
```

もしそこにパスワードを書きしておくのなら、他人が ‘~/ .gnus.el’ ファイルを読めないようにしておいて下さい。メールをローカルマシンの伝統的なスプールファイルから読みたい場合は、以下のよう設定して下さい：

```
(add-to-list 'mail-sources '(file :path "/path/to/spool/file"))
```

もしそれが Postfix, Qmail または (そういうふうに設定されている) fetchmail によって使われる、一通/一ファイル形式の Maildir であるのなら、こんなふうにししょう：

```
(add-to-list 'mail-sources '(maildir :path "/path/to/Maildir/"
                                     :subdirs ("cur" "new")))
```

そして最後に、メールを一つのディレクトリーにある複数のファイルから読むようにしたいのであれば (例えば procmail がすでに分割してあるという理由で)、設定は以下のようになります：

```
(add-to-list 'mail-sources
  '(directory :path "/path/to/procmail-dir/"
    :suffix ".prcml"))
```

ここで `:suffix` に指定した `".prcml"` は、Gnus に拡張子が `.prcml` のファイルだけを使うことを指示するためのものです。

さあ、後はどうやってメールを送信するかを Gnus に教えるだけです。メールの送信に `sendmail` を使いたければ（または、あなたのシステムの MTA が何であれ `sendmail` の役を演じるのならば）、何もしなくても良いのです。でも、もし SMTP サーバーにメールを送りたいのだったら、以下のようなのが `~/gnus.el` ファイルに書かれている必要があります：

```
(setq send-mail-function 'smtpmail-send-it)
(setq message-send-mail-function 'smtpmail-send-it)
(setq smtpmail-default-smtp-server "smtp.yourProvider.net")
```

質問 3.9

それから、IMAP でメールを読めるようにするには、どうすれば良いですか？

回答

Gnus で IMAP を使うには二つの方法があります。最初のは IMAP を POP3 のように使うもので、Gnus は IMAP サーバーからメールを取得してディスクに格納します。これをやりたいのなら（本当にそんなことをしたい人はいないでしょうけれど）、以下の設定を `~/gnus.el` ファイルに加えて下さい：

```
(add-to-list 'mail-sources '(imap :server "mail.mycorp.com"
  :user "username"
  :pass "password"
  :stream network
  :authentication login
  :mailbox "INBOX"
  :fetchflag "\\Seen"))
```

`:stream` および/または `:authentication` の項はいじる必要があるかもしれません。使うことができる値については Gnus マニュアル (see section “メールソース指示子” in *The Gnus Manual*) を参照して下さい。

IMAP をそれが意図された方法で使いたい場合は、違うやり方に従う必要があります。そうするには選択方法に `nnimap` を加え、そこでサーバーに関する情報を与えて下さい：

```
(add-to-list 'gnus-secondary-select-methods
  '(nnimap "Give the baby a name"
    (nnimap-address "imap.yourProvider.net")
    (nnimap-port 143)
    (nnimap-list-pattern "archive.*")))
```

さらに Gnus が正しいやり方を推測できない場合には、サーバーに認証してもらう方法を指定しなければなりません。詳しい情報はマニュアル (see section “IMAP” in *The Gnus Manual*) を参照して下さい。

質問 3.10

職場で MS Exchange サーバーの一つを使っているのですが、Gnus を使ってそこからメールを読むことはできますか？

回答

サーバーに IMAP という新しい運動靴を一足はかせてくれるように、管理人にお願いして下さい。そして上記の説明に従って、必要なことを行なって下さい。

質問 3.11

POP3 でメールを取得するサーバーで、メールを消さないように Gnus に指示することはできますか？

回答

そもそも POP3 はそういうふうに動作することが意図されていません。それが可能なら、メッセージをサーバー上に残すためには IMAP プロトコルを使うべきです。それでもなお、そういう機能が必要とする状況があるかもしれませんが、悲しいかな、Gnus 自体はそうするための既設の機能を持っていないのです。

しかしここは Gnus 村ですから、あなたの希望を達成する見込みが無いわけではありません。最も簡単な方法は、メールのコピーを取り込んで、それらをディスクに格納してくれる外部プログラムを手に入れることです。そうすれば Gnus はそこから読むことができます。例えば Unix システムでは fetchmail がそれです。MS Windows では、優れたニュースとメールのサーバーである Hamster を使うことができます。

別の解は、Gnus が POP3 サーバーからメールを取得する手段を、メールをサーバーに残すことができるもので置き換えることでしょう。XEmacs を使っているのであれば mail-lib パッケージを手に入れて下さい。それは拡張された ‘pop3.el’ を含んでいます。ファイルの中身を見て下さい。それを使うように Gnus に指示するやり方と、取得したメールを削除しない方法の説明が書かれています。GNU Emacs を使っている場合は、同じことを行なうことができる ‘epop3.el’ を探して下さい (そのファイルの元の出どころを知っている人は、私 (訳注: Simon Josefsson さんだと思います、たぶん) にメールを送って下さい)。さらに Gnus に外部プログラム (例えば fetchmail) を使ってメールを取り寄せるようにさせることもできます。やり方は Gnus マニュアル (see section “メールソース指示子” in *The Gnus Manual*) を参照して下さい。

訳注: T-gnus に含まれている ‘pop3.el’ は、XEmacs のやり方をさらに発展させて、完全に Gnus とともに動作するようになっています (した)。

10.10.4 メッセージを読む

質問 4.1

グループに入ると、以前に読んだメッセージが全部なくなっています。もう一度読むには、どうしたら良いですか？

回答

グループバッファにおいて、そのグループの上にポイントが置かれている状態で RET を使うと、未読と可視記事の印 (ticked) が付いたメッセージだけが現れます。存在するすべてのメッセージが表示されるようにするには、代わりに C-u RET を使って下さい。もし最新の、例えば 300 通だけが所望ならば、C-u 300 RET を使いましょう。

スレッド表示を有効にしていると、未読のメッセージだけを表示するのはじれったいかもしれません。スレッドがちぎれてしまうことを防止するのに十分な量の古い記事を取り込むには、以下の設定を追加すれば良いでしょう:

```
(setq gnus-fetch-old-headers 'some)
```

some を t に替えると、すべての記事を表示するようになります (警告: どちらの設定もグループに入ったときに取り込むデータ量を増加させ、グループに入る処理を遅くしてしまいます)。

すでに Gnus 5.10 を使っているならば、最後の n 通のメッセージを取り込むために、概略バッファで `/o N` 命令を使うことができます。この機能は 5.8.8 にはありません。

すべての古いメッセージは要らないけれども、今まさに読んでいるメッセージの親を辿りたいならば `^` 命令を使って下さい。今まさに読んでいるメッセージが属しているスレッドのすべての記事を表示したい場合は、`'A T'` 命令があなたの望むものです。

質問 4.2

大事なメッセージを、すでに読んだことがあっても、グループに入ったときはいつでも見えるように Gnus に指示するには、どうすれば良いですか？

回答

大事なメッセージには可視記事の印 (ticked) を付けることができます。それには、概略バッファでポイントがその記事を指しているときに `u` を叩いて下さい。可視記事の印を消す命令は `d` (これは可視記事の印を消してから未読の印を付けます) または `M c` (そのメッセージに付いているすべての印を消します) です。

質問 4.3

メッセージのヘッダーを見るにはどうしたら良いですか？

回答

`t` 命令を使うとすべてのヘッダーを表示し、もう一度使うとそれらを隠します。

質問 4.4

整形されていない生のメッセージを見るには、どうすれば良いですか？

回答

`C-u g` 命令を使うと生のメッセージを表示し、`g` で通常の表示に戻ります。

質問 4.5

記事バッファの先頭に Gnus がデフォルトで表示するヘッダーを変更するには、どうしたら良いですか？

回答

変数 `gnus-visible-headers` がどのヘッダーを表示するかを制御します。その値は正規表現で、それに合致するヘッダー行が表示されます。したがって、著者、表題、日付、そしてもし存在する場合は Followup-To と MUA / NUA を表示したい場合には、`'~/gnus.el'` ファイルで以下の設定を行なって下さい:

```
(setq gnus-visible-headers
      '("^From" "^Subject" "^Date" "^Newsgroups" "^Followup-To"
        "^User-Agent" "^X-Newsreader" "^X-Mailer"))
```

質問 4.6

HTML メールを描画するのではなく、テキストのパートがある場合には、そちらの方を Gnus に表示して欲しいです。どうすれば良いですか？

回答

‘~/gnus.el’ ファイルで以下の設定を行なって下さい:

```
(add-to-list 'mm-discouraged-alternatives "text/html")
(add-to-list 'mm-discouraged-alternatives "text/richtext")
```

HTML のパートと同じ内容のテキストのパートが無い場合でさえも HTML のパートを描画して欲しくないのであれば、以下の設定を追加して下さい:

```
(setq mm-automatic-display (remove "text/html" mm-automatic-display))
```

質問 4.7

HTML メールを w3 以外のブラウザを使って描画させることはできますか?

回答

Gnus 5.10 またはより新しいものを使う場合ですが、w3, w3m, links, lynx および html2textの中から選ぶことができ、どれを使うかは変数 mm-text-html-renderer で指定することができます。したがって links で HTML メールを描画したいときは、こうして下さい:

```
(setq mm-text-html-renderer 'links)
```

質問 4.8

見苦しい体裁のメールをもっと読みやすくするための何かがありますか?

回答

Gnus は入ってきたメールを「洗濯」するために複数の関数を提供します。それらはメニューの Article->Washing 項目を辿ることによって見つけることができるでしょう。最も興味深いものを挙げてみましょう:

W w (Wrap long lines)

長い行を折り畳みます。

W r (Decode ROT13)

ROT13 をデコードします。

W Y f (Outlook Deuglify)

Outlook を見苦しくなくさせます。これは、マイクロソフトの製品を使っている多くのユーザーが送ってくる間の抜けた引用付き返信 (訳注: 折り畳まれた長い引用行の二行目以降に引用符が前置されないなど) を修繕します。

他の見苦しくなくさせる機能については *W Y C-h* を使うかメニューを眺めて下さい。Outlook Deuglify は Gnus 5.10 以上で使うことができます。

質問 4.9

特定の著者が送信したものや、表題に特定の語が含まれているメッセージを、自動的に無視する方法はありますか? また、より興味深いものを何らかの方法で強調表示させることはできますか?

回答

あなたに必要なのはスコア付けです。スコア付けと言うのは、それぞれのメッセージに整数の値を割り付けるための規則を定義することです。値に基づいて概略バッファでメッセージを強調表示したり (それが +2000 といった高い値だったら)、自動的に既読にしたり (その値が例えば -800 のような低い値だったら)、あるいは他のいくつかの作用を行ったりします。

メッセージにスコアの値を割り付ける規則を設定するための、基本的な三つのやり方があります。

1. 最初の最も簡単なやり方は、今まさに読んでいる記事に基づいて規則を設定することです。いつもたわごとを書いてよこすやつからのメッセージを読んで、今後はそいつからのメッセージを無視しようと決心したならば、`L` を叩いてスコアを下げる規則を設定して下さい。

すると Gnus は、スコアを下げる基準をどれにすれば良いかを尋ねてくるでしょう。`?` を二回叩くと、すべての候補を見ることができますが、ここで選択すべきなのは著者 (From ヘッダー) を意味する `a` です。

次に Gnus はどの種類の合致を使うべきかを尋ねてくるので、厳密な合致のための `e` を叩くか、あるいは文字の一部への合致を求める `s` を叩いてから、後ですべてを削除して下さい。どんな電子メールアドレスであっても、それを持つすべての著者のスコアを下げるために、その名前が使われます。

さらに、その規則をいつ適用すべきか、またどれくらいの時間続けるべきかを Gnus に伝える必要があります。例えば `p` では、その規則を今すぐ適用して、それを永久に持続させます。

スコアを下げるのではなくて上げたいのであれば、`L` の代わりに `I` を使って下さい。

2. 規則を手で設定することもできます。そうするには概略バッファで `Vf` 命令を使って下さい。するとスコアファイルの名前を尋ねられるでしょう。一つのグループだけで有効なのは `'name.of.group.SCORE'` で、すべてのグループに対して有効なのが `'all.SCORE'` です。厳密な構文については Gnus のマニュアルを参照して下さい。それは一つの大きなリストで、その要素もまた多くのリストです。その後者の単位リストの第一要素はスコアの対象であるヘッダーで、残りは、何に合致するか、いくらのスコアを割り当てるか、規則を期限切れ消去するのはいつか、そしてどうやって合致させるかを示すもう一つのリストです。私にとっても興味を持ったならば、例えば以下のようなものを `'all.SCORE'` ファイルに加えてみて下さい:

```
((("references" ("hschmi22.userfqdn.rz-online.de" 500 nil s))
  ("message-id" ("hschmi22.userfqdn.rz-online.de" 999 nil s)))
```

これは私が書いたメッセージのスコアに 999 を加えて、それに対する (たぶん間接的な) 回答のメッセージのスコアに 500 を加えるでしょう。もちろん分別のある人は誰も、こんなことはしないでしょうが。:-)

3. 三番目の選択肢は適応スコア付けです。これは、Gnus があなたを観察して、あなたが何に興味を持って何に幻滅するかを見つけ出し、それらを反映した規則を設定してくれるものです。流通量の多いグループを読むときに、適応スコア付けは大きな助けになるでしょう。適応スコア付けをやってみたいならば、`'~/gnus.el'` ファイルで以下の宣言を行なって下さい:

```
(setq gnus-use-adaptive-scoring t)
```

質問 4.10

いくつかの (例えばメールの) グループで、スレッド表示をさせなくすることはできますか? あるいは、いくつかのグループに固有の変数を設定することができますか?

回答

グループバッファにいるときに、そのグループにポイントを移動してから `Gc` を叩いて下さい。すると、そのグループのためのオプションを設定することができるバッファが開きます。そして、そのバッファのおしまいの方で、そのグループでローカルに変数を設定するための項目を見つけることができるでしょう。スレッド表示をさせなくするには、変数名として `gnus-show-threads` を、値として `nil` を入力して下さい。作業を終えたら、そのバッファの最初の方にある `[done]` ボタンを叩きましょう。

質問 4.11

私を書いたメッセージとフォロー記事を、強調表示させることはできますか？

回答

「できますか？」なんて質問はやめて下さい。Gnus 村では答えはいつも yes なのですから。:-) これは三段階の作業になります。まず、それらの投稿のためのフェース (概略行がどんなふうに見えるかの仕様) を作りましょう。続いて、それらにいくばくかの特別なスコアを与えましょう。最後に、その新しいフェースを使うように Gnus に指示しましょう。それをどう行なうかの詳しい説明が [my.gnus.org \(http://my.gnus.org/node/view/224\)](http://my.gnus.org/node/view/224) にあります。

質問 4.12

特にメールのグループで、Gnus がグループバッファーに表示するメッセージの合計の数が非常に大きいのです。これはバグですか？

回答

いいえ、これは Gnus の設計上の問題で、これを直すには Gnus のバックエンドの主要な部分を実装し直さなければなりません。Gnus は最大の記事番号から最小の記事番号を減算したものが合計の記事数であると解釈します。これは Usenet のグループでは OK なのですが、メールのグループでたくさんのメッセージを消去したり移動すると計算に失敗します。この病を治療するには *C-u RET* でそのグループに入り (この命令は Gnus にすべてのメッセージを持って来させます)、*M P b* ですべてのメッセージに印を付けてから、*B m name.of.group* でメッセージを全部、それらが元あったグループに移動して下さい。この処理によってそれらは新しいメッセージ番号を持たせられて、合計の記事数は再び正しくなるでしょう (またもやそれらを消したり別のグループに移動するまでは)。

質問 4.13

概略バッファーと記事バッファーの配置が気に入らないのですが、どうやったら変更できますか？できるなら三面で表示させたいです。

回答

`gnus-add-configuration` 関数を呼ぶことによって、ウィンドウの配置を制御することができます。構文はいささか複雑ですが、マニュアルで非常に良く説明されています (see section “ウィンドウの配置” in *The Gnus Manual*)。いくつかのやさしい例を挙げてみましょう。

概略 25% 記事 75% というデフォルトの割合を、概略 35% 記事 65% の割合に変更します (記事のための 1.0 は、残った空き地を取るという意味です)：

```
(gnus-add-configuration
  '(article (vertical 1.0 (summary .35 point) (article 1.0))))
```

グループバッファーが左、概略バッファーが右上、記事バッファーが右下という三面配置です：

```
(gnus-add-configuration
  '(article
    (horizontal 1.0
      (vertical 25
        (group 1.0))
      (vertical 1.0
        (summary 0.25 point)
        (article 1.0)))))
```

```
(gnus-add-configuration
  '(summary
    (horizontal 1.0
      (vertical 25
        (group 1.0))
      (vertical 1.0
        (summary 1.0 point))))))
```

質問 4.14

概略バッファを見せるやり方が好きではありません。調整するには、どうすれば良いですか？

回答

変数 `gnus-summary-line-format` をいじくり回す必要があります。その値は著者、日付、表題などのようなものを表すシンボルの文字列です。使うことができる仕様のリストは See section “概略バッファの行” in *The Gnus Manual* と、しばしば忘れられてしまう See section “書法仕様変数” in *The Gnus Manual* およびその下位の各章で見つかるはずですが。そこでは書法仕様の概略を使うことができるようにしてくれる、カーソルの位置指定やタブ位置の指定のようなものを見つけることができるでしょう。残念ながら 5.8.8 ではタブ位置を固定させる機能が壊れています。

5.10 から、Gnus は非常にみごとな新しい書法仕様変数を提供しています。例えば `%B` はスレッド木を描き、また `%&user-date` は、どう表示するかが記事が発信されてからの経過時間に依存している日付を表示します。これらの両方を使っている例です：

```
(setq gnus-summary-line-format
      ":%U%R %B %s %-60=|%4L |%-20,20f |%&user-date; \n")
```

結果はこんなふうになります：

```
:0      Re: [Richard Stallman] rfc2047.el          | 13 |Lars Magne Ingebrigt |Sat
:0      Re: Revival of the ding-patches list      | 13 |Lars Magne Ingebrigt |Sat
:R > Re: Find correct list of articles for a gro| 25 |Lars Magne Ingebrigt |Sat
:0 \-> ...                                         | 21 |Kai Grossjohann   | 0:0
:R > Re: Cry for help: deuglify.el - moving stuf| 28 |Lars Magne Ingebrigt |Sat
:0 \-> ...                                         |115 |Raymond Scholz    | 1:2
:0 \-> ...                                         | 19 |Lars Magne Ingebrigt |15:3
:0      Slow mailing list                         | 13 |Lars Magne Ingebrigt |Sat
:0      Re: '@' mark not documented               | 13 |Lars Magne Ingebrigt |Sat
:R > Re: Gnus still doesn't count messages prope| 23 |Lars Magne Ingebrigt |Sat
:0 \-> ...                                         | 18 |Kai Grossjohann   | 0:3
:0 \-> ...                                         | 13 |Lars Magne Ingebrigt | 0:5
```

質問 4.15

やってきたメールをいろいろなグループに振り分けるには、どうしたら良いですか？

回答

Gnus はメールを分割するための二つの手段として、やさしい `nnmail-split-methods` と、もっと強力な特級メール分割の機能を提供します。ここでは最初のものだけについて述べますが、後者についてはマニュアル (see section “特級メール分割” in *The Gnus Manual*) を参照して下さい。

`nnmail-split-methods` の値はリストで、それぞれの要素は分割の規則を表すリストです。それぞれの規則は「合致する記事が行くべきグループ」と「合致すべき正規表現」の形式を持っていて、

最初に合致した規則が勝ちます。最後の規則は常に汎用の規則（正規表現 ‘.*’）でなければなりません。それは他のどんな規則にも合致しない記事が行くべき場所を示します。もしまだフォルダーが無かったならば、それは記事がそこに着地するとすぐに作られるでしょう。デフォルトでは、メールは規則に合致するすべてのグループに送られます。それを望まないならば（たぶん望まないでしょう）、以下のものを ‘~/gnus.el’ ファイルに入れて下さい:

```
(setq nnmail-crosspost nil)
```

たった一つの例は千の言葉を並べるよりも勝ります。そこで、ここでは私の `nnmail-split-methods` を紹介しましょう。私は、私が講読しているいくつかのメーリングリストから来るか、私を直接に目指して来たすべてのメールを濾過器に通すので、私が特別のグループの複写を送ることと、デフォルトのグループは ‘spam’ であることに注意して下さい。これらの規則は、私宛てに届いた spam のおよそ 80% を殺してくれます (電子メールアドレスは spammers がそれらを使用するのを防ぐために変更されます)。

```

'setq
nnmail-split-methods
'(("duplicates" "^Gnus-Warning:.*duplicate")
  ("XEmacs-NT" "^\\(To:\\|CC:\\).*localpart@xemacs.invalid.*")
  ("Gnus-Tut" "^\\(To:\\|CC:\\).*localpart@socha.invalid.*")
  ("tcsch" "^\\(To:\\|CC:\\).*localpart@mx.gw.invalid.*")
  ("BAfH" "^\\(To:\\|CC:\\).*localpart@.*uni-muenchen.invalid.*")
  ("Hamster-src" "^\\(CC:\\|To:\\).*hamster-sourcen@yahoogroups\\.\\(de\\|com\\).*.")
  ("Tagesschau" "^From: tagesschau <localpart@www.tagesschau.invalid>$")
  ("Replies" "^\\(CC:\\|To:\\).*localpart@Frank-Schmitt.invalid.*")
  ("EK" "^From:.*\\(localpart@privateprovider.invalid\\|localpart@workplace.invalid\\).")
  ("Spam" "^Content-Type:.*\\(ks_c_5601-1987\\|EUC-KR\\|big5\\|iso-2022-jp\\).*.")
  ("Spam" "^Subject:.*\\(This really work\\|XINGA\\|ADV:\\|XXX\\|adult\\|sex\\).*.")
  ("Spam" "^Subject:.*\\(\\=\\?ks_c_5601-1987\\?\\|\\=\\?euc-kr\\?\\|\\=\\?big5\\?\\).*.")
  ("Spam" "^X-Mailer:\\(.*BulkMailer.*\\|.*MIME:.*\\|\\)")
  ("Spam" "^X-Mailer:\\(.*CyberCreek Avalanche\\|.*http:\\|\\.com\\.")
  ("Spam" "^From:.*\\(verizon\\.net\\|prontomail\\.com\\|money\\|ConsumerDirect\\).*.")
  ("Spam" "^Delivered-To: GMX delivery to spamtrap@gmx.invalid$")
  ("Spam" "^Received: from link2buy.com")
  ("Spam" "^CC: .*azzrael@t-online.invalid")
  ("Spam" "^X-Mailer-Version: 1.50 BETA")
  ("Uni" "^\\(CC:\\|To:\\).*localpart@uni-koblenz.invalid.*")
  ("Inbox" "^\\(CC:\\|To:\\).*\\(my\\ name\\|address@one.invalid\\|adress@two.inval")
  ("Spam" "")))

```

10.10.5 メッセージの作成

質問 5.1

メールを送信したりニュース記事を投稿するために知っている必要がある基本的なコマンドは何ですか？

回答

新しいメールの作成を始めるには、グループバッファか概略バッファのどちらかで *m* を叩いて下さい。ニュース記事を投稿する場合には、グループバッファで *a* を叩いた後に手で Newsgroups ヘッダーに書き込むか、投稿するグループの概略バッファで *a* を叩いて下さい。

メールで返信する場合、元の記事を引用しないか後で手作業で引用するつもりならば *r* を、引用文を最初から取り込んでしまうのならば *R* を使って下さい。ニュースグループでフォロー記事を投稿する場合であれば *f* か *F* (*r* と *R* の関係に似ています) を使いましょう。

新たにヘッダーを挿入するならば ‘--text follows this line--’ の上に書いて、本文はその下に書いて下さい。書き上がったメッセージを送信するコマンドは *C-c C-c* で、後で仕上げるために ‘drafts’ グループに保存するためのコマンドは *C-c C-d* です。後者は *D e* で再び編集することができます。

質問 5.2

メールを作成するとき、自動的に行を折り返すにはどうすれば良いですか？

回答

No Gnus の場合は自動で行を折り返す機能がデフォルトで ON になります。変数 *message-fill-column* を参照して下さい。

他の Gnus の版では、こんなものを ‘~/gnus.el’ ファイルに入れて下さい:

```
(unless (boundp 'message-fill-column)
  (add-hook 'message-mode-hook
    (lambda ()
      (setq fill-column 72)
      (turn-on-auto-fill))))
```

いつもの通りに *M-q* を使うことによって、段落を再整形することができます。

質問 5.3

From, Organization, Reply-To などのヘッダーを自動生成したり、定型の署名 (signature) を自動的に挿入するには、どうしたら良いですか？

回答

他のやり方もありますが、これのためには投稿様式 (posting styles) を使うべきです (理由は後で書きます)。この例ならば、その構文が明確にわかるはずです:

```
(setq gnus-posting-styles
  '((".*"
    (name "Frank Schmitt")
    (address "me@there.invalid")
    (organization "Hamme net, kren mer och nimmi")
    (signature-file "~/signature")
    ("X-SampleHeader" "foobar")
    (eval (setq some-variable "Foo bar")))))
```

“.*” という設定はデフォルトのものです ([5.4], page 373 参照)。それ以降のリストの第一要素に使える値 (属性名) は *signature*, *signature-file*, *organization*, *address*, *name* または *body* です。属性名は文字列でも構いません。その場合、属性名はヘッダー名として使われ、その値は記事のヘッダーに挿入されます。ただし値が *nil* だったら、その名前のヘッダーは削除されます。

(eval (foo bar)) の形式を使うことも可能で、その場合 bar を引数に与えられて関数 foo が評価され、結果は捨てられます。

質問 5.4

投稿するグループによって異なる From ヘッダーや署名を自動挿入するには、どうすれば良いですか？

回答

これこそが投稿様式 (posting styles) の強みです。前の回答 (see [[5.3]], page 372) では、すべてのグループのためのデフォルトを設定するために ".*" を使いました。これに "^gmane" のような正規表現を使うと、それ以降の設定を gmane ニュースグループの階層で投稿する記事だけに適用させることができます。代わりに ".*binaries" を使うと、名前などが 'binary' という文字列を含んでいるグループに投稿する記事だけに、それらの設定が適用されます。

正規表現の代わりに関数を指定することもでき、それが評価されて真を返すときだけ、それに対応する設定が有効になります。この興味深い二つの候補は、現在のグループがニュースグループだったら t を返す message-news-p と、それと対になる message-mail-p です。

すべての合致する様式が適用されることに注意して下さい。以下を例にすれば、'gmane.mail.spam.spamassassin.general' に投稿すると ".*" で始まる設定が適用され、message-news-p 以下の設定も適用され、"^gmane" で始まる設定、それに "^gmane\\.mail\\.spam\\.spamassassin\\.general\$" 以下の設定のすべてが適用されるということです。このため、一般的な設定は先頭に置き、特定の条件を持つものは下の方に置くのが良いでしょう。

```
(setq
  gnus-posting-styles
  '((".*" ;;default
    (name "Frank Schmitt")
    (organization "Hamme net, kren mer och nimmi")
    (signature-file "~/signature"))
    ((message-news-p) ;;Usenet news?
    (address "mySpamTrap@Frank-Schmitt.invalid")
    (reply-to "hereRealRepliesOnlyPlease@Frank-Schmitt.invalid"))
    ((message-mail-p) ;;mail?
    (address "usedForMails@Frank-Schmitt.invalid"))
    ("^gmane" ;;this is mail, too in fact
    (address "usedForMails@Frank-Schmitt.invalid")
    (reply-to nil))
    ("^gmane\\.mail\\.spam\\.spamassassin\\.general$"
    (eval (set (make-local-variable 'message-sendmail-envelope-from)
              "Azzrael@rz-online.de")))))
```

質問 5.5

スペルチェッカーはありますか？できれば、書いたその場でチェックしてくれるものがあれば良いのですが。

回答

Emacs では単語の綴りをチェックするために ‘ispell.el’ を使うことができます。したがって最初に行なうのは、外部プログラム ispell (<http://fmg-www.cs.ucla.edu/fmg-members/geoff/ispell.html>) または同 aspell (<http://aspell.sourceforge.net/>) を適当な path にインストールしておくことです。次に ispell.el (<http://www.kdstevens.com/~stevens/ispell-page.html>) と、書いたその場でスペルチェックしてくれる flyspell.el (<http://www-sop.inria.fr/mimosa/personnel/Manuel.Ser>) を用意して下さい。‘ispell.el’ は Emacs とともに配布されているだけでなく、XEmacs パッケージシステムによっても手に入れることができます。‘flyspell.el’ も Emacs に同梱されているのとともに、パッケージシステムを介して入手することができる XEmacs の text-modes パッケージの一部です。したがって、それらを個別にインストールする必要は無いはずです。

‘ispell.el’ は外部プログラム ‘ispell’ を使うことを想定しています。‘aspell’ を選ぶなら、Emacs の設定ファイルで以下の宣言を行なって下さい (訳注: Emacs 22 以上では、‘ispell.el’ がそれらを自動判別します):

```
(setq ispell-program-name "aspell")
```

送出するメッセージがスペルチェックされるようにしたいなら以下の行を

```
(add-hook 'message-send-hook 'ispell-message)
```

単語を書いたその場でスペルチェックすることを好むなら次の行を

```
(add-hook 'message-mode-hook (lambda () (flyspell-mode 1)))
```

‘~/gnus.el’ ファイルに追加して下さい。

質問 5.6

投稿するグループに基づいて、辞書 (スペルチェッカーの) を切り替えることはできますか?

回答

はい、できます。こんなものを ‘~/gnus.el’ ファイルに入れて下さい:

```
(add-hook 'gnus-select-group-hook
  (lambda ()
    (cond
      ((string-match
        "^de\\\\" (gnus-group-real-name gnus-newsgroup-name))
        (ispell-change-dictionary "deutsch8"))
      (t
        (ispell-change-dictionary "english")))))
```

“^de\\.” と “deutsch8” は、必要に応じて変更して下さい。

質問 5.7

全員の電子メールアドレスを思い出さなくても済むようにするための、アドレス帳のようなものはありますか?

回答

そのための非常に基本的な解は「メールの別名」(mail aliases) です。以下のように単純な別名の構文を使って、‘~/mailrc’ ファイルにメールアドレスを登録しておくことができます:

```
alias al "Al <al@english-heritage.invalid>"
```


そうしておいて、メッセージバッファの To: または Cc: 行で、別名に続いてスペースか句読点をタイプすることによって、Gnus に完全なアドレスを挿入してもらうことができます。詳細は Message マニュアル (see section “メールの別名” in *The Message Manual*) を参照して下さい。

でも、あなたが本当に使いたいのは BBDB (the Insidious Big Brother Database) でしょう。XEmacs のパッケージシステムを使うか、bbdb のホームページ (<http://bbdb.sourceforge.net/>) から入手して下さい。そして Gnus で BBDB を有効にするために、以下のものを ‘~/gnus.el’ ファイルに書き込んで下さい:

```
(require 'bbdb)
(bbdb-initialize 'gnus 'message)
```

さて、いくつかの一般的な BBDB の設定が必要かもしれません。それらは ‘~/emacs’ ファイルに置きましょう。例です:

```
(require 'bbdb)
;; 北アメリカに住んでいるのであれば、以下によって電話番号の
;; チェックをやめさせるべきです。
(setq bbdb-north-american-phone-numbers-p nil)
;; あなたの電子メールアドレスを BBDB に教えましょう。
(setq bbdb-user-mail-names
  (regexp-opt '("Your.Email@here.invalid"
                "Your.other@mail.there.invalid")))
;; メールアドレスを補完するときに、候補をエンドレスで出します。
;; cycling while completing email addresses
(setq bbdb-complete-name-allow-cycling t)
;; BBDB のバッファをポップアップさせません。
(setq bbdb-use-pop-up nil)
```

これで BBDB を使う準備ができたはずです。M-x bbdb RET RET で、すべての登録した項目を表示する BBDB のバッファを開いて下さい。新たに登録するには c、検索には b、そして登録してある項目に新しいフィールドを加えるには C-o を使いましょう。送信者を BBDB に登録するには、概略バッファのその記事の場所で単に : を叩くだけで、あなたの仕事は終わりです。一方、新規にメールを作成しているときに TAB を叩くことによって、順繰りに現れる候補の中から受取人を選ぶことができます。

質問 5.8

記事バッファの上の方で、ときどき小さな画像を目にします。あれは何ですか？ また、どうしたら私も投稿するときに付けることができますか？

回答

あの画像は X-Face というものです。ヘッダー行で 48 × 48 画素の白黒画像がエンコードされています。それを送信する記事に含めたいなら、何かの画像を X-Face に変換する必要があります。そうするには、何らかの画像を加工するためのプログラム (例えば Gimp) に点火して、記事に含めたい画像のファイルを開き、必要な部分を切り抜いて、色の深度を 1-bit まで減らし、48 × 48 の大きさに縮小または拡大して、bitmap としてファイルに保存して下さい。次に ‘compface’ パッケージをこのサイト (<ftp://ftp.cs.indiana.edu:/pub/faces/>) から入手して、以下を実行することによって実際の X-Face を作りましょう:

```
cat file.xbm | xbm2ikon | compface > file.face
cat file.face | sed 's/\\/\n/g;s/"/\"/g;' > file.face.quoted
```

‘compface’を使うことができなくても、オンラインの X-Face 変換器が <http://www.dairiki.org/xface/> にあります。MS Windows を使っているのならば、<http://www.xs4all.nl/~walterln/winface/> から ‘WinFace’ プログラムを取って来て使うこともできます。後は、送信する記事に X-Face を含めてくれるように Gnus に指示するだけです。それには ‘~/ .gnus.el’ ファイルに以下のようなものを入れて下さい:

```
(setq message-default-headers
  (with-temp-buffer
    (insert "X-Face: ")
    (insert-file-contents "~/ .xface")
    (buffer-string)))
```

ただし Gnus 5.10 を使っているのであれば、単に次の項を

```
(x-face-file "~/ .xface")
```

gnus-posting-styles に加えるだけで済みます。

質問 5.9

ときどきニュースグループで、*f* の代わりにうっかり *r* を打ってしまいます。ニュースグループなのにもかかわらずメールで返信しようとしたときに、Gnus に警告してもらうことはできますか?

回答

もう Gnus 5.10 を使っているのであれば、これを ‘~/ .gnus.el’ ファイルに入れて下さい:

```
(setq gnus-confirm-mail-reply-to-news t)
```

まだ 5.8.8 や 5.9 を使っているのなら、代わりにこれをどうぞ:

```
(eval-after-load "gnus-msg"
  '(unless (boundp 'gnus-confirm-mail-reply-to-news)
    (defadvice gnus-summary-reply (around reply-in-news activate)
      "Request confirmation when replying to news."
      (interactive)
      (when (or (not (gnus-news-group-p gnus-newsgroup-name))
        (y-or-n-p "Really reply by mail to article author? "))
        ad-do-it))))))
```

質問 5.10

Gnus が sender ヘッダーを生成しないようにするには、どうしたら良いですか?

回答

Gnus 5.10 はデフォルトで sender ヘッダーを作らないのですけれどね。まだ古い Gnus を使っているのなら、‘~/ .gnus.el’ ファイルでこれを試して下さい:

```
(add-to-list 'message-syntax-checks '(sender . disabled))
```

質問 5.11

送信したメールやニュースの控えをローカルに残しておきたいのですが、どうすれば良いですか?

回答

それを行なうためには変数 `gnus-message-archive-group` を設定しなければなりません。それには、控えを保存しておくグループの名前を与える文字列を設定することができます。あるいは以下の例のように、評価されるとグループ名を返す関数（訳注: と言うよりは Lisp の式）を設定することもできます。

```
(setq gnus-message-archive-group
      '((if (message-news-p)
            "nnml:Send-News"
            "nnml:Send-Mail")))
```

質問 5.12

Message-ID が不正だと言われてしまうのですが、それはなぜですか？ また、どうやって直したら良いですか？

回答

Message-ID は、送信したメッセージが、それがそれであることを確認するためのものです。それが、あるメッセージにとって固有であるようにするために、Gnus は '@' の後に置くマシン名を知る必要があります。Gnus が走っているマシンの名前が適切ではないならば（ほとんどの個人のマシンが該当するでしょう）、Gnus が使う名前を以下のように '~/.gnus.el' ファイルで設定して下さい:

```
(setq message-user-fqdn "yourmachine.yourdomain.tld")
```

Gnus 5.9 かそれ以前のものを使っている場合は、代わりにこれを使って下さい（新しい版でも動作します):

```
(let ((fqdn "yourmachine.yourdomain.tld")) ;; <-- 変更してね!
  (if (boundp 'message-user-fqdn)
      (setq message-user-fqdn fqdn)
      (gnus-message 1 "Redefining 'message-make-fqdn'."))
  (defun message-make-fqdn ()
    "Return user's fully qualified domain name."
    fqdn)))
```

"yourmachine.yourdomain.tld" に何を入れるかを決めることができないならば、複数の選択肢があります。あなたが 'yourUserName.userfqdn.provider.net' のようなものを使っても良いかどうかをプロバイダーに尋ねてみても良いし、個人でドメイン 'yourdomain.tld' を保有している場合は 'somethingUnique.yourdomain.tld' のようなものを使うことができます。あるいは、ユーザーに無料で FQDN を提供してくれるサービス、例えば <http://www.stura.tu-freiberg.de/~dlx/addfqdn.html> に登録するのも良いでしょう（でもこのウェブサイトはドイツにあります。同じサービスを提供している英語圏のものを知っているのならば、私に手紙を書いて下さい）。

最後に、Gnus にニュース記事では Message-ID をまったく作らせない（そしてその仕事はサーバーに任せる）ことができます。以下の設定を使って下さい:

```
(setq message-required-news-headers
      (remove 'Message-ID message-required-news-headers))
```

さらに次の設定によって、メールでも Gnus に Message-ID を作らせないことは可能です:

```
(setq message-required-mail-headers
      (remove 'Message-ID message-required-mail-headers))
```

しかしながら、メールサーバーのあるものは適切な Message-ID を作ってくれないので、自分でメールを送信して Message-ID を眺めてみることによって、あなたのメールサーバーが正しく振る舞うかどうかをテストして下さい。

10.10.6 古いメッセージ

質問 6.1

Gnus を使っていなかったときの古いメールを、Gnus に編入させるにはどうしたら良いですか？

回答

最も楽な方法は、古いメールプログラムにメッセージを 'mbox' 形式にまとめてもらうことです。ほとんどの Unix のメイラーはそれを行うことができ、あなたが MS Windows の世界の出身であっても <http://mbx2mbox.sourceforge.net/> で、そのための道具を見つけることができるでしょう。

では、この 'mbox' ファイルを Gnus に編入させましょう。それにはグループバッファで `gf /path/file.mbox RET` を実行して、'mbox' ファイルを扱うための `nndoc` グループを作して下さい。そうすれば、あなたのメールに読み出し専用でアクセスできるようになります。そのメッセージを通常の Gnus のメールグループの階層に編入させたいならば、たった今作った `nndoc` グループに `C-u RET` 命令で入って (つまりすべてのメッセージが読めるようにしておいて)、すべてのメッセージに `MP b` で印を付けてから、それらをお望みのグループに `B c name.of.group RET` でコピーするか、または `nnmail-split-methods` を使ってそれらを分割 (再スプール) するために `B r` を使って下さい。

質問 6.2

興味を持った記事を保存するにはどうすれば良いですか？

回答

例えば 'gnu.emacs.gnus' で興味深い記事に出くわして、それを保存しようと思ったときには、複数の解があります。第一の最もやさしい方法は、`Of` 命令を使ってそれをファイルに保存することです。でもそれは、Gnus から保存されたメッセージをより直接にアクセスする手段としては、とても便利だとは言えないのではないのでしょうか？ それに同意してくれるのならば、Frank Haun pille3003@fhaun.de が書いてくれたこのコードの切れ端を '~/.gnus.el' ファイルに入れてみて下さい:

```
(defun my-archive-article (&optional n)
  "一個以上の記事を、対応する 'nnml:' グループにコピーします。
  例えば 'gnus.ding' の記事は 'nnml:1.gnus.ding' グループに行き、
  'nnml:List-gnus.ding' の記事は 'nnml:1.List-gnus-ding' に行きます。

  一個以上の記事を保存するには、概略バッファでプロセス/接頭引数の
  習慣を使って下さい。"
  (interactive "P")
  (let ((archive-name
        (format
         "nnml:1.%s"
         (mm-replace-in-string gnus-newsgroup-name "^.*:" ""))))
    (gnus-summary-copy-article n archive-name)))
```

訳注: See section “プロセス/接頭引数” in *The Gnus Manual*.

これにより、概略バッファで `M-x my-archive-article` を実行すれば、指定した記事を `nnml` グループに保存することができます。

もちろん、以下の設定によってキャッシュを常に有効にすることもできます：

```
(setq gnus-use-cache t)
```

これによって、維持したい記事には可視 (ticked) か保留 (dormant) の印を付けるだけで済むようになります。また、既読 (read) の印を付けることによって、それらの記事はキャッシュから削除されます。

質問 6.3

指定したメッセージを探すにはどうしたら良いですか？

回答

これにも複数の方法があります。Usenet グループに投稿されたものについては、おそらく `groups.google.com` (<http://groups.google.com>) に尋ねるのが最も楽な解決方法です。そこで見つかったならば、Google に生の記事を表示してもらって Message-ID を探し、概略バッファで `M-^ the@message.id RET` を実行して下さい。さらに Gnus 5.10 以降では 'groups.google.com' へのインターフェースがあるので、グループバッファで `G W` を使うこともできます。

メールとニュースの両方のグループで動作するもう一つの方法は、探しているメッセージが存在するグループに入って、Emacs の標準の探索コマンドである `C-s` を使うことです。それは壊れたスレッドにある記事を探すためにも十分に賢いものです。本文の中で探したいのならば、代わりに `M-s` を試して下さい。さらに付け加えると、これも役に立つ `gnus-summary-limit-to-*` コマンド群があります。

もちろんローカルに保存されているメールを 'grep' で探すこともできますが、大きなアーカイブが相手だと遅いし、見つかったメールを Gnus で表示するわけではないので不便でもあります。そこで `nnir` の出番がやってきました。`nnir` は 'swish-e', 'swish++' および他の検索エンジンへのフロントエンドです。それらの検索エンジンの一つを使ってメールに索引を付けておけば、`nnir` の助けを借りることによって、索引が付けられたメールを探して、検索の基準に合致するすべてのメールを含む一時的なグループを作ることができます。これがクールに聞こえるなら、<ftp://ls6-ftp.cs.uni-dortmund.de/pub/src/emacs/> または <ftp://ftp.is.informatik.uni-duisburg.de/pub/src/emacs/> から 'nnir.el' を持って来ましょう。使い方の説明は、そのファイルの始めの方にあります。

質問 6.4

要らなくなった古いメールを削除するにはどうすれば良いですか？

回答

もはや要らなくなったメールにポイントを置いて # で印を付けてから、それらを `B DEL` で永久に消してしまうことは、もちろんできます。さらに、それらを実際に消してしまう代わりに `B m nnml:trash-bin` を使ってジャンク・グループ (それは時々消さなければなりません) に送ることもできるのですが、両方とも Gnus が意図するやり方ではありません。

Gnus では、ニュースサーバーでニュース記事が期限切れ消去されるように、メールも消します。そうするためには、概略バッファのそのメールの上で `E` を叩くことによって、そのメールが期限切れ消去可能であることを (「このメールはもう要らないよ」と) Gnus に伝えれば良いのです。すると、そのグループを抜け出たときに、Gnus は以前に印が付けられたすべてのメッセージを検査して、十分に古くなった (デフォルトでは一週間より古くなった) メールを消去します。

質問 6.5

読み終わったすべてのメッセージを期限切れ消去したいのですが (少なくともいくつかのグループで)、どうしたら良いですか?

回答

読み終わったすべてのメッセージが期限切れ消去されるようにしたいなら (例えばオンラインのアーカイブが別に存在しているメーリングリストで)、auto-expire と total-expire という二つの選択肢があります。auto-expire というのは、印が付いていないけれども読むために選択されたことがあるすべての記事に期限切れ消去可能の印が付けられることで、メッセージを読むたびに Gnus が E を叩いてくれるようなものです。total-expire は少し異なるやり方に従っていて、既読の印が付けられたすべての記事が期限切れ消去可能になります。

auto-expire を有効にするには、そのグループのグループパラメーターに auto-expire を含めて下さい (グループバッファでグループパラメーターを変更するグループの上にポイントを置いて G c を叩きます)。total-expire の方は、グループパラメーターに total-expire を加えて下さい。

どちらの手段を選ぶかは、単に好みの問題です。auto-expire は速いのですが、適応スコア付けと共存できないので、この機能を使いたい場合は total-expire を使わなければなりません。

total-expire か auto-expire が設定されているグループで、あるメッセージを期限切れ消去の対象から外したい場合には、u で可視 (ticked) の印を付けるか ? で保留 (dormant) の印を付けて下さい。auto-expire を使っている場合は、さらに d で既読の印を付けることができます。

質問 6.6

期限が来たメールを削除するのではなくて、他のグループに移動させたいのですが。

回答

‘~/gnus.el’ ファイルに、このようなものを書き込んで下さい:

```
(setq nnmail-expiry-target "nnml:expired")
```

(nnmail-expiry-target にグループによって異なる値を設定したい場合には、質問「いくつかの (例えばメールの) グループで、スレッド表示をさせなくすることはできますか? あるいは、いくつかのグループに固有の変数を設定することができますか?」(see [[4.10]], page 368) を参照して下さい。)

10.10.7 ダイアルアップ環境で Gnus を使う

質問 7.1

ネットに常時接続していないのですが、どうしたら接続する時間を最小限にできますか?

回答

二つのやり方があります。一つは Gnus エージェントを使うこと (see [[7.2]], page 381) で、もう一方はニュース記事とメールをローカルディスクに取り込むプログラムをインストールして、Gnus にそれらをローカルマシンから読んでもらう方法です。

二つ目のやり方で行きたいのなら、ニュース記事を取り込んでそれらを Gnus に渡すプログラム、同じことをメールに対して行なうプログラム、およびあなたが書いたメールを Gnus から受け取って、オンラインになったらそれらを送信するプログラムが必要です。

最初に Unix システムについて書きましょう。ニュース記事を取り込むための最も安易な解は、Leafnode (<http://www.leafnode.org/>) や sn (<http://infa.abo.fi/~patrik/sn/>) のよう

に小規模な NNTP サーバーを使うことです。もちろん inn (<http://www.isc.org/products/INN/>) のような本格的なニュースサーバーをインストールすることもできます。メールの取り込みを使うもので人気があるのは fetchmail (<http://www.catb.org/~esr/fetchmail/>) および getmail (<http://www.qcc.ca/~charlesc/software/getmail-3.0/>) です。それらにメールをディスクに書き込むように指示して下さい。Gnus はそこから読むことになります。最後ですがおそれるべきでないのはメールの送信です。これにはあらゆる MTA、例えば sendmail (<http://www.sendmail.org/>), postfix (<http://www.qmail.org/>), exim (<http://www.exim.org/>) または qmail (<http://www.qmail.org/>) を使うことができます。

Windows 小屋のためには Hamster (<http://www.tglsoft.de/>) を推します。それは小さくてフリーなオープンソースのプログラムで、遠隔サーバーからメールとニュースを取り込んで、NNTP および POP3 または IMAP のそれぞれを介して Gnus (あるいは他のメール/ニュースリーダー) に渡します。さらにそれは Gnus が送信するメールを受け取るための SMTP サーバーも含んでいます。

質問 7.2

ならば、そのエージェントに関係するものは何ですか？

回答

Gnus エージェントは Gnus の一部で、メールとニュースを取り込んでディスクに格納し、後でオフラインのときにそれらを読むことができますようにします。それは、言うならば、例えば Forte Agent のようなオフライン・ニュースリーダーの真似をします。まだ 5.8.8 か 5.9 を使っていてエージェントを使いたければ、以下の行を `~/gnus.el` ファイルに書き込んで下さい (これは 5.10 からデフォルトになっています):

```
(setq gnus-agent t)
```

そうしたら、数あるグループをローカルに格納することができるサーバーを選ばなければなりません。そのためには、サーバーバッファを開いて (グループバッファで `^` を押して) 下さい。そして、選んだサーバー名の場所にポイントを移動させましょう。最後に `J a` をタイプして、そのサーバーをエージェント化して下さい。もし間違ってしまったら、あるいは気が変わったら、`J r` をタイプすれば元に戻すことができます。終わったら `q` をタイプして、グループバッファに戻して下さい。次回エージェント化されたサーバーのグループに入るとヘッダーがディスクに格納され、その次の回にグループバッファに入ると、そこ (ディスク) から読むようになります。

質問 7.3

記事の本文もディスクに格納したいのですが、どうすれば良いですか？

回答

ある述語を満足させる記事の本文を自動的に取り込むように、エージェントに指示することができます。それは、グループバッファで `J c` 命令を使うことによって行くことができる、特別なバッファで行ないます。どの述語を使うことができるか、およびそれを正しく行なう方法に関する情報については Gnus のマニュアルを参照して下さい。

さらに、どの記事をディスクに格納するかを、手作業でエージェントに指示することもできます。それには二つのやり方があります:

1. 概略バッファで、`#` 命令を実行してエージェントに格納されるべき記事にプロセス印を付けてから、`J s` をタイプして下さい。
2. 概略バッファで、`@` 命令を実行して欲しい記事にダウンロード可能印 (`'`) を付けてから、`J u` をタイプして下さい。

どんな違いがあるのかって？ えーと、ダウンロード可能印が永久的なのに対して、プロセス印は概略バッファを抜け出たとたんに消される点です。実際、複数のグループでダウンロード可能印を設定して、それらすべての記事を（グループバッファで `J s` 命令を使うことによって）まとめて取り込むことができます。唯一の欠点は、エージェント化されているサーバーの、選択されたすべてのグループのヘッダーをも取り込んでしまうことです。ヘッダーの量にもよりますが、最初の取り込みには何時間もかかるかもしれません。

質問 7.4

オフラインのときに Gnus にメールやニュース記事を送信させないようにするには、どうしたら良いですか？

回答

現在の状態がオンライン (plugged) かオフライン (unplugged) かを、Gnus に言ってあげなければなりません。それ以外のもろもろのことは自動的にやってくれます。plugged と unplugged の切り替えは、グループバッファで `J j` 命令を実行することによって行なうことができます。Gnus を unplugged の状態で起動したいなら、`M-x gnus` の代わりに `M-x gnus-unplugged` を使って下さい。ただし、これが働くためにはエージェントを有効にしなければならないことに注意して下さい。

10.10.8 助けを得る

質問 8.1

Emacs の中で情報を探したり助けを求めるには、どうすれば良いですか？

回答

最初に立ち寄るべき場所は Gnus マニュアルです (`C-h i d m Gnus RET` で Gnus マニュアルを開いたら、メニューを眺め回すか `s` でテキスト全体を探して下さい)。一方 Emacs には標準のヘルプ・コマンドがあって、それには `C-h` ともう一つのキーを使います (`C-h ? ?` をタイプすると、利用可能なヘルプ・コマンドとそれらの意味が現れます)。さらに `M-x apropos-command` で利用可能なすべてのコマンドを、`'M-x apropos'` で存在する変数を探することができます。

質問 8.2

Gnus のマニュアルの中で、あることがら（例えば添付ファイル、PGP、MIME...）に関する情報を、何も見つけることができません。それらは書かれていないのですか？

回答

Gnus マニュアル以外に message, emacs-mime, sieve および pgg のマニュアルがあります。これらのパッケージは Gnus とともに配布され、Gnus によって使われますが、真に Gnus の核心部分ではないので別の info ファイルで文書化されています。あなたはそれらのマニュアルも覗いてみるべきでしょう。

質問 8.3

私はどのウェブサイトを知っていなければなりませんか。

回答

二つの最も重要なものは 公式 Gnus ウェブサイト (<http://www.gnus.org>) と、その姉妹サイトである my.gnus.org (MGO) (<http://my.gnus.org>) です。後者には Lisp の断片、HowTo、チュートリアル (未完成)、およびこの FAQ があります。

他にも興味深いサイトがあったら教えて下さい。

質問 8.4

どんなメーリングリストとニュースグループがありますか。

回答

Gnus の一般的な質問を取り扱う ‘gnu.emacs.gnus’ ニュースグループ (gmane.emacs.gnus.user (<http://dir.gmane.org/gmane.emacs.gnus.user>) でも利用可) と、Gnus の開発を扱っている ding メーリングリスト (ding@gnus.org) があります。‘ding’ リストは ‘news.gmane.org’ から NNTP を介して、および gmane.emacs.gnus.general (<http://dir.gmane.org/gmane.emacs.gnus.general>) から読むこともできます。

Big8 (訳注: comp, humanities, misc, news, rec, sci, soc, talk の八大ニュースグループ) にとどまっていたいのならば、‘news.software.newssreaders’ もまた、いくつかの Gnus のユーザーによって読まれています (でも、的確な助けを得ることができる見込みという点では、上記のグループの方がはるかに良いでしょう)。ドイツ語を話す人向けには ‘de.comm.software.gnus’ があります。

質問 8.5

バグはどこに報告すれば良いですか？

回答

`M-x gnus-bug` 命令を使うことによって gnus bug メーリングリスト (bugs@gnus.org) に送るメッセージを書き始めることができます。それにはあなたを助けやすくするための、あなたの環境に関する情報が添付されます。(訳注: 2005 年の暮れの時点では、このメーリングリストはまともに機能していません。)

質問 8.6

その場で即座に得られる助け (real-time help) が欲しいのですが、どこで見つかりますか？

回答

あなたの IRC を ‘irc.freenode.net’ の ‘#mygnus’ チャンネルに接続して下さい。

10.10.9 Gnus をチューンする

質問 9.1

Gnus の起動が本当に遅いのですが、どうしたら速くすることができますか？

回答

この原因は、Gnus が active ファイルを読む方法に関係していることがあり得ます。Gnus マニュアルの See section “アクティブファイル” in *The Gnus Manual* を参照して下さい。そこには、その処理を速くするために試してみる価値があることがらがあります。もう一つの案は、‘~/gnus.el’ ファイルをバイトコンパイルすること (`M-x byte-compile-file RET ~/gnus.el RET` を実行すること) です。最後に、‘~/gnus.el’ ファイルで require を使っているならば、それらを eval-after-load で置き換えることができますでしょう。それは対象のモジュールを起動時ではなく、必要になったときに読み込むようにします。あなたの ‘~/gnus.el’ ファイルに以下のようなものがある場合には、

```
(require 'message)
(add-to-list 'message-syntax-checks '(sender . disabled))
```

Gnus を起動したとたんに 'message.el' が読み込まれます。これを次のように置き換えると、

```
(eval-after-load "message"
  '(add-to-list 'message-syntax-checks '(sender . disabled)))
```

それは必要になったときに読み込まれるようになります。

訳注: 原典の例は不適切です。現在の Gnus は多くの主要なモジュールを起動時に load するようになっていて、例えば 'message.el' モジュールは '~/.gnus.el' ファイルを読み込む時点ですでに load されています。ですから少なくとも 'message.el' モジュールに対しては require も eval-after-load も使う必要がありません。実害はありませんが何の効果もありません。変数 message-syntax-checks に項目を追加する上記の例は、単に以下のように書けば十分です。

```
(add-to-list 'message-syntax-checks '(sender . disabled))
```

質問 9.2

グループに入るときの動作を速くするには、どうすれば良いですか？

回答

スピードの殺し屋は、nil 以外の値を設定された gnus-fetch-old-headers 変数です。ですからスピードが問題なのであれば、それをやらないで下さい。概略バッファの構築を速くするには、

```
(gnus-compile)
```

を '~/.gnus.el' ファイルの最後に書き込んで下さい。これによって Gnus は gnus-summary-line-format のようなものをバイトコンパイルします。(訳注: それを行なうと '~/.newsrsrc.eld' ファイルにバイナリーコードが書き込まれるので、非-ASCII テキストで表現されたトピック名などと共存できなくなる危険があります。)

それから、 '~/.emacs' に以下のようなものを書き込んで、gc-cons-threshold の値を増やしてみして下さい。

```
(setq gc-cons-threshold 3500000)
```

CJK (中国語/日本語/韓国語) の文字の幅を気にしないのならば、または最近の Emacs とともに Gnus 5.10 かそれより新しいものを使っているのならば、次の行を '~/.gnus.el' ファイルに入れましょう:

```
(setq gnus-use-correct-string-widths nil)
```

訳注: See section "Wide Characters" in *The Gnus Manual*, も参照して下さい。

(これら二つの提言をしてくれた Jesper harder さんに感謝します。)

最後に、まだ 5.8.8 か 5.9 を使っていて、概略バッファを構築する場合の速度に問題があることを経験するのであれば、絶対に 5.10 に上げるべきです。そこではそれを改善するための、並外れた作業が行なわれました。

質問 9.3

メールの送信が遅くなってしまいました。何が起きたのでしょうか？

回答

その理由は、gnus-message-archive-group 変数の設定によって、送信メッセージの控えを保存する場所や方法を変更したせいかもしれません。アーカイブグループの代わりに nnml グループを使って下さい。そうすれば正常な速度が戻ってくるはずです。

10.10.10 用語集

- ~/.gnus.el* ‘~/.gnus.el’ という用語が使われるとき、それは Gnus の設定ファイルを指します。それを ‘~/.gnus’ と呼ぶか、あるいは別名を指定しても構いません。
- Back End* Gnus の用語大系では、バックエンドは Gnus の核心部分と実際のサーバーの間の層に位置を占める仮想サーバーのことです。実際のサーバーとは、NNTP サーバー、POP3 サーバー、IMAP サーバー、または、何であれ「メッセージの取得」や「ヘッダーの取得」を行なうサーバーの機能を持つ関数のことです。
- Emacs* この FAQ で使われている Emacs という用語は、GNU Emacs か XEmacs のどちらかを指します。
- Message* この FAQ で「メッセージ」はそれがどんな種類のものでも、メール、または Usenet ニュースグループか他の何らかの特製のバックエンドに送られるニュース記事のどちらかを意味します。
- MUA* MUA は Mail User Agent の頭字語で、電子メールを読んだり書いたりするためのプログラムのことです。
- NUA* NUA は News User Agent の頭字語で、Usenet ニュースを読んだり書いたりするためのプログラムのことです。

11 GNU フリー文書利用許諾契約書

訳注: 非公式な日本語訳 (<http://www.opensource.jp/fdl/fdl.ja.html.euc-jp>) があります。

Appendix A GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque.”

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements." Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at

your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications." You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted

document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being list their titles, with the
Front-Cover Texts being list, and with the Back-Cover Texts being
list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

12 Index

\$

\$ 278

%

% 14, 217

(

(ding) archive 22

*

* 15, 83

.

.newsrsc 8

.newsrsc.el 8

.newsrsc.eld 8

/

/ 83

<

< 63

>

> 63

A

Access Control Lists 193

activating groups 40, 329

active file 10, 329

adapt file group parameter 25

adaptive scoring 237

admin-address 25

adopting articles 64

advertisements 84

agent 210

agent expiry 219

Agent Parameters 212

agent regeneration 220

ange-ftp 40

ANSI control sequences 87

archive group 22

archived messages 126

archiving mail 179

article 329

article backlog 73

article buffer 115

article caching 71

article customization 118

article emphasis 82

article expiry 163

article hiding 83

article history 50

article marking 56

article pre-fetch 70

article scrolling 50

article series 77

article signature 93

article threading 63

article ticking 56

article washing 85

article-de-quoted-unreadable 167

asterisk 83

asynchronous article fetching 70

attachments 94

attachments, selection via dired 297

authentication 138

authinfo 138

auto-expire 25

auto-save 10

B

Babyl 196

back end 328

backlog 73

backup files 175

backup of mail 179

banner 27, 84

batch scoring 230

Bayesian spam filtering, naive 292

BBDB whitelists, spam filtering 284

BBDB, spam filtering 284

binary groups 101

blackholes, spam filtering 285

blacklists, spam filtering 283

BNF 338

body 329

body split 158

bogofilter, spam filtering 286

'bogus' group 147

bogus groups 32, 329

bookmarks 57

bouncing mail 52

broken-reply-to 24

browsing servers 32

browsing the web 179

bugs 303, 333

button levels 91

buttons 89, 260

byte-compilation 258

C

caching	71
calendar	205
canceling articles	54
changing servers	8
characters in file names	298
charset	26
charsets	97
charter	40
child	330
ClariNet Briefs	22
click	260
coding system aliases	97
colophon	327
colors	258
comment	26
compatibility	302
compilation	258
composing mail	51
composing messages	123
composing news	53
contributors	305
control message	40
converting kill files	244
copy mail	104
Courier IMAP server	189, 190
cross-posting	110
crosspost	147, 190
crosspost mail	104
crossposting	53
crossposts	241
customizing	21
customizing nndiary	207
customizing threading	63

D

daemons	261
date	234
DCC	272
decays	247
decoding articles	77
dejanews	180
delayed sending	55
delete-file	156
deleting headers	115
demons	261
describing groups	41
diary	205
diary articles sorting	209
diary group parameters	209
diary headers generation	209
diary summary buffer line	208
diary summary line format	208
diary summary lines sorting	209
digest	330
digests	54
ding Gnus	301

ding mailing list	334
direct connection functions	140
directory groups	194
dired	296
disk space	332
display	25
display-time	258
documentation group	196
drafts	130
dribble file	10
duplicate mails	167

E

edebug	333
editing imap acls	193
Editing IMAP ACLs	193
elp	333
Emacs	304
Emacsen	304, 353
email based diary	205
email spam	269, 270, 271
emphasis	82
ephemeral groups	330
Eudora	166
excessive crossposting	53
exiting Gnus	33
exiting groups	108
expirable mark	57
expiring IMAP mail	189, 190, 192
expiring mail	25, 32, 35, 103, 163
expiry, in Gnus agent	219
expiry-target	25
expiry-wait	25
expunge	193
expunging	188, 189, 193
extending the spam elisp package	291

F

face	267
faces	258
fancy mail splitting	157
FAQ	40
fetching a group	265
fetching by Message-ID	99
file commands	42
file names	298
filtering approaches, spam	270
finding news	3
firewall	135
first time usage	4
follow up	328
followup	123
fonts	258
foreign	328
foreign groups	21, 133
foreign servers	32

format-time-string..... 92
 formatting variables..... 250
 forwarded messages..... 196
 functions..... 335
 fuzzy article gathering..... 64
 fuzzy matching..... 269

G

gateways..... 202
 Gcc..... 126
 gcc-self..... 24
 generating sieve script..... 42
 general customization..... 331
 global score files..... 242
 gmane..... 22, 180
 Gmane, spam reporting..... 285
 gnu.emacs.gnus..... 334
 gnus..... 3
 Gnus agent..... 210
 Gnus agent expiry..... 219
 Gnus agent regeneration..... 220
 Gnus unplugged..... 210
 Gnus utility functions..... 335
 Gnus versions..... 301
 gnus-activate-all-groups..... 40
 gnus-activate-foreign-newsgroups..... 23
 gnus-activate-level..... 20
 gnus-active..... 335
 gnus-adaptive-file-suffix..... 238
 gnus-adaptive-pretty-print..... 238
 gnus-adaptive-word-length-limit..... 238
 gnus-adaptive-word-minimum..... 239
 gnus-adaptive-word-no-group-words..... 239
 gnus-adaptive-word-syntax-table..... 239
 gnus-add-configuration..... 257
 gnus-add-current-to-buffer-list..... 335
 gnus-add-to-list..... 24, 123
 gnus-after-exiting-gnus-hook..... 33
 gnus-after-getting-new-news-hook..... 40
 gnus-agent..... 221
 gnus-agent-add-group..... 217
 gnus-agent-add-server..... 218
 gnus-agent-auto-agentize-methods..... 223
 gnus-agent-batch..... 224
 gnus-agent-cache..... 222
 gnus-agent-catchup..... 218
 gnus-agent-consider-all-articles..... 222
 gnus-agent-directory..... 221
 gnus-agent-enable-expiration..... 217
 gnus-agent-expire..... 219
 gnus-agent-expire-all..... 220
 gnus-agent-expire-days..... 217, 219
 gnus-agent-expire-group..... 219
 gnus-agent-fetch-group..... 218
 gnus-agent-fetch-groups..... 217
 gnus-agent-fetch-session..... 217
 gnus-agent-fetched-hook..... 222

gnus-agent-go-online..... 222
 gnus-agent-handle-level..... 222
 gnus-agent-high-score..... 217
 gnus-agent-long-article..... 216
 gnus-agent-low-score..... 217
 gnus-agent-mark-article..... 217
 gnus-agent-mark-unread-after-downloaded..... 222
 gnus-agent-max-fetch-size..... 222
 gnus-agent-plugged-hook..... 222
 gnus-agent-prompt-send-queue..... 223
 gnus-agent-queue-mail..... 223
 gnus-agent-regenerate..... 220
 gnus-agent-regenerate-group..... 220
 gnus-agent-remove-group..... 217
 gnus-agent-remove-server..... 218
 gnus-agent-short-article..... 216
 gnus-agent-summary-fetch-group..... 218
 gnus-agent-summary-fetch-series..... 218
 gnus-agent-synchronize-flags.... 217, 220, 222
 gnus-agent-toggle-mark..... 218
 gnus-agent-toggle-plugged..... 217
 gnus-agent-unmark-article..... 217
 gnus-agent-unplugged-hook..... 222
 gnus-alter-articles-to-read-function.... 105
 gnus-alter-header-function..... 68
 gnus-always-force-window-configuration.. 257
 gnus-always-read-dribble-file..... 10
 gnus-ancient-mark..... 57
 gnus-apply-kill-file..... 244
 gnus-apply-kill-file-unless-scored..... 244
 gnus-apply-kill-hook..... 244
 gnus-article-add-buttons..... 88
 gnus-article-add-buttons-to-head..... 88
 gnus-article-address-banner-alist..... 84
 gnus-article-babel..... 93
 gnus-article-banner-alist..... 84
 gnus-article-boring-faces..... 50
 gnus-article-button-face..... 90
 gnus-article-capitalize-sentences..... 86
 gnus-article-date-english..... 92
 gnus-article-date-iso8601..... 91
 gnus-article-date-lapsed..... 92
 gnus-article-date-lapsed-new-header..... 92
 gnus-article-date-local..... 92
 gnus-article-date-original..... 92
 gnus-article-date-user..... 92
 gnus-article-date-ut..... 91
 gnus-article-de-base64-unreadable..... 87
 gnus-article-de-quoted-unreadable..... 86
 gnus-article-decode-charset..... 94
 gnus-article-decode-encoded-words..... 156
 gnus-article-decode-hook..... 121
 gnus-article-decode-HZ..... 87
 gnus-article-decode-mime-words..... 94
 gnus-article-describe-briefly..... 121
 gnus-article-display-face..... 92, 267
 gnus-article-display-x-face..... 92, 265

gnus-article-dumbquotes-map	86	gnus-article-sort-functions	70
gnus-article-emphasize	82	gnus-article-strip-all-blank-lines	88
gnus-article-emulate-mime	95	gnus-article-strip-banner	84
gnus-article-encrypt-body	105	gnus-article-strip-blank-lines	88
gnus-article-encrypt-protocol	105	gnus-article-strip-headers-in-body	88
gnus-article-fill-cited-article	86	gnus-article-strip-leading-blank-lines ...	88
gnus-article-fill-long-lines	86	gnus-article-strip-leading-space	88
gnus-article-followup-with-original	121	gnus-article-strip-multiple-blank-lines ..	88
gnus-article-hide	83	gnus-article-strip-trailing-space	88
gnus-article-hide-boring-headers	83, 115	gnus-article-time-format	92
gnus-article-hide-citation	84	gnus-article-treat-ansi-sequences	87
gnus-article-hide-citation-in-followups ..	85	gnus-article-treat-dumbquotes	86
gnus-article-hide-citation-maybe	84	gnus-article-treat-fold-headers	89
gnus-article-hide-headers	83	gnus-article-treat-fold-newsgroups	88
gnus-article-hide-list-identifiers	83	gnus-article-treat-overstrike	86
gnus-article-hide-pem	84	gnus-article-treat-types	119
gnus-article-hide-signature	83	gnus-article-treat-unfold-headers	88
gnus-article-highlight	81	gnus-article-unsplit-urls	87
gnus-article-highlight-citation	81	gnus-article-verify-x-pgp-sig	88
gnus-article-highlight-headers	81	gnus-article-wash-function	87
gnus-article-highlight-signature	82	gnus-article-wash-html	87
gnus-article-loose-mime	95	gnus-article-x-face-command	265
gnus-article-mail	121	gnus-article-x-face-too-ugly	265
gnus-article-maybe-highlight	81	gnus-async-prefetch-article-p	71
gnus-article-menu-hook	260	gnus-async-read-p	71
gnus-article-mime-part-function	95	gnus-asynchronous	71
gnus-article-mode-hook	121	gnus-auto-center-summary	49
gnus-article-mode-line-format	121	gnus-auto-expirable-newsgroups	163
gnus-article-mode-syntax-table	121	gnus-auto-extend-newsgroup	50
gnus-article-mouse-face	90	gnus-auto-goto-ignores	223
gnus-article-next-button	121	gnus-auto-select-first	17
gnus-article-next-page	120	gnus-auto-select-next	48
gnus-article-outlook-deuglify-article ...	86	gnus-auto-select-same	49
gnus-article-outlook-rearrange-citation ..	86	gnus-auto-select-subject	17
gnus-article-outlook-repair-attribution ..	86	gnus-auto-subscribed-groups	7
gnus-article-outlook-unwrap-lines	86	gnus-backup-startup-file	9
gnus-article-over-scroll	121	gnus-batch-score	230
gnus-article-prepare-hook	121	gnus-before-startup-hook	11
gnus-article-press-button	117	gnus-binary-mode	101
gnus-article-prev-button	121	gnus-binary-mode-hook	101
gnus-article-prev-page	121	gnus-binary-show-article	101
gnus-article-refer-article	121	gnus-body-boundary-delimiter	119
gnus-article-remove-cr	86	gnus-boring-article-headers	115
gnus-article-remove-images	93	gnus-break-pages	122
gnus-article-remove-leading-whitespace ...	89	gnus-browse-describe-briefly	33
gnus-article-remove-trailing-blank-lines ..	88	gnus-browse-describe-group	33
gnus-article-reply-with-original	121	gnus-browse-exit	32
gnus-article-save-directory	75	gnus-browse-menu-hook	260
gnus-article-show-summary	121	gnus-browse-mode	32
gnus-article-skip-boring	50	gnus-browse-read-group	32
gnus-article-sort-by-author	70	gnus-browse-select-group	32
gnus-article-sort-by-date	70	gnus-browse-unsubscribe-current-group ...	32
gnus-article-sort-by-number	70	gnus-buffer-configuration	254
gnus-article-sort-by-random	70	gnus-bug	303, 333
gnus-article-sort-by-schedule	209	gnus-bug-create-help-buffer	303
gnus-article-sort-by-score	70	gnus-build-sparse-threads	66
gnus-article-sort-by-subject	70	gnus-button-alist	89
		gnus-button-browse-level	91

gnus-button-ctan-handler	90	gnus-cite-parse-max-size	82
gnus-button-emacs-level	91	gnus-cited-closed-text-button-line-format	84
gnus-button-man-handler	89, 90	gnus-cited-lines-visible	84
gnus-button-man-level	91	gnus-cited-opened-text-button-line-format	84
gnus-button-message-level	91	gnus-compile	258
gnus-button-mid-or-mail-heuristic	90	gnus-configure-frame	256
gnus-button-mid-or-mail-regexp	90	gnus-confirm-mail-reply-to-news	123
gnus-button-prefer-mid-or-mail	90	gnus-confirm-treat-mail-like-news	123
gnus-button-tex-level	91	gnus-continuum-version	335
gnus-button-url-regexp	90	gnus-convert-image-to-face-command	267
gnus-buttonized-mime-types	95	gnus-convert-image-to-x-face-command	266
gnus-cache-active-file	72	gnus-convert-pbm-to-x-face-command	266
gnus-cache-directory	71	gnus-convert-png-to-face	267
gnus-cache-enter-article	73	gnus-crosspost-complaint	53
gnus-cache-enter-articles	72	gnus-ctan-url	90
gnus-cache-generate-active	72	gnus-current-home-score-file	240
gnus-cache-generate-nov-databases	72	gnus-current-prefix-symbol	352
gnus-cache-move-cache	72	gnus-current-prefix-symbols	352
gnus-cache-remove-article	73	gnus-dead-summary-mode	109
gnus-cache-remove-articles	72	gnus-decay-score	247
gnus-cacheable-groups	72	gnus-decay-score-function	247
gnus-cached-mark	57	gnus-decay-scores	247
gnus-canceled-mark	57	gnus-declare-backend	347
gnus-carpal	260	gnus-default-adaptive-score-alist	237
gnus-carpal-browse-buffer-buttons	260	gnus-default-adaptive-word-score-alist	238
gnus-carpal-button-face	260	gnus-default-article-saver	74
gnus-carpal-group-buffer-buttons	260	gnus-default-directory	297
gnus-carpal-header-face	260	gnus-default-ignored-adaptive-words	238
gnus-carpal-mode-hook	260	gnus-default-subscribed-newsgroups	4
gnus-carpal-server-buffer-buttons	260	gnus-del-mark	57
gnus-carpal-summary-buffer-buttons	260	gnus-delay-article	55
gnus-catchup-mark	57	gnus-delay-default-delay	55
gnus-category-add	216	gnus-delay-default-hour	55
gnus-category-copy	216	gnus-delay-group	55
gnus-category-customize-category	216	gnus-delay-header	56
gnus-category-edit-groups	216	gnus-delay-initialize	56
gnus-category-edit-predicate	216	gnus-delay-send-queue	56
gnus-category-edit-score	216	gnus-demon-add-disconnection	261
gnus-category-exit	216	gnus-demon-add-handler	261
gnus-category-kill	216	gnus-demon-add-nocem	261
gnus-category-line-format	216	gnus-demon-add-rescan	261
gnus-category-list	216	gnus-demon-add-scan-timestamps	261
gnus-category-mode-hook	216	gnus-demon-add-scanmail	261
gnus-category-mode-line-format	216	gnus-demon-cancel	261
gnus-change-server	8	gnus-demon-handlers	261
gnus-check-backend-function	336	gnus-demon-init	261
gnus-check-bogus-newsgroups	11	gnus-demon-timestep	261
gnus-check-new-newsgroups	5	gnus-diary	208
gnus-cite-attribution-face	82	gnus-diary-check-message	209
gnus-cite-attribution-prefix	82	gnus-diary-delay-format-function	209
gnus-cite-attribution-suffix	82	gnus-diary-summary-line-format	208
gnus-cite-face-list	82	gnus-diary-time-format	208
gnus-cite-hide-absolute	85	gnus-directory	297
gnus-cite-hide-percentage	85	gnus-dired-attach	297
gnus-cite-ignore-quoted-from	82	gnus-dired-find-file-mailcap	297
gnus-cite-max-prefix	82	gnus-dired-print	297
gnus-cite-minimum-match-count	82		

gnus-display-mime	116	gnus-group-brew-soup	200
gnus-display-mime-function	116	gnus-group-browse-foreign-server	4, 32
gnus-dormant-mark	56	gnus-group-catchup-current	19
gnus-downloadable-mark	58	gnus-group-catchup-current-all	19
gnus-downloaded-mark	58	gnus-group-catchup-group-hook	19
gnus-draft-edit-message	131	gnus-group-charset-alist	97
gnus-draft-send-all-messages	131	gnus-group-charter-alist	40
gnus-draft-send-message	131	gnus-group-check-bogus-groups	32
gnus-draft-toggle-sending	131	gnus-group-clear-data	8, 19
gnus-dribble-directory	10	gnus-group-clear-data-on-native-groups	8, 19
gnus-duplicate-file	111	gnus-group-compact-group	39
gnus-duplicate-list-length	111	gnus-group-customize	21
gnus-duplicate-mark	57	gnus-group-default-list-level	20
gnus-emphasis-alist	82	gnus-group-delete-group	22
gnus-emphasis-bold	83	gnus-group-describe-all-groups	41
gnus-emphasis-bold-italic	83	gnus-group-describe-briefly	41
gnus-emphasis-italic	83	gnus-group-describe-group	41
gnus-emphasis-underline	83	gnus-group-description-apropos	30
gnus-emphasis-underline-bold	83	gnus-group-edit-global-kill	244
gnus-emphasis-underline-bold-italic	83	gnus-group-edit-group	22
gnus-emphasis-underline-italic	83	gnus-group-edit-group-method	21
gnus-empty-thread-mark	58	gnus-group-edit-group-parameters	22
gnus-enter-category-buffer	217	gnus-group-edit-local-kill	244
gnus-ephemeral-group-p	335	gnus-group-enter-directory	22
gnus-exit-gnus-hook	33	gnus-group-enter-server-mode	39
gnus-exit-group-hook	109	gnus-group-exit	33
gnus-expert-user	249	gnus-group-expire-all-groups	32
gnus-expirable-mark	57	gnus-group-expire-articles	32
gnus-extra-header	46	gnus-group-faq-directory	40, 107
gnus-extra-headers	46	gnus-group-fetch-charter	40
gnus-extract-address-components	43	gnus-group-fetch-control	40
gnus-face-from-file	267	gnus-group-fetch-control-use-browse-url	40
gnus-face-properties-alist	266	gnus-group-fetch-faq	40
gnus-fetch-group	265	gnus-group-find-new-groups	32
gnus-fetch-old-ephemeral-headers	66	gnus-group-find-parameter	336
gnus-fetch-old-headers	66	gnus-group-first-unread-group	17
gnus-file-save-name	74	gnus-group-foreign-p	336
gnus-find-method-for-group	335	gnus-group-get-new-news	40
gnus-find-subscribed-addresses	24	gnus-group-get-new-news-this-group	40
gnus-folder-save-name	75	gnus-group-goto-unread	17
gnus-Folder-save-name	75	gnus-group-ham-exit-processor-BBDB	285
gnus-forwarded-mark	57	gnus-group-ham-exit-processor-bogofilter	287
gnus-gather-threads-by-references	66	gnus-group-ham-exit-processor-spamoracle	290
gnus-gather-threads-by-subject	66	gnus-group-ham-exit-processor-stat	289
gnus-gcc-externalize-attachments	128	gnus-group-ham-exit-processor-whitelist	284
gnus-gcc-mark-as-read	128	gnus-group-highlight	15
gnus-generate-horizontal-tree	102	gnus-group-highlight-line	16
gnus-generate-tree-function	102	gnus-group-highlight-words-alist	83
gnus-generate-vertical-tree	102	gnus-group-jump-to-group	16
gnus-get-info	335	gnus-group-kill-all-zombies	18
gnus-get-new-news-hook	40	gnus-group-kill-group	18
gnus-global-score-files	242	gnus-group-kill-level	18
gnus-goto-colon	253	gnus-group-kill-region	18
gnus-goto-next-group-when-activating	40	gnus-group-line-format	13
gnus-group-add-to-virtual	23		
gnus-group-apropos	30		
gnus-group-archive-directory	22		
gnus-group-best-unread-group	16		

gnus-group-list-active	30	gnus-group-recent-archive-directory	22
gnus-group-list-all-groups	29	gnus-group-rename-group	21
gnus-group-list-all-matching	30	gnus-group-restart	40
gnus-group-list-cached	30	gnus-group-save-newsrc	42
gnus-group-list-dormant	30	gnus-group-secondary-p	336
gnus-group-list-flush	30	gnus-group-select-group	17
gnus-group-list-groups	29	gnus-group-select-group-ephemerally	17
gnus-group-list-inactive-groups	20	gnus-group-send-queue	217
gnus-group-list-killed	29	gnus-group-set-current-level	19
gnus-group-list-level	29	gnus-group-set-parameter	336
gnus-group-list-limit	30	gnus-group-sort-by-alphabet	30
gnus-group-list-matching	30	gnus-group-sort-by-level	30
gnus-group-list-plus	30	gnus-group-sort-by-method	31
gnus-group-list-zombies	30	gnus-group-sort-by-rank	31
gnus-group-mail	39	gnus-group-sort-by-real-name	30
gnus-group-make-archive-group	22	gnus-group-sort-by-score	30
gnus-group-make-directory-group	22	gnus-group-sort-by-server	31
gnus-group-make-doc-group	22	gnus-group-sort-by-unread	31
gnus-group-make-empty-virtual	23	gnus-group-sort-function	30
gnus-group-make-group	21	gnus-group-sort-groups	30
gnus-group-make-help-group	22	gnus-group-sort-groups-by-alphabet	31
gnus-group-make-kiboze-group	22	gnus-group-sort-groups-by-level	31
gnus-group-make-rss-group	22	gnus-group-sort-groups-by-method	31
gnus-group-make-useful-group	22	gnus-group-sort-groups-by-rank	31
gnus-group-make-warchiver-group	182	gnus-group-sort-groups-by-real-name	31
gnus-group-make-web-group	22	gnus-group-sort-groups-by-score	31
gnus-group-mark-buffer	21	gnus-group-sort-groups-by-unread	31
gnus-group-mark-group	21	gnus-group-sort-selected-groups	32
gnus-group-mark-regexp	21	gnus-group-sort-selected-groups-by-alphabet	31
gnus-group-mark-region	21	gnus-group-sort-selected-groups-by-level	31
gnus-group-menu-hook	260	gnus-group-sort-selected-groups-by-method	32
gnus-group-mode-hook	39	gnus-group-sort-selected-groups-by-rank ..	32
gnus-group-mode-line-format	15	gnus-group-sort-selected-groups-by-real-name	32
gnus-group-move-group-to-server	8	gnus-group-sort-selected-groups-by-score	31
gnus-group-name-charset-group-alist	40	gnus-group-sort-selected-groups-by-unread	31
gnus-group-name-charset-method-alist	39	gnus-group-spam-exit-processor-blacklist	284
gnus-group-native-p	336	gnus-group-spam-exit-processor-bogofilter	287
gnus-group-news	39	gnus-group-spam-exit-processor-report-gmane	285
gnus-group-next-group	16, 32	gnus-group-spam-exit-processor-spamoracle	290
gnus-group-next-unread-group	16	gnus-group-spam-exit-processor-stat	289
gnus-group-next-unread-group-same-level ..	16	gnus-group-split	160
gnus-group-nnimap-edit-acl	193	gnus-group-split-default-catch-all-group	161
gnus-group-nnimap-expunge	193	gnus-group-split-fancy	161
gnus-group-no-more-groups-hook	108	gnus-group-split-setup	162
gnus-group-post-news	39	gnus-group-split-update	162
gnus-group-posting-charset-alist	97	gnus-group-split-updated-hook	162
gnus-group-prefixed-name	335	gnus-group-suspend	33
gnus-group-prepare-hook	39		
gnus-group-prev-group	16, 32		
gnus-group-prev-unread-group	16		
gnus-group-prev-unread-group-same-level ..	16		
gnus-group-quick-select-group	17		
gnus-group-quit	33		
gnus-group-read-ephemeral-group	21		
gnus-group-read-group	17		
gnus-group-read-init-file	42		
gnus-group-read-only-p	335		
gnus-group-real-name	335		

gnus-group-toolbar	269	gnus-kill-file-mode-hook	244
gnus-group-transpose-groups	18	gnus-kill-file-name	244
gnus-group-uncollapsed-levels	14	gnus-kill-files-directory	230
gnus-group-universal-argument	21	gnus-kill-killed	230
gnus-group-unmark-all-groups	21	gnus-kill-save-kill-file	244
gnus-group-unmark-group	21	gnus-kill-summary-on-exit	109
gnus-group-unread	335	gnus-killed-mark	57
gnus-group-unsubscribe-current-group	18	gnus-large-ephemeral-newsgroup	17
gnus-group-unsubscribe-group	18	gnus-large-newsgroup	17
gnus-group-update-hook	16	gnus-level-default-subscribed	20
gnus-group-use-permanent-levels	20	gnus-level-default-unsubscribed	20
gnus-group-visible-select-group	17	gnus-level-killed	19
gnus-group-yank-group	18	gnus-level-subscribed	19
gnus-ham-process-destinations	275, 279	gnus-level-unsubscribed	19
gnus-header-button-alist	89	gnus-level-zombie	19
gnus-header-face-alist	81	gnus-list-groups-with-ticked-articles	30
gnus-hidden-properties	298	gnus-list-identifiers	27, 83
gnus-hierarchical-home-score-file	240	gnus-load-hook	11
gnus-home-adapt-file	240	gnus-low-score-mark	57
gnus-home-directory	297	gnus-mail-save-name	74
gnus-home-score-file	239	gnus-mailing-list-archive	113
gnus-ignored-adaptive-words	238	gnus-mailing-list-groups	125
gnus-ignored-from-addresses	46	gnus-mailing-list-help	112
gnus-ignored-headers	115	gnus-mailing-list-insinuate	112
gnus-ignored-mime-types	95	gnus-mailing-list-mode	24
gnus-ignored-newsgroups	10	gnus-mailing-list-owner	113
gnus-info-find-node	41, 107	gnus-mailing-list-post	113
gnus-info-group	351	gnus-mailing-list-subscribe	112
gnus-info-level	351	gnus-mailing-list-unsubscribe	112
gnus-info-marks	352	gnus-make-predicate	264
gnus-info-method	352	gnus-mark-article-hook	50, 164
gnus-info-params	352	gnus-mark-unpicked-articles-as-read	101
gnus-info-rank	351	gnus-message-archive-group	126
gnus-info-read	352	gnus-message-archive-method	126
gnus-info-score	352	gnus-message-highlight-citation	125
gnus-info-set-group	351	gnus-message-replyencrypt	131
gnus-info-set-level	351	gnus-message-replysign	131
gnus-info-set-marks	352	gnus-message-replysignencrypted	131
gnus-info-set-method	352	gnus-mime-action-on-part	118
gnus-info-set-params	352	gnus-mime-copy-part	117
gnus-info-set-rank	351	gnus-mime-delete-part	117
gnus-info-set-read	352	gnus-mime-display-multipart-alternative-as- mixed	96
gnus-info-set-score	352	gnus-mime-display-multipart-as-mixed	96
gnus-inhibit-mime-unbuttonizing	95	gnus-mime-display-multipart-related-as- mixed	96
gnus-inhibit-slow-scoring	241	gnus-mime-inline-part	117
gnus-inhibit-startup-message	11	gnus-mime-multipart-functions	96
gnus-inhibit-user-auto-expire	165	gnus-mime-pipe-part	118
gnus-init-file	9, 42	gnus-mime-print-part	117
gnus-insert-pseudo-articles	81	gnus-mime-replace-part	117
gnus-insert-random-x-face-header	266	gnus-mime-save-part	117
gnus-interactive	352	gnus-mime-save-part-and-strip	117
gnus-interactive-catchup	249	gnus-mime-view-all-parts	94
gnus-interactive-exit	250	gnus-mime-view-part	117
gnus-invalid-group-regexp	298	gnus-mime-view-part-as-charset	117
gnus-jog-cache	72	gnus-mime-view-part-as-type	117
gnus-keep-backlog	73	gnus-mime-view-part-externally	118
gnus-keep-same-level	20		
gnus-kill-file-mark	57		

- gnus-mime-view-part-internally..... 117
- gnus-mode-non-string-length..... 258
- gnus-mouse-face..... 259
- gnus-move-split-methods..... 105
- gnus-narrow-to-body..... 336
- gnus-new-mail-mark..... 14
- gnus-news-group-p..... 335
- gnus-newsgroup-ignored-charsets..... 97
- gnus-newsgroup-name..... 335
- gnus-newsgroup-variables..... 106
- gnus-nntp-server..... 3
- gnus-nntpserver-file..... 3
- gnus-no-groups-message..... 11
- gnus-no-server..... 4
- gnus-nocem-check-article-limit..... 263
- gnus-nocem-check-from..... 263
- gnus-nocem-directory..... 263
- gnus-nocem-expiry-wait..... 263
- gnus-nocem-groups..... 262
- gnus-nocem-issuers..... 262
- gnus-nocem-verifier..... 263
- gnus-not-empty-thread-mark..... 58
- gnus-nov-is-evil..... 110
- gnus-novice-user..... 249
- gnus-numeric-save-name..... 74, 76
- gnus-Numeric-save-name..... 75
- gnus-options-not-subscribe..... 7
- gnus-options-subscribe..... 7
- gnus-other-frame..... 3
- gnus-outgoing-message-group..... 128
- gnus-outlook-deuglify-unwrap-max..... 86
- gnus-outlook-deuglify-unwrap-min..... 86
- gnus-page-delimiter..... 122
- gnus-parameters..... 28
- gnus-parameters-case-fold-search..... 28
- gnus-parse-headers-hook..... 68, 298
- gnus-part-display-hook..... 120
- gnus-permanently-visible-groups..... 30, 39
- gnus-pick-article-or-thread..... 100
- gnus-pick-display-summary..... 100
- gnus-pick-mode..... 100
- gnus-pick-mode-hook..... 101
- gnus-pick-next-page..... 100
- gnus-pick-start-reading..... 100
- gnus-pick-unmark-article-or-thread..... 100
- gnus-picon-databases..... 268
- gnus-picon-domain-directories..... 268
- gnus-picon-file-types..... 268
- gnus-picon-news-directories..... 268
- gnus-picon-style..... 268
- gnus-picon-user-directories..... 268
- gnus-plain-save-name..... 74, 76
- gnus-Plain-save-name..... 76
- gnus-play-startup-jingle..... 11
- gnus-post-method..... 123
- gnus-posting-styles..... 128
- gnus-prefetched-article-deletion-strategy..... 71
- gnus-preserve-marks..... 103
- gnus-process-mark..... 58
- gnus-prompt-before-saving..... 74
- gnus-ps-print-hook..... 98
- gnus-random-x-face..... 266
- gnus-read-active-file..... 10
- gnus-read-all-available-headers..... 66
- gnus-read-mark..... 57
- gnus-read-method..... 336
- gnus-read-newsrc-file..... 9
- gnus-recent-mark..... 58
- gnus-refer-article-method..... 99
- gnus-refer-thread-limit..... 99
- gnus-replied-mark..... 57
- gnus-rmail-save-name..... 74
- gnus-save-all-headers..... 73
- gnus-save-duplicate-list..... 111
- gnus-save-killed-list..... 9
- gnus-save-newsrc-file..... 9
- gnus-save-newsrc-hook..... 9
- gnus-save-quick-newsrc-hook..... 9
- gnus-save-score..... 230
- gnus-save-standard-newsrc-hook..... 9
- gnus-saved-headers..... 73
- gnus-saved-mark..... 58
- gnus-score-after-write-file-function.... 232
- gnus-score-below-mark..... 231
- gnus-score-change-score-file..... 227
- gnus-score-customize..... 228
- gnus-score-decay-constant..... 247
- gnus-score-decay-scale..... 247
- gnus-score-edit-all-score..... 229
- gnus-score-edit-current-scores..... 227
- gnus-score-edit-done..... 237
- gnus-score-edit-file..... 227
- gnus-score-edit-insert-date..... 237
- gnus-score-exact-adapt-limit..... 238
- gnus-score-expiry-days..... 232
- gnus-score-file-suffix..... 230
- gnus-score-find-bnews..... 231
- gnus-score-find-favourite-words..... 227
- gnus-score-find-hierarchical..... 231
- gnus-score-find-score-files-function.... 231
- gnus-score-find-single..... 231
- gnus-score-find-trace..... 227
- gnus-score-flush-cache..... 228, 229
- gnus-score-followup-article..... 240
- gnus-score-followup-thread..... 240
- gnus-score-interactive-default-score.... 230
- gnus-score-menu-hook..... 260
- gnus-score-mimic-keymap..... 229
- gnus-score-mode-hook..... 237
- gnus-score-over-mark..... 231
- gnus-score-pretty-print..... 237
- gnus-score-search-global-directories.... 242
- gnus-score-set-expunge-below..... 228
- gnus-score-set-mark-below..... 228
- gnus-score-thread-simplify..... 232

gnus-score-uncacheable-files	230	gnus-sort-gathered-threads-function	67
gnus-secondary-select-methods	4	gnus-sorted-header-list	115
gnus-secondary-servers	3	gnus-soup-add-article	200
gnus-select-article-hook	50	gnus-soup-directory	201
gnus-select-group-hook	17	gnus-soup-pack-packet	200
gnus-select-method	3	gnus-soup-packer	201
gnus-selected-tree-face	101	gnus-soup-packet-directory	201
gnus-sender-save-name	76	gnus-soup-packet-regexp	201
gnus-server-add-server	134	gnus-soup-prefix-file	201
gnus-server-close-all-servers	137	gnus-soup-replies-directory	201
gnus-server-close-server	137	gnus-soup-save-areas	200
gnus-server-compact-server	134	gnus-soup-send-replies	200
gnus-server-copy-server	134	gnus-soup-unpacker	201
gnus-server-deny-server	137	gnus-souped-mark	57
gnus-server-edit-server	134	gnus-spam-mark	278
gnus-server-equal	336	gnus-spam-newsgroup-contents	278
gnus-server-exit	134	gnus-spam-process-destinations	275, 280
gnus-server-kill-server	134	gnus-spam-process-newsgroups	278
gnus-server-line-format	133	gnus-sparse-mark	57
gnus-server-list-servers	134	gnus-split-methods	76
gnus-server-menu-hook	260	gnus-start-date-timer	92
gnus-server-mode-hook	133	gnus-started-hook	11
gnus-server-mode-line-format	134	gnus-startup-file	9
gnus-server-offline-server	137	gnus-startup-hook	11
gnus-server-open-all-servers	137	gnus-startup-jingle	11
gnus-server-open-server	137	gnus-stop-date-timer	92
gnus-server-read-server	134	gnus-subscribe-alphabetically	6
gnus-server-regenerate-server	134	gnus-subscribe-hierarchical-interactive ...	7
gnus-server-remove-denials	137	gnus-subscribe-hierarchically	6
gnus-server-scan-server	134	gnus-subscribe-interactively	7
gnus-server-to-method	336	gnus-subscribe-killed	7
gnus-server-unopen-status	223	gnus-subscribe-newsgroup-method	6
gnus-server-yank-server	134	gnus-subscribe-options-newsgroup-method ...	7
gnus-set-active	335	gnus-subscribe-randomly	6
gnus-setup-news-hook	11	gnus-subscribe-topics	7
gnus-shell-command-separator	298	gnus-subscribe-zombies	6
gnus-show-all-headers	115	gnus-sum-thread-tree-false-root	44
gnus-show-threads	67	gnus-sum-thread-tree-indent	45
gnus-sieve-crosspost	42	gnus-sum-thread-tree-leaf-with-other	45
gnus-sieve-file	42	gnus-sum-thread-tree-root	44
gnus-sieve-generate	42	gnus-sum-thread-tree-single-indent	44
gnus-sieve-region-end	42	gnus-sum-thread-tree-single-leaf	45
gnus-sieve-region-start	42	gnus-sum-thread-tree-vertical	45
gnus-sieve-update	42	gnus-summary-article-posted-p	104
gnus-signature-face	82	gnus-summary-beginning-of-article	51
gnus-signature-limit	93	gnus-summary-best-unread-article	50
gnus-signature-separator	82, 93	gnus-summary-bubble-group	20
gnus-simplify-all-whitespace	65	gnus-summary-caesar-message	85
gnus-simplify-ignored-prefixes	64	gnus-summary-cancel-article	54
gnus-simplify-subject-functions	65	gnus-summary-catchup	59
gnus-simplify-subject-fuzzy	65	gnus-summary-catchup-all	59
gnus-simplify-subject-fuzzy-regexp	64	gnus-summary-catchup-all-and-exit	109
gnus-simplify-subject-re	65	gnus-summary-catchup-and-exit	109
gnus-simplify-whitespace	65	gnus-summary-catchup-and-goto-next-group	109
gnus-single-article-buffer	121	gnus-summary-catchup-and-goto-prev-group	109
gnus-site-init-file	9	gnus-summary-catchup-from-here	59
gnus-slave	5		
gnus-smiley-file-types	268		

- gnus-summary-catchup-to-here 59
- gnus-summary-check-current 49
- gnus-summary-clear-above 59
- gnus-summary-clear-mark-forward 58
- gnus-summary-copy-article 104
- gnus-summary-create-article 104
- gnus-summary-crosspost-article 104
- gnus-summary-current-score 227
- gnus-summary-customize-parameters 108
- gnus-summary-default-score 230
- gnus-summary-delete-article 103
- gnus-summary-describe-briefly 107
- gnus-summary-describe-group 107
- gnus-summary-display-arrow 105
- gnus-summary-display-while-building 105
- gnus-summary-down-thread 69
- gnus-summary-dummy-line-format 64
- gnus-summary-edit-article 104
- gnus-summary-edit-article-done 104
- gnus-summary-edit-global-kill 244
- gnus-summary-edit-local-kill 244
- gnus-summary-edit-parameters 108
- gnus-summary-end-of-article 51
- gnus-summary-enter-digest-group 108
- gnus-summary-execute-command 107
- gnus-summary-exit 108
- gnus-summary-exit-hook 108
- gnus-summary-exit-no-update 109
- gnus-summary-expand-window 108
- gnus-summary-expire-articles 103
- gnus-summary-expire-articles-now 103
- gnus-summary-expunge-below 231
- gnus-summary-fetch-faq 107
- gnus-summary-first-unread-article 50
- gnus-summary-followup 53
- gnus-summary-followup-to-mail 54
- gnus-summary-followup-to-mail-with-original
..... 54
- gnus-summary-followup-with-original 53
- gnus-summary-force-verify-and-decrypt 88
- gnus-summary-gather-exclude-subject 65
- gnus-summary-gather-subject-limit 64
- gnus-summary-generate-hook 105
- gnus-summary-goto-article 50
- gnus-summary-goto-last-article 50
- gnus-summary-goto-subject 48
- gnus-summary-goto-unread 59, 249
- gnus-summary-hide-all-threads 69
- gnus-summary-hide-thread 68
- gnus-summary-highlight 48
- gnus-summary-idna-message 86
- gnus-summary-ignore-duplicates 105
- gnus-summary-import-article 104
- gnus-summary-increase-score 228
- gnus-summary-insert-cached-articles 107
- gnus-summary-insert-dormant-articles 108
- gnus-summary-insert-new-articles 62
- gnus-summary-insert-old-articles 62
- gnus-summary-insert-ticked-articles 108
- gnus-summary-isearch-article 51
- gnus-summary-kill-below 59
- gnus-summary-kill-process-mark 61
- gnus-summary-kill-same-subject 59
- gnus-summary-kill-same-subject-and-select
..... 59
- gnus-summary-kill-thread 68
- gnus-summary-limit-exclude-childless-
dormant 62
- gnus-summary-limit-exclude-dormant 62
- gnus-summary-limit-exclude-marks 62
- gnus-summary-limit-include-cached 62
- gnus-summary-limit-include-dormant 62
- gnus-summary-limit-include-expunged 62
- gnus-summary-limit-include-thread 62
- gnus-summary-limit-mark-excluded-as-read
..... 62
- gnus-summary-limit-to-age 62
- gnus-summary-limit-to-articles 62
- gnus-summary-limit-to-author 61
- gnus-summary-limit-to-bodies 63
- gnus-summary-limit-to-display-predicate .. 62
- gnus-summary-limit-to-extra 61
- gnus-summary-limit-to-headers 63
- gnus-summary-limit-to-marks 62
- gnus-summary-limit-to-recipient 61
- gnus-summary-limit-to-replied 62
- gnus-summary-limit-to-score 62
- gnus-summary-limit-to-singletons 61
- gnus-summary-limit-to-subject 61
- gnus-summary-limit-to-unread 61
- gnus-summary-limit-to-unseen 62
- gnus-summary-line-format 43, 46
- gnus-summary-lower-score 228
- gnus-summary-lower-thread 68
- gnus-summary-mail-crosspost-complaint 53
- gnus-summary-mail-forward 52
- gnus-summary-mail-other-window 52
- gnus-summary-mail-toolbar 269
- gnus-summary-make-false-root 63
- gnus-summary-make-false-root-always 64
- gnus-summary-mark-above 59
- gnus-summary-mark-as-dormant 58
- gnus-summary-mark-as-expirable 59
- gnus-summary-mark-as-processable 60
- gnus-summary-mark-as-read-backward 59
- gnus-summary-mark-as-read-forward 59
- gnus-summary-mark-as-spam 275
- gnus-summary-mark-below 227
- gnus-summary-mark-read-and-unread-as-read
..... 50
- gnus-summary-mark-region-as-read 59
- gnus-summary-mark-unread-as-read 50
- gnus-summary-menu-hook 260
- gnus-summary-mode-hook 105
- gnus-summary-mode-line-format 47
- gnus-summary-morse-message 85

gnus-summary-move-article	103	gnus-summary-rethread-current	69
gnus-summary-muttprint	74	gnus-summary-same-subject	43
gnus-summary-muttprint-program	74	gnus-summary-save-article	73
gnus-summary-news-other-window	52	gnus-summary-save-article-body-file	74
gnus-summary-next-article	49	gnus-summary-save-article-file	74
gnus-summary-next-group	109	gnus-summary-save-article-folder	74
gnus-summary-next-page	49, 50	gnus-summary-save-article-mail	73
gnus-summary-next-same-subject	50	gnus-summary-save-article-rmail	73
gnus-summary-next-thread	69	gnus-summary-save-article-vm	74
gnus-summary-next-unread-article	49	gnus-summary-save-body-in-file	75
gnus-summary-next-unread-subject	48	gnus-summary-save-in-file	74
gnus-summary-pick-line-format	101	gnus-summary-save-in-folder	75
gnus-summary-pipe-output	74	gnus-summary-save-in-mail	74
gnus-summary-pop-article	50	gnus-summary-save-in-rmail	74
gnus-summary-pop-limit	62	gnus-summary-save-in-vm	75
gnus-summary-post-forward	54	gnus-summary-save-newsrc	109
gnus-summary-post-news	53	gnus-summary-save-parts	94
gnus-summary-prepare	107	gnus-summary-save-process-mark	61
gnus-summary-prepare-exit-hook	108	gnus-summary-scroll-down	51
gnus-summary-prepare-hook	105	gnus-summary-scroll-up	51
gnus-summary-prepared-hook	105	gnus-summary-search-article-backward	107
gnus-summary-prev-article	49	gnus-summary-search-article-forward	107
gnus-summary-prev-group	109	gnus-summary-select-article-buffer	51
gnus-summary-prev-page	51	gnus-summary-selected-face	48
gnus-summary-prev-same-subject	50	gnus-summary-set-bookmark	59
gnus-summary-prev-thread	69	gnus-summary-set-score	227
gnus-summary-prev-unread-article	49	gnus-summary-show-all-threads	68
gnus-summary-prev-unread-subject	48	gnus-summary-show-article	51
gnus-summary-print-article	98	gnus-summary-show-article-charset-alist	51
gnus-summary-raise-thread	68	gnus-summary-show-thread	68
gnus-summary-read-document	108	gnus-summary-sort-by-author	98
gnus-summary-refer-article	99	gnus-summary-sort-by-chars	98
gnus-summary-refer-parent-article	99	gnus-summary-sort-by-date	98
gnus-summary-refer-references	99	gnus-summary-sort-by-lines	98
gnus-summary-refer-thread	99	gnus-summary-sort-by-number	98
gnus-summary-remove-bookmark	59	gnus-summary-sort-by-original	99
gnus-summary-repair-multipart	94	gnus-summary-sort-by-random	98
gnus-summary-reparent-children	69	gnus-summary-sort-by-recipient	98
gnus-summary-reparent-thread	69	gnus-summary-sort-by-schedule	209
gnus-summary-repeat-search-article-backward	107	gnus-summary-sort-by-score	98
gnus-summary-repeat-search-article-forward	107	gnus-summary-sort-by-subject	98
gnus-summary-reply	51	gnus-summary-stop-page-breaking	85
gnus-summary-reply-broken-reply-to	52	gnus-summary-supersede-article	55
gnus-summary-reply-broken-reply-to-with-original	52	gnus-summary-thread-gathering-function	65
gnus-summary-reply-with-original	51	gnus-summary-tick-above	59
gnus-summary-rescan-group	109	gnus-summary-tick-article-forward	58
gnus-summary-rescore	227	gnus-summary-toggle-display-buttonized	94
gnus-summary-reselect-current-group	109	gnus-summary-toggle-header	86
gnus-summary-resend-bounced-mail	52	gnus-summary-toggle-threads	68
gnus-summary-resend-message	53	gnus-summary-toggle-truncation	108
gnus-summary-resend-message-edit	53	gnus-summary-toolbar	269
gnus-summary-respool-article	104	gnus-summary-top-thread	69
gnus-summary-respool-default-method	104	gnus-summary-universal-argument	107
gnus-summary-respool-query	104	gnus-summary-unmark-all-processable	60
gnus-summary-respool-trace	104	gnus-summary-unmark-as-processable	60
		gnus-summary-up-thread	69
		gnus-summary-update-hook	48
		gnus-summary-verbose-headers	86

gnus-summary-very-wide-reply	52	gnus-topic-move-group	35
gnus-summary-very-wide-reply-with-original	52	gnus-topic-move-matching	35
gnus-summary-wake-up-the-dead	109	gnus-topic-remove-group	35
gnus-summary-wide-reply	51	gnus-topic-rename	35
gnus-summary-wide-reply-with-original	52	gnus-topic-select-group	34
gnus-summary-write-article-file	74	gnus-topic-show-topic	35
gnus-summary-write-body-to-file	75	gnus-topic-sort-groups	37
gnus-summary-write-to-file	75	gnus-topic-sort-groups-by-alphabet	36
gnus-summary-yank-message	54	gnus-topic-sort-groups-by-level	36
gnus-summary-yank-process-mark	61	gnus-topic-sort-groups-by-method	36
gnus-summary-zcore-fuzz	45	gnus-topic-sort-groups-by-rank	36
gnus-supercite-regexp	82	gnus-topic-sort-groups-by-score	36
gnus-supercite-secondary-regexp	82	gnus-topic-sort-groups-by-server	36
gnus-suppress-duplicates	111	gnus-topic-sort-groups-by-unread	36
gnus-suspend-gnus-hook	33	gnus-topic-toggle-display-empty-topics ...	35
gnus-symbolic-argument	250	gnus-topic-topology	37
gnus-thread-expunge-below	67	gnus-topic-unindent	34
gnus-thread-hide-killed	67	gnus-topic-unmark-topic	35
gnus-thread-hide-subtree	67	gnus-topic-yank-group	34
gnus-thread-ignore-subject	67	gnus-total-expirable-newsgroups	165
gnus-thread-indent-level	67	gnus-treat-display-face	120
gnus-thread-operation-ignore-subject	69	gnus-treat-display-x-face	119
gnus-thread-score-function	70	gnus-treat-emphasize	120
gnus-thread-sort-by-author	69	gnus-treat-fill-article	120
gnus-thread-sort-by-date	69	gnus-treat-fill-long-lines	120
gnus-thread-sort-by-date-reverse	69	gnus-treat-fold-headers	120
gnus-thread-sort-by-most-recent-date	69	gnus-treat-fold-newsgroups	120
gnus-thread-sort-by-most-recent-number ...	69	gnus-treat-from-picon	93
gnus-thread-sort-by-number	69	gnus-treat-hide-boring-headers	120
gnus-thread-sort-by-random	69	gnus-treat-hide-citation	120
gnus-thread-sort-by-recipient	69	gnus-treat-hide-citation-maybe	120
gnus-thread-sort-by-schedule	209	gnus-treat-hide-headers	120
gnus-thread-sort-by-score	69	gnus-treat-hide-signature	120
gnus-thread-sort-by-subject	69	gnus-treat-highlight-citation	120
gnus-thread-sort-by-total-score	69	gnus-treat-highlight-headers	120
gnus-thread-sort-functions	69	gnus-treat-highlight-signature	120
gnus-ticked-mark	56	gnus-treat-leading-whitespace	120
gnus-toolbar-thickness	269	gnus-treat-mail-picon	93
gnus-topic-copy-group	35	gnus-treat-newsgroups-picon	93
gnus-topic-copy-matching	35	gnus-treat-play-sounds	120
gnus-topic-create-topic	34	gnus-treat-smiley	92
gnus-topic-delete	35	gnus-treat-strip-banner	120
gnus-topic-display-empty-topics	36	gnus-treat-strip-list-identifiers	120
gnus-topic-edit-parameters	36	gnus-treat-translate	120
gnus-topic-expire-articles	35	gnus-treat-unfold-headers	120
gnus-topic-goto-next-topic	36	gnus-treat-x-pgp-sig	120
gnus-topic-goto-previous-topic	36	gnus-tree-brackets	102
gnus-topic-hide-topic	35	gnus-tree-line-format	101
gnus-topic-indent	34	gnus-tree-minimize-window	102
gnus-topic-indent-level	36	gnus-tree-mode-hook	101
gnus-topic-jump-to-topic	35	gnus-tree-mode-line-format	101
gnus-topic-kill-group	34	gnus-tree-parent-child-edges	102
gnus-topic-line-format	36	gnus-unbuttonized-mime-types	95
gnus-topic-list-active	35	gnus-uncacheable-groups	72
gnus-topic-mark-topic	35	gnus-undo	264
gnus-topic-mode	33	gnus-undo-mode	264
gnus-topic-mode-hook	36	gnus-undownloaded-mark	58
		gnus-unplugged	211

gnus-unread-mark	50, 57	gnus-uu-notify-files	78
gnus-unseen-mark	58	gnus-uu-post-include-before-composing	80
gnus-update-format	250	gnus-uu-post-length	80
gnus-update-score-entry-dates	232	gnus-uu-post-news	54
gnus-updated-mode-lines	258	gnus-uu-post-separate-description	80
gnus-use-adaptive-scoring	237	gnus-uu-post-threaded	80
gnus-use-article-prefetch	71	gnus-uu-pre-uudecode-hook	80
gnus-use-cache	71	gnus-uu-save-in-digest	80
gnus-use-correct-string-widths	254	gnus-uu-tmp-dir	79
gnus-use-cross-reference	109	gnus-uu-unmark-articles-not-decoded	80
gnus-use-dribble-file	10	gnus-uu-unmark-by-regexp	60
gnus-use-full-window	254	gnus-uu-unmark-region	60
gnus-use-idna	122	gnus-uu-unmark-thread	61, 68
gnus-use-long-file-name	71, 76	gnus-uu-user-archive-rules	79
gnus-use-nocem	262	gnus-uu-user-view-rules	79
gnus-use-scoring	230	gnus-uu-user-view-rules-end	79
gnus-use-toolbar	269	gnus-uu-view-and-save	79
gnus-use-trees	101	gnus-uu-view-with-metamail	80
gnus-use-undo	264	gnus-valid-select-methods	347
gnus-useful-groups	22	gnus-verbose	297
gnus-user-agent	125	gnus-verbose-backends	297
gnus-uu-be-dangerous	79	gnus-version	41
gnus-uu-correct-stripped-unicode	80	gnus-view-pseudo-asynchronously	81
gnus-uu-decode-binhex	78	gnus-view-pseudos	81
gnus-uu-decode-postscript	78	gnus-view-pseudos-separately	81
gnus-uu-decode-postscript-and-save	78	gnus-visible-headers	115
gnus-uu-decode-postscript-and-save-view ..	78	gnus-visual	259
gnus-uu-decode-postscript-view	78	gnus-visual-mark-article-hook	48
gnus-uu-decode-save	78	gnus-window-min-height	255
gnus-uu-decode-unshar	78	gnus-window-min-width	255
gnus-uu-decode-unshar-and-save	78	gnus-x-face	266
gnus-uu-decode-unshar-and-save-view	78	gnus-x-face-directory	266
gnus-uu-decode-unshar-view	78	gnus-x-face-from-file	266
gnus-uu-decode-uu	77	gnus-xmas-glyph-directory	269
gnus-uu-decode-uu-and-save	77	gnus-xmas-modeline-glyph	269
gnus-uu-decode-uu-and-save-view	77	Google	22, 180
gnus-uu-decode-uu-view	77	Graham, Paul	292
gnus-uu-digest-headers	123	group buffer	13
gnus-uu-digest-mail-forward	53	group buffer format	13
gnus-uu-digest-post-forward	54	group description	41
gnus-uu-do-not-unpack-archives	79	group highlighting	15
gnus-uu-grab-move	79	group information	40
gnus-uu-grab-view	79	group level	19
gnus-uu-grabbed-file-functions	79	group listing	29
gnus-uu-ignore-default-archive-rules	80	group mail splitting	160
gnus-uu-ignore-default-view-rules	79	group mode line	15
gnus-uu-ignore-files-by-name	79	group movement	16
gnus-uu-ignore-files-by-type	79	group parameters	23, 36
gnus-uu-invert-processable	60	group rank	20
gnus-uu-kill-carriage-return	80	group score	20
gnus-uu-mark-all	61	group score commands	229
gnus-uu-mark-buffer	61	group selection	17
gnus-uu-mark-by-regexp	60	group sieve commands	42
gnus-uu-mark-over	61	group timestamps	41
gnus-uu-mark-region	60		
gnus-uu-mark-series	61		
gnus-uu-mark-sparse	61		
gnus-uu-mark-thread	60, 68		

H

ham-marks	279
-----------------	-----

hashcash 273
 hashcash, spam filtering 285
hashcash-default-payment 274
hashcash-path 274
hashcash-payment-alist 274
 head 329
 header 329
 headers 329
 help group 22, 196
 hiding headers 115
 highlighting 15, 81, 259, 303
 highlights 297
 hilit19 303
 history 50, 301
 html 185
 http 179

I

IDNA 122
 ifile, spam filtering 288
 ignored groups 10
 ignored-charsets 26
 IMAP 185
 IMAP debugging 194
 IMAP namespace 193
imap-gssapi-program 187
imap-kerberos4-program 187
imap-log 194
imap-shell-host 188
imap-shell-program 188
imap-ssl-program 187
 import old mail 162
 importing PGP keys 112
 inbox 190
 incoming mail treatment 165
 Incoming* 305
 incorporating old mail 162
 indirect connection functions 141
 info 41
 information on groups 40
 installing under XEmacs 301
 interaction 249
 interactive 352
 internal variables 335
 internationalized domain names 122
 invalid characters in file names 298
 ISO 8601 91
 iso-8859-5 97
 ISO8601 234
 ispell 125
ispell-message 125

K

keys, reserved for users (Article) 120
 keys, reserved for users (Group) 38
 keys, reserved for users (Server) 134

keys, reserved for users (Summary) 43
 kibozing 204
 kill files 243, 244
 killed groups 329
 koi8-r 97
 koi8-u 97

L

Latin 1 86
 level 19
 levels 329
 limiting 61
 links 147
 LIST overview.fmt 110
 list server brain damage 165
 local variables 236
 loose threads 63

M

M****s*** sm*rtq**t*s 86
 mail 51, 123, 145, 328
 mail filtering (splitting) 147, 330
 mail folders 175
 mail group commands 103
 mail list groups 24
 mail message 329
 mail NOV spool 169
 mail server 148
 mail sorting 330
 mail source 148
 mail splitting 147, 157, 160
 mail spool 148
 mail washing 165
mail-extract-address-components 43
 Mail-Followup-To 24
mail-source-crash-box 154
mail-source-default-file-modes 155
mail-source-delete-incoming 155, 305
mail-source-delete-old-incoming-confirm
 155
mail-source-directory 155
mail-source-ignore-errors 155
mail-source-incoming-file-prefix 155
mail-source-movemail-program 155
mail-source-touch-pop 124
mail-sources 155
 mail-to-news gateways 202
 maildir 171
 mailing list 112
 mailing lists 125
 making digests 54
 making groups 21
 manual 41, 327
 manual expunging 193
 mark as unread 58
 marking groups 21

marks.....	56, 169, 175
max-lisp-eval-depth.....	333
mbox.....	196
mbox folders.....	175
menus.....	259
merging groups.....	203
message.....	329
Message-ID.....	99
message-mail-p.....	129
message-news-p.....	129
message-reply-headers.....	129
message-sent-hook.....	240
message-smtpmail-send-it.....	124
messages.....	123
metamail.....	80
MH folders.....	75
mh-e mail spool.....	170
MIME.....	116, 121, 304
MIME decoding.....	94
mm-decrypt-option.....	112
mm-file-name-collapse-whitespace.....	96
mm-file-name-delete-whitespace.....	96
mm-file-name-replace-whitespace.....	96
mm-file-name-rewrite-functions.....	96
mm-file-name-trim-whitespace.....	96
mm-verify-option.....	111
MMDF mail box.....	196
mml-secure-message-encrypt-pgp.....	132
mml-secure-message-encrypt-pgpmime.....	132
mml-secure-message-encrypt-smime.....	132
mml-secure-message-sign-pgp.....	131, 132
mml-secure-message-sign-smime.....	131
mml-unsecure-message.....	132
mml1991-use.....	112
mml2015-use.....	112
mode lines.....	258, 297
MODE READER.....	138
moderation.....	264
mouse.....	260
move mail.....	103
moving articles.....	105
Mule.....	304

N

naive Bayesian spam filtering.....	292
namespaces.....	193
native.....	328
Netscape.....	185
new features.....	307
new groups.....	5
new messages.....	40
news.....	328
news back ends.....	137
news spool.....	144
newsgroup.....	24
Newsgroups.....	46
nnbabyl.....	168

nnbabyl-active-file.....	168
nnbabyl-get-new-mail.....	168, 169
nnbabyl-mbox-file.....	168
nnchoke.....	337
nn diary.....	206
nn diary customization.....	207
nn diary mails.....	206
nn diary messages.....	206
nn diary operation modes.....	206
nn diary-mail-sources.....	207
nn diary-reminders.....	208
nn diary-split-methods.....	207
nn diary-week-starts-on-monday.....	208
nn dir.....	22, 194
nn doc.....	22, 196
nn doc-article-type.....	197
nn doc-post-type.....	197
nndraft.....	130
nndraft-directory.....	130
nneething.....	22, 195
nneething-exclude-files.....	195
nneething-include-files.....	195
nneething-map-file.....	196
nneething-map-file-directory.....	195
nn folder.....	175
nn folder-active-file.....	175
nn folder-delete-mail-hook.....	176
nn folder-directory.....	175
nn folder-generate-active-file.....	176
nn folder-get-new-mail.....	168, 175
nn folder-marks-directory.....	176
nn folder-marks-file-suffix.....	176
nn folder-marks-is-evil.....	176
nn folder-newsgroups-file.....	175
nn folder-nov-directory.....	176
nn folder-nov-file-suffix.....	176
nn folder-nov-is-evil.....	176
nn folder-save-buffer-hook.....	175
nngateway-address.....	202
nngateway-header-transformation.....	202
nngateway-mail2news-header-transformation.....	203
nngateway-simple-header-transformation.....	203
nnheader-file-name-translation-alist.....	298
nnheader-get-report.....	344
nnheader-head-chop-length.....	298
nnheader-max-head-length.....	297
nnheader-ms-strip-cr.....	166
nnheader-report.....	344
nnimap.....	185
nnimap-address.....	186
nnimap-authenticator.....	188
nnimap-authinfo-file.....	189
nnimap-expunge-on-close.....	188
nnimap-expunge-search-string.....	189
nnimap-importantize-dormant.....	189
nnimap-list-pattern.....	186

nnimap-need-unselect-to-notice-new-mail	189	nnmbox-active-file	168
nnimap-nov-is-evil	189	nnmbox-get-new-mail	168
nnimap-search-uids-not-since-is-evil	190	nnmbox-mbox-file	168
nnimap-server-port	186	nnmh	170
nnimap-split-crosspost	190	nnmh-be-safe	170
nnimap-split-download-body	192, 276	nnmh-directory	170
nnimap-split-fancy	192	nnmh-get-new-mail	168, 170
nnimap-split-inbox	190	nnml	169
nnimap-split-predicate	192	nnml-active-file	169
nnimap-split-rule	191	nnml-compressed-files-size-threshold	170
nnimap-stream	187	nnml-directory	169
nnkiboze	22, 204	nnml-generate-nov-databases	170
nnkiboze-directory	205	nnml-get-new-mail	168, 169
nnkiboze-generate-groups	205	nnml-marks-file-name	170
nnmail-cache-accepted-message-ids	157, 160	nnml-marks-is-evil	170
nnmail-cache-ignore-groups	157	nnml-newsgroups-file	169
nnmail-crosspost	147	nnml-nov-file-name	170
nnmail-crosspost-link-function	147	nnml-nov-is-evil	169
nnmail-delete-file-function	156	nnml-prepare-save-mail-hook	170
nnmail-expiry-target	164	nnml-use-compressed-files	170
nnmail-expiry-wait	164	nrrss	183
nnmail-expiry-wait-function	25, 164	nrrss-directory	183
nnmail-extra-headers	46	nrrss-file-coding-system	183
nnmail-fancy-expiry-target	165	nrrss-generate-download-script	184
nnmail-fancy-expiry-targets	165	nrrss-opml-export	183
nnmail-ignore-broken-references	166	nrrss-opml-import	183
nnmail-keep-last-article	165	nrrss-use-local	184
nnmail-mail-splitting-charset	148	nnslashdot	181
nnmail-mail-splitting-decodes	148	nnslashdot-active-url	182
nnmail-message-id-cache-file	167	nnslashdot-article-url	182
nnmail-message-id-cache-length	167	nnslashdot-comments-url	182
nnmail-post-get-new-mail-hook	156	nnslashdot-directory	182
nnmail-pre-get-new-mail-hook	156	nnslashdot-group-number	182
nnmail-prepare-incoming-header-hook	166	nnslashdot-login-name	181
nnmail-prepare-incoming-hook	166	nnslashdot-password	181
nnmail-prepare-incoming-message-hook	167	nnslashdot-threshold	182
nnmail-read-incoming-hook	156	nnsoup	201
nnmail-remove-leading-whitespace	166	nnsoup-active-file	202
nnmail-remove-list-identifiers	166	nnsoup-always-save	202
nnmail-remove-tabs	166	nnsoup-directory	201
nnmail-resplit-incoming	148, 149	nnsoup-pack-replies	200
nnmail-scan-directory-mail-source-once	149	nnsoup-packer	202
nnmail-split-abbrev-alist	159	nnsoup-packet-directory	202
nnmail-split-fancy	157	nnsoup-packet-regexp	202
nnmail-split-fancy-match-partial-words	159	nnsoup-replies-directory	201
nnmail-split-fancy-syntax-table	159	nnsoup-replies-format-type	201
nnmail-split-fancy-with-parent	159	nnsoup-replies-index-type	201
nnmail-split-header-length-limit	148	nnsoup-set-variables	202
nnmail-split-history	147	nnsoup-tmp-directory	201
nnmail-split-hook	156	nnsoup-unpacker	202
nnmail-split-lowercase-expanded	159	nnspool	144
nnmail-split-methods	147	nnspool-active-file	145
nnmail-spool-file	155	nnspool-active-times-file	145
nnmail-treat-duplicates	167	nnspool-history-file	145
nnmail-use-long-file-names	156	nnspool-inews-program	144
nnmaildir	171	nnspool-inews-switches	144
nnmbox	168	nnspool-lib-dir	145
		nnspool-newsgroups-file	145

nnsPOOL-nov-directory	145	nnwarchive-login	183
nnsPOOL-nov-is-evil	145	nnwarchive-passwd	183
nnsPOOL-sift-nov-with-sed	145	nnweb	22, 180
nnsPOOL-sPOOL-directory	144	nnweb-max-hits	180
nnTP	138	nnweb-search	180
nnTP authentication	138	nnweb-type	180
NNTP server	3	nnweb-type-definition	181
nnTP-address	143	No Gnus	301, 323
nnTP-authinfo-file	138	nocem	262
nnTP-authinfo-function	138	NOV	110, 139, 189, 329
nnTP-connection-timeout	139		
nnTP-end-of-line	143	O	
nnTP-marks-directory	144	offline	199, 210
nnTP-marks-is-evil	144	OneList	84
nnTP-maximum-request	139	Oort Gnus	301, 315
nnTP-never-echoes-commands	140	OPML	183
nnTP-nov-gap	139	Outlook Express	86
nnTP-nov-is-evil	139	overview.fmt	110
nnTP-open-connection-function	140		
nnTP-open-connection-functions-never-echo- commands	140	P	
nnTP-open-network-stream	140	parameters	36
nnTP-open-ssl-stream	141	parent	330
nnTP-open-telnet-stream	141	parent articles	99
nnTP-open-tls-stream	140	patches	333
nnTP-open-via-rlogin-and-netcat	142	Paul Graham	292
nnTP-open-via-rlogin-and-telnet	141	Pegasus	166
nnTP-open-via-telnet-and-telnet	142	persistent articles	72
nnTP-port-number	143	pgg-verify	263
nnTP-pre-command	143	PGP key ring import	112
nnTP-prepare-post-hook	140	pick and read	100
nnTP-prepare-server-hook	140	picons	92
nnTP-record-commands	140	POP	148
nnTP-send-authinfo	138	pop before smtp	124
nnTP-send-mode-reader	138	pop3-leave-mail-on-server	151
nnTP-server-action-alist	139	pop3-movemail	151
nnTP-server-opened-hook	138	post	53, 123
nnTP-telnet-command	143	post-method	26
nnTP-telnet-switches	143	posting styles	128
nnTP-via-address	143	posting-style	26
nnTP-via-envuser	143	PostScript	78, 98
nnTP-via-netcat-command	142	pre-fetch	70
nnTP-via-netcat-switches	142	predicate specifiers	264
nnTP-via-rlogin-command	142	preferred charset	97
nnTP-via-rlogin-command-switches	142	printing	98
nnTP-via-shell-prompt	143	process mark	57
nnTP-via-telnet-command	142	process/prefix convention	249
nnTP-via-telnet-switches	142	procmail	148
nnTP-via-user-name	143	profile	333
nnTP-via-user-password	143	protocol dump (IMAP)	194
nnTP-xover-commands	139	proxy	135
NNTPSERVER	3	pseudo-articles	81
nnultimate	182	Pterodactyl Gnus	301
nnultimate-directory	182		
nnvirtual	203	Q	
nnvirtual-always-rescan	204	Quassia Gnus	301
nnwarchive	182		
nnwarchive-directory	182		

R

rank	20
rcvstore	75
reading init file	42
reading mail	145
reading news	137
Red Gnus	301
referring articles	99
regeneration	220
regular expressions header matching, spam filtering	286
rejected articles	131
renaming groups	21
reply	123, 328
reporting bugs	303, 333
restarting	40
reverse scoring	242
RFC 1036	303
RFC 1522 decoding	156
RFC 1991	304
RFC 2047 decoding	156
RFC 2396	112
RFC 2440	304
RFC 2822	303
RFC 822	303
Rmail mbox	168, 196
rnews batch files	196
root	330
RSS	183
rule variables	78
running nndiary	206
Russian	97

S

saving .newsrsc	42
saving articles	73
scanning new news	40
score cache	230
score commands	227
score decays	247
score file atoms	235
score file format	232
score file group parameter	25
score variables	230
scoring	227
scoring crossposts	241
scoring on other headers	241
scoring tips	241
searching the Usenet	180
secondary	329
sed	145
select method	330
select methods	133
selecting articles	49
self contained nnfolder servers	175
self contained nnnml servers	169
Semi-gnus	302

send delayed	55
sending mail	123
sent messages	126
September Gnus	301
series	77
server	330
server buffer format	133
server commands	134
server errors	4
server parameters	136
server variables	136
setting marks	58
setting process marks	60
shared articles	78
shell archives	78
sieve	27
signatures	93
slash	83
Slashdot	181
slave	5
slow	333
slow machine	332
Smartquotes	86
smiley-data-directory	267
smiley-regexp-alist	267
smileys	92, 267
snarfing keys	112
solid groups	330
Son-of-RFC 1036	303
sorting groups	30
SOUP	199
sox	79
spam .. 262, 269, 271, 274, 276, 278, 280, 283, 284, 285, 286, 288, 289	
spam back ends	283
spam configuration examples	280
spam elisp package, extending	291
spam filtering ... 274, 276, 278, 280, 283, 284, 285, 286, 288, 289, 291	
spam filtering approaches	270
spam filtering configuration examples	280
spam filtering incoming mail	276
spam filtering sequence of events	274
spam filtering variables	278
spam filtering, naive Bayesian	292
spam reporting	285
spam variables	278
spam-autodetect-recheck-messages	280
spam-blackhole-good-server-regexp	286
spam-blackhole-servers	286
spam-bogofilter-database-directory	287
spam-bogofilter-score	287
spam-ifile-all-categories	288
spam-ifile-database	288
spam-ifile-spam-category	288
spam-initialize	274
spam-log-to-registry	280

spam-mark-ham-unread-before-move-from-spam-group	280	splitting, inbox	190
spam-mark-only-unseen-as-spam	280	splitting, rules	191
spam-marks	279	splitting, terminology	330
spam-process-ham-in-nonham-groups	279	spool	330
spam-process-ham-in-spam-groups	279	starting up	3
spam-regex-headers-ham	286	startup files	8
spam-regex-headers-spam	286	storing NNTP marks	144
spam-report-gmane-use-article-number	285	stripping advertisements	84
spam-report-user-mail-address	285	styles	128
spam-spamassassin-program	288	subscribed	24
spam-spamoracle-binary	290	subscription	5, 18
spam-spamoracle-database	290	summary buffer	43
spam-split-group	276	summary buffer format	43
spam-stat	289	summary exit	108
spam-stat	294	summary movement	48
spam-stat, spam filtering	289	summary sorting	98
spam-stat-buffer-change-to-non-spam	295	superseding articles	54
spam-stat-buffer-change-to-spam	295	symbolic prefixes	250
spam-stat-buffer-is-no-spam	295		
spam-stat-buffer-is-spam	295	T	
spam-stat-file	294	temporary groups	330
spam-stat-load	296	terminology	328
spam-stat-process-non-spam-directory	293	the Gnus diary library	208
spam-stat-process-spam-directory	293	the nndiary back end	206
spam-stat-reduce-size	294	thread commands	68
spam-stat-reset	294	thread root	330
spam-stat-save	294, 295	threading	63, 330
spam-stat-score-buffer	296	timestamps	41
spam-stat-score-word	296	To	46
spam-stat-split-fancy	296	to-address	23
spam-stat-split-fancy-spam-group	294	to-group	24
spam-use-BBDB	284	to-list	23
spam-use-BBDB-exclusive	284	topic commands	34
spam-use-blackholes	285	topic parameters	36, 37
spam-use-blacklist	283	topic sorting	36
spam-use-bogofilter	286	topic topology	37
spam-use-bogofilter-headers	287	topic variables	36
spam-use-dig	286	topics	33
spam-use-hashcash	285	topology	37
spam-use-ifile	288	total-expire	25
spam-use-regex-headers	286	transient-mark-mode	249
spam-use-spamassassin	288	trees	101
spam-use-spamassassin-headers	288	troubleshooting	333
spam-use-spamoracle	290		
spam-use-stat	274, 289	U	
spam-use-whitelist	283	UCE	269, 270, 271
spam-use-whitelist-exclusive	283	Ultimate Bulletin Board	182
SpamAssassin	272	underline	83
spamassassin, spam filtering	288	undo	264
spamming	110	unix mail box	168
SpamOracle	289	Unix mbox	196
sparse articles	330	unplugged	210
splitting	192	unshar	78
splitting imap mail	190	unsolicited commercial email	269, 270, 271
splitting mail	147	updating sieve script	42
splitting, crosspost	190	url	185
splitting, fancy	192		

USEFOR 303
 Usenet searches 180
 User-Agent 125
 using gpg 123, 131
 using s/mime 123, 131
 using smime 123, 131
 UTF-8 group names 40
 utility functions 335
 uuencode 77
 uuencode 95
 uuencoded articles 77

V

velveeta 110
 version 41
version-control 9
 viewing attachments 94
 viewing files 81
 Vipul's Razor 272
 virtual groups 203
 virtual server 330
 visible 24
 visible group parameter 30
 visual 259

W

W3 185
 washing 85, 330
 web 179
 Web Archive 182
 whitelists, spam filtering 283
 window height 255
 window layout 254
 window width 255
 www 179

X

x-face 92, 265
 X-Hashcash 273
 XEmacs 301, 304, 353
 XOVER 139
 Xref 110

Y

yEnc 95

Z

zombie groups 329

13 Key Index

!		/ v (概略)	62
! (概略)	58	/ w (概略)	62
		/ x (概略)	61
#		<	
# (概略)	60	< (概略)	51
# (グループ)	21		
&		=	
& (概略)	107	= (概略)	108
*		>	
* (概略)	73	> (概略)	51
,		?	
, (概略)	50	? (記事)	121
, (グループ)	16	? (概略)	58
		? (グループ)	41
.		? (閲覧)	33
. (選択)	100	@	
. (記事)	118	@ (エージェント 概略)	218
. (概略)	50	^	
. (グループ)	17	^ (概略)	99
/		^ (グループ)	39
/ * (概略)	62		
/ . (概略)	62	(記事)	118
/ / (概略)	61	(概略)	74
/ a (概略)	61		
/ b (概略)	63	A	
/ c (概略)	62	a (分類)	216
/ C (概略)	62	a (概略)	53
/ d (概略)	62	a (サーバー)	134
/ D (概略)	62	a (グループ)	39
/ E (概略)	62	A / (グループ)	30
/ h (概略)	63	A < (概略)	51
/ m (概略)	62	A > (概略)	51
/ M (概略)	62	A ? (グループ)	30
/ n (概略)	62	A a (グループ)	30
/ N (概略)	62	A A (グループ)	30
/ o (概略)	62	A c (グループ)	30
/ p (概略)	62	A D (概略)	108
/ r (概略)	62	A d (グループ)	30
/ R (概略)	61	A f (グループ)	30
/ S (概略)	61	A g (概略)	51
/ t (概略)	62		
/ T (概略)	62		
/ u (概略)	61		

A k (グループ)	29
A l (グループ)	29
A M (概略)	112
A m (グループ)	30
A M (グループ)	30
A P (概略)	98
A p (グループ)	30
A R (概略)	99
A s (概略)	51
A s (グループ)	29
A t (概略)	93
A T (概略)	99
A T (トピック)	35
A u (グループ)	29
A z (グループ)	30

B

b (概略)	94
b (グループ)	32
B (グループ)	4, 32
B B (概略)	104
B c (概略)	104
B C-M-e (概略)	103
B DEL (概略)	103
B e (概略)	103
B i (概略)	104
B I (概略)	104
B m (概略)	103
B p (概略)	104
B q (概略)	104
B r (概略)	104
B t (概略)	104
B w (概略)	104

C

c (記事)	117
C (記事)	117
c (分類)	216
c (概略)	109
C (概略)	54
c (サーバー)	134
C (サーバー)	137
c (グループ)	19
C (グループ)	19
C-c ^ (記事)	121
C-c C-c (記事)	104
C-c C-c (投稿)	123
C-c C-c (スコア)	237
C-c C-d (グループ)	41
C-c C-d (スコア)	237
C-c C-f (概略)	52
C-c C-i (グループ)	41
C-c C-m (記事)	121
C-c C-m c o (Message)	132
C-c C-m c p (Message)	132
C-c C-m c s (Message)	132

C-c C-m C-n (Message)	132
C-c C-m s o (Message)	131
C-c C-m s p (Message)	132
C-c C-m s s (Message)	131
C-c C-M-x (グループ)	32
C-c C-n a (概略)	113
C-c C-n h (概略)	112
C-c C-n o (概略)	113
C-c C-n p (概略)	113
C-c C-n s (概略)	112
C-c C-n u (概略)	112
C-c C-p (スコア)	237
C-c C-s (グループ)	30
C-c C-s C-a (概略)	98
C-c C-s C-c (概略)	98
C-c C-s C-d (概略)	98
C-c C-s C-i (概略)	98
C-c C-s C-l (概略)	98
C-c C-s C-n (概略)	98
C-c C-s C-o (概略)	99
C-c C-s C-r (概略)	98
C-c C-s C-s (概略)	98
C-c C-s C-t (概略)	98
C-c C-x (グループ)	32
C-c C-x (トピック)	35
C-c M-g (グループ)	40
C-d (概略)	108
C-k (概略)	59
C-k (グループ)	18
C-k (トピック)	34
C-M-a (概略)	108
C-M-b (概略)	69
C-M-d (概略)	108
C-M-e (概略)	108
C-M-f (概略)	69
C-M-k (概略)	68
C-M-l (概略)	68
C-M-RET (グループ)	17
C-o (記事)	117
C-t (概略)	108
C-w (概略)	59
C-w (グループ)	18
C-x C-s (概略)	109
C-x C-t (グループ)	18
C-y (グループ)	18
C-y (トピック)	34

D

d (記事)	117
d (概略)	59
D (概略)	59
D (サーバー)	137
d (閲覧)	33
D e (下書き)	131
D g (グループ)	42
D s (下書き)	131
D S (下書き)	131

D t (下書き)	131
D u (グループ)	42
DEL (記事)	121
DEL (概略)	51
DEL (グループ)	16

E

e (記事)	118
E (記事)	118
e (分類)	216
e (概略)	104
E (概略)	59
e (サーバー)	134

F

F (記事)	121
f (概略)	53
F (概略)	53
F (グループ)	32

G

g (分類)	216
g (概略)	51
g (サーバー)	134
g (グループ)	40
g (バイナリー)	101
G a (グループ)	22
G b (概略)	50
G c (グループ)	21
G C-n (概略)	50
G C-p (概略)	50
G d (グループ)	22
G D (グループ)	22
G DEL (グループ)	22
G e (グループ)	21
G E (グループ)	22
G f (概略)	50
G f (グループ)	22
G g (概略)	48
G h (グループ)	22
G j (概略)	50
G k (グループ)	22, 204
G l (概略)	50
G l (グループ)	193
G m (グループ)	21
G M (グループ)	21
G M-n (概略)	48
G M-p (概略)	48
G n (概略)	49
G N (概略)	49
G o (概略)	50
G P (概略)	49
G p (グループ)	22
G p (トピック)	36
G P a (グループ)	31

G P l (グループ)	31
G P m (グループ)	32
G P n (グループ)	32
G P r (グループ)	32
G P s (グループ)	32
G P u (グループ)	31
G P v (グループ)	31
G r (グループ)	21
G R (グループ)	22, 183
G S a (グループ)	31
G s b (グループ)	200
G S l (グループ)	31
G S m (グループ)	31
G S n (グループ)	31
G s p (グループ)	200
G s r (グループ)	200
G S r (グループ)	31
G s s (グループ)	200
G S u (グループ)	31
G S v (グループ)	31
G S w (グループ)	200
G u (グループ)	22
G v (グループ)	23
G V (グループ)	23
G w (グループ)	22
G x (グループ)	193
G z (グループ)	39
gnus-summary-raise-thread	68

H

h (概略)	51
H c (グループ)	40
H C (グループ)	40
H d (概略)	107
H d (グループ)	41
H f (概略)	107
H f (グループ)	40
H h (概略)	107
H i (概略)	107
H v (グループ)	41

I

i (記事)	117
i (概略)	52
i (グループ)	39

J

J # (エージェント 概略)	217
j (概略)	50
j (グループ)	16
J a (エージェント サーバー)	218
J a (エージェント グループ)	217
J c (エージェント 概略)	218
J c (エージェント グループ)	217
J j (エージェント)	217

J M-# (エージェント 概略)	217
J r (エージェント サーバー)	218
J r (エージェント グループ)	217
J s (エージェント 概略)	218
J S (エージェント 概略)	218
J s (エージェント グループ)	217
J S (エージェント グループ)	217
J u (エージェント 概略)	218
J u (エージェント グループ)	217
J Y (エージェント グループ)	217

K

k (分類)	216
k (概略)	59
k (サーバー)	134
K l (概略)	94
K b (概略)	94
K c (概略)	94
K d (概略)	94
K e (概略)	94
K E (概略)	105
K i (概略)	94
K m (概略)	94
K o (概略)	94
K O (概略)	94
K r (概略)	94
K v (概略)	94

L

l (分類)	216
l (概略)	50
l (サーバー)	134
L (サーバー)	137
l (グループ)	29
L (グループ)	29
l (閲覧)	32

M

m (概略)	52
m (グループ)	39
M ? (概略)	58
M b (概略)	59
M B (概略)	59
M b (グループ)	21
M c (概略)	58
M C (概略)	59
M C-c (概略)	59
M d (概略)	59
M e (概略)	59
M h (概略)	59
M H (概略)	59
M k (概略)	59
M K (概略)	59
M m (グループ)	21
M P a (概略)	61

M P b (概略)	61
M P g (概略)	60
M P G (概略)	60
M P i (概略)	60
M P k (概略)	61
M P p (概略)	60
M P r (概略)	60
M P R (概略)	60
M P s (概略)	61
M P S (概略)	61
M P t (概略)	60
M P T (概略)	61
M P u (概略)	60
M P U (概略)	60
M P v (概略)	61
M P w (概略)	61
M P y (概略)	61
M r (グループ)	21
M S (概略)	62
M s t	287
M s x	275
M t (概略)	58
M u (グループ)	21
M U (グループ)	21
M V c (概略)	59
M V k (概略)	59
M V m (概略)	59
M V u (概略)	59
M w (グループ)	21
M-# (概略)	60
M-# (グループ)	21
M-& (概略)	107
M-* (概略)	73
M-^ (概略)	99
M-c (サーバー)	137
M-c (グループ)	19
M-d	275
M-d (グループ)	41
M-down (概略)	69
M-g (概略)	109
M-g (グループ)	40
M-i (概略)	250
M-k (概略)	244
M-K (概略)	244
M-k (グループ)	244
M-K (グループ)	244
M-n (概略)	48
M-n (グループ)	16
M-o (サーバー)	137
M-p (概略)	48
M-p (グループ)	16
M-r (概略)	107
M-R (概略)	107
M-RET (記事)	117
M-RET (概略)	51
M-RET (グループ)	17
M-s (概略)	107
M-S (概略)	107

M-SPACE (グループ)	17
M-t (概略)	94
M-TAB (記事)	121
M-TAB (トピック)	34
M-u (概略)	58
M-up (概略)	69
M-x gnus	3
M-x gnus-agent-expire	219
M-x gnus-agent-expire-group	219
M-x gnus-agent-regenerate	220
M-x gnus-agent-regenerate-group	220
M-x gnus-binary-mode	101
M-x gnus-bug	303, 333
M-x gnus-change-server	8
M-x gnus-group-clear-data	8
M-x gnus-group-clear-data-on-native-groups	8, 19
M-x gnus-group-move-group-to-server	8
M-x gnus-no-server	4
M-x gnus-other-frame	3
M-x gnus-pick-mode	100
M-x gnus-update-format	250
M-x nnfolder-generate-active-file	176
M-x nnkiboze-generate-groups	205
M-x nnmail-split-history	147

N

n (概略)	49
N (概略)	49
n (グループ)	16
N (グループ)	16
n (閲覧)	32

O

o (記事)	117
o (概略)	73
O (サーバー)	137
O b (概略)	74
O f (概略)	74
O F (概略)	74
O h (概略)	74
O m (概略)	73
O o (概略)	73
O p (概略)	74
O P (概略)	74
O r (概略)	73
O s (概略)	200
O v (概略)	74

P

p (記事)	117
p (分類)	216
p (概略)	49
P (概略)	49
p (グループ)	16

P (グループ)	16
p (閲覧)	32

Q

q (分類)	216
q (概略)	108
Q (概略)	109
q (サーバー)	134
q (グループ)	33
Q (グループ)	33
q (閲覧)	32

R

r (記事)	117
R (記事)	121
r (概略)	51
R (概略)	51
R (サーバー)	137
r (グループ)	42
R (グループ)	40
RET (選択)	100
RET (記事)	117
RET (概略)	51
RET (グループ)	17
RET (トピック)	34
RET (閲覧)	32

S

s (記事)	121
s (分類)	216
s (概略)	51
S (概略)	55
s (サーバー)	134
s (グループ)	42
S B r (概略)	52
S B R (概略)	52
S C-k (グループ)	18
S D b (概略)	52
S D e (概略)	53
S D r (概略)	53
S f (概略)	53
S F (概略)	53
S i (概略)	52
S k (グループ)	18
S l (グループ)	19
S m (概略)	52
S M-c (概略)	53
S n (概略)	54
S N (概略)	54
S o m (概略)	52
S O m (概略)	53
S o p (概略)	54
S O p (概略)	54
S p (概略)	53
S r (概略)	51

S R (概略)	51
S s (グループ)	18
S t	287
S t (グループ)	18
S u (概略)	54
S v (概略)	52
S V (概略)	52
S w (概略)	51
S W (概略)	52
S w (グループ)	18
S x	275
S y (概略)	54
S y (グループ)	18
S z (グループ)	18
SPACE (選択)	100
SPACE (記事)	120
SPACE (概略)	49, 50
SPACE (サーバー)	134
SPACE (グループ)	17
SPACE (閲覧)	32

T

T # (概略)	68
T # (トピック)	35
t (記事)	117
t (概略)	86
t (グループ)	33
T ^ (概略)	69
T c (トピック)	35
T C (トピック)	35
T d (概略)	69
T D (トピック)	35
T DEL (トピック)	35
T h (概略)	68
T H (概略)	69
T h (トピック)	35
T H (トピック)	35
T i (概略)	68
T j (トピック)	35
T k (概略)	68
T l (概略)	68
T m (トピック)	35
T M (トピック)	35
T M-# (概略)	68
T M-# (トピック)	35
T M-^ (概略)	69
T M-n (トピック)	36
T M-p (トピック)	36
T n (概略)	69
T n (トピック)	34
T o (概略)	69
T p (概略)	69
T r (トピック)	35
T s (概略)	68
T S (概略)	68
T s (トピック)	35
T S a (トピック)	36

T S e (トピック)	36
T S l (トピック)	36
T S m (トピック)	36
T S r (トピック)	36
T S s (トピック)	37
T S u (トピック)	36
T S v (トピック)	36
T t (概略)	69
T T (概略)	68
T TAB (トピック)	34
T u (概略)	69
TAB (記事)	121
TAB (トピック)	34

U

u (選択)	100
u (グループ)	18
U (グループ)	18
u (閲覧)	32

V

v (記事)	120
v (概略)	43
v (サーバー)	134
v (グループ)	38
V (グループ)	41
V c (概略)	227
V C (概略)	228
V e (概略)	227
V f (概略)	227
V F (概略)	228
V m (概略)	228
V R (概略)	227
V s (概略)	227
V S (概略)	227
V t (概略)	227
V w (概略)	227
V x (概略)	228

W

W 6 (概略)	87
W a (概略)	88
W A (概略)	87
W b (概略)	88
W B (概略)	88
W c (概略)	86
W C (概略)	86
W d (概略)	86
W D d (概略)	92
W D D (概略)	93
W D f (概略)	93
W D m (概略)	93
W D n (概略)	93
W D s (概略)	92
W D x (概略)	92

W e (概略)	82	W W s (概略)	83
W e (グループ)	229	W Y a (概略)	86
W E a (概略)	88	W Y c (概略)	86
W E A (概略)	88	W Y f (概略)	86
W E e (概略)	88	W Y u (概略)	86
W E l (概略)	88	W Z (概略)	87
W E m (概略)	88		
W E s (概略)	88	X	
W E t (概略)	88	x (概略)	61
W E w (概略)	89	X b (概略)	78
W f (グループ)	229	X m (概略)	94
W G f (概略)	89	X o (概略)	78
W G n (概略)	88	X p (概略)	78
W G u (概略)	88	X P (概略)	78
W h (概略)	87	X s (概略)	78
W H a (概略)	81	X S (概略)	78
W H c (概略)	81	X u (概略)	77
W H h (概略)	81	X U (概略)	77
W H s (概略)	82	X v p (概略)	78
W i (概略)	86	X v P (概略)	78
W l (概略)	85	X v s (概略)	78
W m (概略)	85	X v S (概略)	78
W M c (概略)	94	X v u (概略)	77
W M v (概略)	94	X v U (概略)	77
W M w (概略)	94		
W o (概略)	86	Y	
W p (概略)	88	y (サーバー)	134
W q (概略)	86	Y c (概略)	107
W Q (概略)	86	Y d (概略)	108
W r (概略)	85	Y g (概略)	107
W s (概略)	88	Y t (概略)	108
W t (概略)	86		
W T e (概略)	92	Z	
W T i (概略)	91	z (サーバー)	134
W T l (概略)	92	z (グループ)	33
W T o (概略)	92	Z c (概略)	109
W T p (概略)	92	Z C (概略)	109
W T s (概略)	92	Z E (概略)	109
W T u (概略)	91	Z G (概略)	109
W u (概略)	87	Z n (概略)	109
W v (概略)	86	Z N (概略)	109
W w (概略)	86	Z p (概略)	109
W W a (概略)	83	Z P (概略)	109
W W b (概略)	83	Z Q (概略)	108
W W B (概略)	84	Z R (概略)	109
W W c (概略)	84	Z s (概略)	109
W W C (概略)	85	Z Z (概略)	108
W W C-c (概略)	84		
W W h (概略)	83		
W W l (概略)	83		
W W P (概略)	84		

Short Contents

The Gnus Newsreader	1
1 Gnus の起動	3
2 グループバッファー	13
3 概略バッファー	43
4 記事バッファー	115
5 メッセージの作成	123
6 選択方法	133
7 スコア	227
8 いろいろ	249
9 終わり	299
10 付録	301
11 GNU フリー文書利用許諾契約書	387
A GNU Free Documentation License	389
12 Index	397
13 Key Index	419

Table of Contents

The Gnus Newsreader	1
---------------------------	---

1 Gnus の起動 3

1.1 ニュースを見つける	3
1.2 一番初め	4
1.3 サーバーが落ちている	4
1.4 Gnus をスレーブにする	5
1.5 新しいグループ	5
1.5.1 新しいグループを調べる	5
1.5.2 購読方法	6
1.5.3 新しいグループを選別する	7
1.6 サーバーを換える	8
1.7 起動ファイル	8
1.8 自動保存	9
1.9 アクティブファイル	10
1.10 起動変数	11

2 グループバッファ 13

2.1 グループバッファの形式	13
2.1.1 グループ行の仕様	13
2.1.2 グループモード行の仕様	15
2.1.3 グループのハイライト	15
2.2 グループ操作	16
2.3 グループの選択	17
2.4 購読制御コマンド	18
2.5 グループデータ	19
2.6 グループレベル	19
2.7 グループのスコア	20
2.8 グループへの印	21
2.9 外部グループ	21
2.10 グループパラメーター	23
2.11 グループの一覧表示	29
2.12 グループの並べ替え	30
2.13 グループの管理	32
2.14 外部サーバーの閲覧	32
2.15 Gnus の終了	33
2.16 Group Topics	33
2.16.1 トピック命令	34
2.16.2 トピック変数	36
2.16.3 トピックの並べ替え	36
2.16.4 トピックの位相構造	37
2.16.5 トピックパラメーター	37

2.17	その他のグループ関連	38
2.17.1	新着メッセージを探す	40
2.17.2	グループ情報	40
2.17.3	グループの日付	41
2.17.4	ファイル命令	42
2.17.5	Sieve コマンド	42

3 概略バッファ

43

3.1	概略バッファの様式	43
3.1.1	概略バッファの行	43
3.1.2	To From Newsgroups	46
3.1.3	概略バッファのモード行	47
3.1.4	概略のハイライト	48
3.2	概略間の移動	48
3.3	記事の選択	49
3.3.1	選択命令	49
3.3.2	選ぶための変数	50
3.4	記事のスクロール	50
3.5	返答、フォローアップ、投稿	51
3.5.1	概略でのメールの命令	51
3.5.2	概略の投稿命令	53
3.5.3	概略メッセージ命令	54
3.5.4	記事を取り消す	54
3.6	遅延記事	55
3.7	記事に印を付ける	56
3.7.1	未読記事	56
3.7.2	既読記事	57
3.7.3	他の印	57
3.7.4	印を付ける	58
3.7.5	Generic Marking Commands	60
3.7.6	プロセス印を付ける	60
3.8	制限をする	61
3.9	スレッド	63
3.9.1	スレッドをカスタマイズする	63
3.9.1.1	無束縛スレッド	63
3.9.1.2	スレッドを埋める	66
3.9.1.3	もっとスレッドを	67
3.9.1.4	低レベルにおけるスレッド作成	68
3.9.2	スレッドの命令	68
3.10	並べ替え	69
3.11	非同期記事取得	70
3.12	記事のキャッシュ	71
3.13	永続記事	72
3.14	記事のバックログ	73
3.15	記事の保存	73
3.16	記事のデコード	77
3.16.1	uuencode された記事	77
3.16.2	シェルアーカイブ	78

3.16.3	ポストスクリプトファイル	78
3.16.4	他のファイル	78
3.16.5	デコードのための変数	78
3.16.5.1	規則変数	78
3.16.5.2	他のデコードのための変数	79
3.16.5.3	uuencode と投稿	80
3.16.6	ファイルの表示	81
3.17	記事のトリートメント	81
3.17.1	記事のハイライト	81
3.17.2	記事中の文の強調表示	82
3.17.3	記事を隠す	83
3.17.4	記事の洗濯	85
3.17.5	記事ヘッダー	88
3.17.6	記事のボタン	89
3.17.6.1	関連する変数と関数	90
3.17.7	Article button levels	91
3.17.8	記事の日付	91
3.17.9	Article Display	92
3.17.10	記事の署名	93
3.17.11	記事いろいろ	93
3.18	MIME コマンド	94
3.19	文字セット	97
3.20	記事命令	98
3.21	概略の並べ替え	98
3.22	親記事を探す	99
3.23	代替手段	100
3.23.1	選んで読む	100
3.23.2	バイナリーグループ	101
3.24	木表示	101
3.25	メールグループ命令	103
3.26	概略のいろいろなもの	105
3.26.1	概略グループ情報	107
3.26.2	記事を探す	107
3.26.3	概略生成命令	107
3.26.4	本当にいろいろな概略命令	108
3.27	概略バッファを抜ける	108
3.28	クロスポストの扱い	110
3.29	重複の抑制	110
3.30	セキュリティ	111
3.31	メーリングリスト	112
4	記事バッファ	115
4.1	余分なヘッダーを隠す	115
4.2	MIME を使う	116
4.3	記事のカスタマイズ	118
4.4	記事のキーマップ	120
4.5	記事のその他	121

5	メッセージの作成	123
5.1	メール	123
5.2	投稿するサーバー	123
5.3	POP before SMTP	124
5.4	メールと投稿	125
5.5	メッセージの保管	126
5.6	投稿様式	128
5.7	下書き	130
5.8	拒否された記事	131
5.9	署名と暗号化	131
6	選択方法	133
6.1	サーバーバッファー	133
6.1.1	サーバーバッファーの表示様式	133
6.1.2	サーバー命令	134
6.1.3	方法の例	135
6.1.4	仮想サーバーを作成する	136
6.1.5	サーバー変数	136
6.1.6	サーバーと選択方法	137
6.1.7	使用不可能なサーバー	137
6.2	ニュースの取得	137
6.2.1	NNTP	138
6.2.1.1	直接接続するための関数	140
6.2.1.2	間接的に接続するための関数	141
6.2.1.3	共通の変数	143
6.2.1.4	NNTP marks	144
6.2.2	ニューススプール	144
6.3	メール取得	145
6.3.1	ニュースリーダーでメール	145
6.3.2	メールを読むことを始める	146
6.3.3	メールの分割	147
6.3.4	メールソース	148
6.3.4.1	メールソース指示子	148
6.3.4.2	関数インターフェース	154
6.3.4.3	メールソースのカスタマイズ	154
6.3.4.4	メールの取得	155
6.3.5	メールバックエンド変数	156
6.3.6	特級メール分割	157
6.3.7	グループメール分割	160
6.3.8	古いメールを取り込む	162
6.3.9	メールの期限切れ消去	163
6.3.10	メール洗濯	165
6.3.11	重複	167
6.3.12	メールを読むのではない	167
6.3.13	メールバックエンドを選ぶ	168
6.3.13.1	Unix メールボックス	168
6.3.13.2	Rmail Babyl	168
6.3.13.3	メールスプール	169

6.3.13.4	MH スプール	170
6.3.13.5	Maildir	171
6.3.13.6	グループパラメーター	172
6.3.13.7	記事の識別	174
6.3.13.8	NOV データ	174
6.3.13.9	記事の印	174
6.3.13.10	メールフォルダー	175
6.3.13.11	メールバックエンドの比較	176
6.4	Browsing the Web	179
6.4.1	メールの保存	179
6.4.2	ウェブ検索	180
6.4.3	Slashdot	181
6.4.4	Ultimate	182
6.4.5	Web Archive	182
6.4.6	RSS	183
6.4.7	W3 のカスタマイズ	185
6.5	IMAP	185
6.5.1	IMAP での分割	190
6.5.2	IMAP での期限切れ消去	192
6.5.3	IMAP の ACL を編集する	193
6.5.4	メールボックスの削除	193
6.5.5	名前空間に関する注意	193
6.5.6	IMAP のデバッグ	194
6.6	その他のグループ源	194
6.6.1	ディレクトリーグループ	194
6.6.2	なんでもグループ	195
6.6.3	文書グループ	196
6.6.3.1	文書サーバーの内部	197
6.6.4	SOUP	199
6.6.4.1	SOUP 命令	200
6.6.4.2	SOUP グループ	201
6.6.4.3	SOUP 返信	202
6.6.5	メールからニュースへのゲートウェイ	202
6.7	合併グループ	203
6.7.1	仮想グループ	203
6.7.2	Kiboze グループ	204
6.8	電子メールによる日程管理	205
6.8.1	NNDiary バックエンド	206
6.8.1.1	日程メッセージ	206
6.8.1.2	NNDiary を動かす	206
6.8.1.3	NNDiary のカスタマイズ	207
6.8.2	Gnus Diary ライブラリー	208
6.8.2.1	日程の概略行仕様	208
6.8.2.2	日程記事の並べ替え	209
6.8.2.3	日程ヘッダーの生成	209
6.8.2.4	日程グループのパラメーター	209
6.8.3	送信するべきか、しないべきか	209
6.9	Gnus の切り離し	210

6.9.1 エージェントの基礎	210
6.9.2 エージェント分類	211
6.9.2.1 分類の文法	212
6.9.2.2 分類バッファ	216
6.9.2.3 分類変数	216
6.9.3 エージェント命令	217
6.9.3.1 グループエージェント命令	217
6.9.3.2 概略エージェント命令	217
6.9.3.3 サーバーエージェント命令	218
6.9.4 エージェントの視覚効果	218
6.9.5 キャッシュとしてのエージェント	219
6.9.6 エージェント期限切れ消去	219
6.9.7 エージェントを作り直す	220
6.9.8 エージェントとフラグ	220
6.9.9 エージェントを IMAP で使う方法	221
6.9.10 差出用メッセージ	221
6.9.11 エージェント変数	221
6.9.12 設定例	223
6.9.13 一括エージェント処理	224
6.9.14 エージェントの問題点	224
7 スコア	227
7.1 概略スコア命令	227
7.2 グループスコア命令	229
7.3 スコア変数	230
7.4 スコアファイル様式	232
7.5 スコアファイルの編集	236
7.6 適応スコア付け	237
7.7 ホームスコアファイル	239
7.8 自分自身へのフォローアップ	240
7.9 他のヘッダーにスコアを付ける	241
7.10 スコア付けの奥義	241
7.11 逆スコア	242
7.12 グローバルスコアファイル	242
7.13 消去ファイル	243
7.14 消去ファイルの変換	244
7.15 上級スコア付け	244
7.15.1 上級スコア付け構文	245
7.15.2 上級スコア付けの例	245
7.15.3 上級スコアのちょっとした秘訣	246
7.16 スコアを減衰させる	247

8	いろいろ	249
8.1	プロセス/接頭引数	249
8.2	利用者との相互作用	249
8.3	シンボルの接頭引数	250
8.4	書法仕様変数	250
8.4.1	書法仕様の基本	251
8.4.2	モード行書法仕様	251
8.4.3	上級書法仕様	251
8.4.4	利用者定義の指定	252
8.4.5	書法仕様フォント	252
8.4.6	ポイントの移動	253
8.4.7	整列	253
8.4.8	Wide Characters	254
8.5	ウィンドウの配置	254
8.5.1	ウィンドウ配置の例	257
8.6	フェースとフォント	258
8.7	コンパイル	258
8.8	モード行	258
8.9	ハイライトとメニュー	259
8.10	ボタン	260
8.11	デーモン	261
8.12	NoCeM	262
8.13	やり直し	264
8.14	述語指示子	264
8.15	司会役	264
8.16	グループを取得する	265
8.17	画像の拡張	265
8.17.1	X-Face	265
8.17.2	Face	267
8.17.3	スマイリー	267
8.17.4	Picons	268
8.17.5	さまざまな XEmacs 変数	269
8.17.5.1	ツールバー	269
8.18	ファジーな一致	269
8.19	spam メールの裏をかく	269
8.19.1	Spam の問題	270
8.19.2	Spam 退治の基礎	271
8.19.3	SpamAssassin, Vipul's Razor, DCC, etc	272
8.19.4	Hashcash	273
8.20	Spam パッケージ	274
8.20.1	Spam パッケージ序説	274
8.20.2	やって来るメールの濾過	276
8.20.3	グループにおける spam の検出	277
8.20.4	Spam と Ham プロセッサ	278
8.20.5	Spam パッケージの設定例	280
8.20.6	Spam バックエンド	283
8.20.6.1	ブラックリストとホワイトリスト	283
8.20.6.2	BBDB ホワイトリスト	284

8.20.6.3	Gmane Spam 報告	285
8.20.6.4	非-spam Hashcash 印	285
8.20.6.5	ブラックホール	285
8.20.6.6	正規表現によるヘッダーの合致検査	286
8.20.6.7	Bogofilter	286
8.20.6.8	SpamAssassin back end	288
8.20.6.9	ifile による spam の濾過	288
8.20.6.10	Spam 統計濾過	289
8.20.6.11	Gnus で SpamOracle を使うには	289
8.20.7	Spam パッケージの拡張	291
8.20.8	Spam 統計パッケージ	292
8.20.8.1	spam-統計 (spam-stat) 辞書を作る	293
8.20.8.2	spam-統計 (spam-stat) を使ってメールを分割する	294
8.20.8.3	spam-統計 (spam-stat) 辞書への低階層インターフェース	295
8.21	他のモードとの相互作用	296
8.21.1	Dired	296
8.22	いろいろのいろいろ	297
9	終わり	299
10	付録	301
10.1	XEmacs	301
10.2	歴史	301
10.2.1	Gnus Versions	301
10.2.2	他の Gnus のバージョン	302
10.2.3	なぜ?	302
10.2.4	互換性	302
10.2.5	標準への準拠	303
10.2.6	Emacsen	304
10.2.7	Gnus の開発	304
10.2.8	貢献者	305
10.2.9	新しい機能	307
10.2.9.1	(ding) Gnus	307
10.2.9.2	September Gnus	309
10.2.9.3	Red Gnus	311
10.2.9.4	Quassia Gnus	313
10.2.9.5	Pterodactyl Gnus	314
10.2.9.6	Oort Gnus	315
10.2.9.7	No Gnus	323
10.3	マニュアル	327
10.4	マニュアルを書く	327
10.5	用語	328
10.6	カスタマイズ	331
10.6.1	遅くて高価な NNTP 接続	331
10.6.2	遅いターミナル接続	331
10.6.3	少ないディスク容量	332
10.6.4	遅いマシン	332

10.7	問題解決	333
10.8	Gnus リファレンスガイド	335
10.8.1	Gnus の有用な関数	335
10.8.2	バックエンド インターフェース	336
10.8.2.1	必須バックエンド 関数	337
10.8.2.2	任意バックエンド 関数	340
10.8.2.3	エラーメッセージの発行	344
10.8.2.4	新しいバックエンドを書く	344
10.8.2.5	新しいバックエンドを Gnus に繋げる	346
10.8.2.6	メール風バックエンド	347
10.8.3	スコアファイルの構文	348
10.8.4	ヘッダー	349
10.8.5	範囲	350
10.8.6	グループ情報	350
10.8.7	対話形式の拡張	352
10.8.8	Emacs/XEmacs コード	353
10.8.9	いろいろなファイル様式	353
10.8.9.1	アクティブファイルの様式	353
10.8.9.2	ニュースグループファイルの様式	354
10.9	異教徒への Emacs	355
10.9.1	打鍵	355
10.9.2	Emacs Lisp	355
10.10	よく尋ねられる質問	357
10.10.1	インストールに関する FAQ	357
10.10.2	起動 / グループバッファ	359
10.10.3	メッセージの取得	360
10.10.4	メッセージを読む	365
10.10.5	メッセージの作成	371
10.10.6	古いメッセージ	378
10.10.7	ダイアルアップ環境で Gnus を使う	380
10.10.8	助けを得る	382
10.10.9	Gnus をチューンする	383
10.10.10	用語集	385
11	GNU フリー文書利用許諾契約書	387
 Appendix A GNU Free Documentation License		
	389
	ADDENDUM: How to use this License for your documents.....	395
12	Index	397
13	Key Index	419

