Wanderlust ユーザマニュアル

Yet another message interface on Emacsen
for Wanderlust version 2.15.9

寺西裕一
奥西藤和
村田全寛
岡田健一
高橋 郁
山岡克美
村田浩也
中山洋一

1 はじめに

Wanderlust は Emacs で動くメール/ニュース管理システムです。IMAP4rev1(RFC2060), NNTP, POP およびローカルのメッセージに対応しています。

Wanderlust の主な特徴/特長は以下の通りです。

- elisp のみによる実装。
- IMAP4rev1, NNTP, POP(POP3/APOP), MH, Maildir 形式のサポート。
- Mew っぽい Folder Specification に基づくメッセージへの統一的アクセス。
- Mew っぽいキーバインドとマーク処理。
- 未読管理。
- インタラクティブなスレッド表示。
- 購読フォルダの一覧を表示するフォルダモード。
- メッセージキャッシュ、Disconnected Operation。
- MH 的 FCC。('`Fcc: %Backup`' や '`Fcc: $Backup`' も可)。
- MIME 対応 (by SEMI)。
- ニュース/メールの送信を統合したメッセージ送信ドラフト。
- フォルダ一覧のアイコン表示 (XEmacs と Emacs 21)。
- 大きなパートを取り寄せずに表示 (IMAP4)。
- メッセージの検索をサーバ側で実行 (IMAP4)。日本語検索も可。
- 仮想フォルダ。
- 多い日も安心の、マルチアーカイバ対応圧縮フォルダ。
- フォルダ中の古い記事を自動的にアーカイブ/削除して整理する expire 機能。
- 自動リファイル。
- 定型メッセージの送信に便利なテンプレート機能。

1.1 動作環境

Wanderlust は以下の Emacsen で動作することが確認されています。

- Mule 2.3 based on Emacs 19.34
- Emacs 20.2 以降
- XEmacs 20.4 以降
- Meadow 1.00 以降
- NTEmacs 20.4 以降
- PMMule

Wanderlust から利用できると確認されている IMAP サーバは以下の通りです。

- UW imapd 4.1〜4.7, 4.7a, 4.7b, 4.7c, 2000 以降
- Cyrus imapd 1.4, 1.5.19, 1.6.22〜1.6.24, 2.0.5 以降
- Courier-IMAP 1.3.2 以降
- AIR MAIL (AIRC imapd release 2.00)
- Express Mail

- Microsoft Exchange Server 5.5
- Sun Internet Mail Server 3.5, 3.5.alpha, 4.0

Wanderlust から利用できると確認されている LDAP サーバは以下の通りです。

- OpenLDAP 2.0.6 以降

2 Wanderlust を起動する

Wanderlust を起動するまでに必要な手順を順番に説明します。

(当然のことながら、これらよりも先に、メール/ニュースを読むことのできる環境があらかじめ必要です。)

2.1 MIME 用モジュールのインストール

Wanderlust を使うためには、APEL, FLIM, SEMI と呼ばれるパッケージが必要です。

Emacs 23 以降については、以下の場所にあるものを使用してください。SEMI は SEMI-EPG というパッケージ名になります。

```
APEL:     https://github.com/wanderlust/apel
FLIM:     https://github.com/wanderlust/flim
SEMI-EPG: https://github.com/wanderlust/semi
```

オリジナルの APEL, FLIM, SEMI は以下の場所からダウンロードできます。

```
APEL:   http://git.chise.org/elisp/dist/semi/
FLIM:   http://git.chise.org/elisp/dist/apel/
SEMI:   http://git.chise.org/elisp/dist/semi/
```

APEL, FLIM, SEMI[-EPG] の順にインストールしてください。基本的にすべて '`make install`' の実行で済むはずです (XEmacs 21 では '`make install-package`')。

インストールの方法の詳細については、各パッケージに添付されているドキュメントを参照してください [1]。

推奨される APEL, FLIM, SEMI のバージョンの組合せは、以下の通りです。

- APEL 10.8, FLIM 1.14.9, SEMI-EPG 1.14.7

その他、FLIM, SEMI にはいろいろな変形バージョンが存在しますが、それらのいずれも利用可能です。基本的に最新版の組合せなら動作するはずです。例えば、以下の組合せで動作することが確認されています。

- APEL 10.6, SLIM 1.14.9, SEMI 1.14.5
- APEL 10.6, CLIME 1.14.5, EMIKO 1.14.1

APEL, FLIM もしくは SEMI のバージョンアップを行った場合は、Wanderlust をインストールし直してください。

2.2 パッケージの入手と展開

Wanderlust 本体は以下の場所からダウンロードできます。

一次配布元:

```
https://github.com/wanderlust/wanderlust
```

ミラーしていただいている ftp, http サイト:

```
http://www.jpl.org/ftp/pub/github-snapshots/
```

[1] Emacs 19.34 ベースの Mule を使っている場合、
`http://www.jpl.org/ftp/attic/INSTALL-SEMI-ja.html` が参考になるかもしれません。

入手したパッケージは適当な作業ディレクトリに展開しましょう。

```
% cd ~/work
% tar zxvf wl-version.tar.gz
% cd wl-version
```

2.2.1 SSL (Secure Socket Layer) の利用

Wanderlust では、SMTP, IMAP, NNTP, POP のコネクションにおいて、SSL (Secure Socket Layer) を使用できます。

SSL の利用形態には、コネクションと同時に SSL negotiation を始めるものと、STARTTLS command の後に SSL negotiation を始めるものの二種類があります。

前者については、Emacs 24 以降でサポートされた、組み込み GnuTLS が利用可能な場合はそれを使います。次に、 tls.el が存在すればそれを使います。この時に GnuTLS に含まれる gnutls-cli にパスが通っている必要があります。どちらもない場合は、本パッケージの utils ディレクトリにある ssl.el をインストールする必要があります。なおかつ、OpenSSL に含まれる openssl にパスが通っている必要があります。

後者については starttls.el が必要になります。変数 starttls-use-gnutls の値によって、GnuTLS もしくは starttls パッケージが必要です。starttls.el が無い、もしくは starttls.el 内に starttls-use-gnutls の定義が無い場合は starttls パッケージをインストールする必要があります。

starttls パッケージは以下の場所から入手できます。

```
ftp://opaopa.org/pub/elisp/
```

2.3 バイトコンパイルとインストール

2.3.1 通常のインストール

Makefile の LISPDIR, EMACS のあたりを編集します。LISPDIR にはパッケージのインストール先、EMACS には利用する Emacs のコマンド名を指定します。

```
% make
% make install
```

Makefile 中の LISPDIR を変更せず、そのまま ('NONE' のままでも) インストールした場合、適当なインストール先を自動的に検出します。実際のインストール先については後述します。

Mule 2.3 など、subdirs.el が無く、load-path にサブディレクトリが自動的に加わらない Emacs では、

```
Cannot open load file: mime-setup
```

というエラーが出ることがあります。この場合は、custom, APEL, FLIM, SEMI のインストール先を環境変数 EMACSLOADPATH に加えるか、展開ディレクトリの WL-CFG というファイル中で load-path を通しておくと良いでしょう。

また、新聞フォルダを利用する場合や、BBDB を利用する場合には、それぞれ emacs-w3m、BBDB がインストールされているディレクトリに load-path を通しておくと必要なモジュールがバイトコンパイル/インストールされます。3.8 節「Shimbun Folder」(15 ページ) を参照してください, 14.1.2 節 「BBDB」(96 ページ) を参照してください.

2.3.2 WL-CFG

WL-CFG というファイルが展開ディレクトリに存在すると、インストール時に読み込まれるようになっています。インストールで SEMI 等の load-path の設定が必要であれば、WL-CFG に設定してください。

インストール先は Makefile 中の LISPDIR で指定しますが、実際に *.el, *.elc が入るディレクトリ (LISPDIR からの相対パス) は WL-CFG 中の変数 WL_PREFIX, ELMO_PREFIX で指定します。

WL_PREFIX
: WL モジュールをインストールするディレクトリを指定します。これは LISPDIR からの相対パスです。WL モジュールはファイル wl*.el wl*.elc を含んでいます。

ELMO_PREFIX
: ELMO モジュールをインストールするディレクトリを指定します。これは LISPDIR からの相対パスです。ELMO モジュールはファイル elmo*.el elmo*.elc を含んでいます。

WL_PREFIX, ELMO_PREFIX のデフォルトはいずれも wl です。

elmo* (ELMO モジュール) を elmo 配下にインストールしたければ、

```
(setq ELMO_PREFIX "elmo")
```

とします。

2.3.3 XEmacs package としてインストール

Wanderlust は XEmacs (21.0 〜) の package のひとつとしてインストールすることも可能です。package としてインストールすると、autoload の設定、アイコンのパス設定を個人の ~/.emacs に記述しなくても Wanderlust を正常に起動できるようになります。

XEmacs の package としてインストールするには以下のようにします。

```
% vi Makefile
% make package
% make install-package
```

package のディレクトリは、SEMI をインストールしてあれば自動検出されます。(Makefile 中の PACKAGEDIR でも設定可)

2.3.4 インストールしないで利用

Wanderlust はバイトコンパイル、インストールをしなくても、wl, elmo のディレクトリに load-path を設定すれば起動することができます。例えば ~/work にパッケージを展開した場合、~/.emacs に以下の設定をすれば起動できます。

```
(add-to-list 'load-path "~/work/wl-version/wl")
(add-to-list 'load-path "~/work/wl-version/elmo")
```

2.3.5 マニュアルについて

マニュアルは Info 形式です。インストールするには下記を実行してください。

```
% make info
% make install-info
```

XEmacs の package としてインストールした場合は自動的に Info ファイルもインストールされているので、これらの操作は必要ありません。

また、下記にもマニュアルがあります。

```
http://wanderlust.github.io/wl-docs/wl-ja.html
```

2.4 .emacs, .wl の設定

Wanderlust のパッケージは、大きく分けて二つのモジュール群を含んでいます。

'ELMO (elmo-*.el)'
: すべてをフォルダに見せるモジュール群です。WL のバックエンドです。

'WL (wl-*.el)'
: Wanderlust 本体の動作を決めるモジュール群です。ELMO のフロントエンドです。

ユーザは elmo-, wl- で始まる変数の設定を変えることによって Wanderlust の動作をカスタマイズできます。

最低限必要な設定は以下の通りです。

```
;; autoload の設定
;; (XEmacs の package としてインストールした場合は必要ありません)
(autoload 'wl "wl" "Wanderlust" t)
(autoload 'wl-other-frame "wl" "Wanderlust on new frame." t)
(autoload 'wl-draft "wl-draft" "Write draft with Wanderlust." t)

;; アイコンを置くディレクトリ。初期設定は Emacs 固有のデフォルト値。
;; (デフォルトの値が正しければ必要ありません)
(setq wl-icon-directory "~/work/wl/etc")

;; メールを送信する SMTP サーバ。 初期設定は nil。
(setq wl-smtp-posting-server "your.smtp.example.com")
;; ニュース投稿用の NNTP サーバ。 初期設定は nil。
(setq wl-nntp-posting-server "your.nntp.example.com")
```

Wanderlust 起動後、~/.wl が存在すればそれをロードします。したがって Wanderlust に固有の設定は ~/.wl に記述しておくと整理しやすいでしょう。face の設定は ~/.emacs に書くことはできないので~/.wl に書いてください。14.2 節 「Highlights」(98 ページ) を参照してください.

上記のうち、autoload の設定は ~/.emacs に書く必要があります。それ以外の設定は ~/.wl に記述できます。

2.4.1 mail-user-agent

以下のような設定を ~/.emacs にしておくと、*C-x m* (compose-mail) によって Wanderlust のドラフトモードを起動するようになります。Wanderlust を Emacs 上の標準メーラとして使いたい場合は設定しておくと良いでしょう。ただし、これは mail-user-agent の定義が可能な Emacsen の場合のみ有効です。*GNU Emacs Manual* の “Mail-Composition Methods” 節 を参照してください.

```
(autoload 'wl-user-agent-compose "wl-draft" nil t)
(if (boundp 'mail-user-agent)
    (setq mail-user-agent 'wl-user-agent))
(if (fboundp 'define-mail-user-agent)
    (define-mail-user-agent
      'wl-user-agent
      'wl-user-agent-compose
      'wl-draft-send
      'wl-draft-kill
      'mail-send-hook))
```

2.5 購読するフォルダの定義

購読するフォルダをファイル ~/.folders に定義します。~/.folders に書かれた内容がそのままあなたの購読するフォルダとなります。

起動した状態でフォルダ一覧のバッファから購読フォルダを追加/編集することも可能ですので、この項は読み飛ばしても構いません。4.2 節 「Folder Manager」(28 ページ) を参照してください.

~/.folders の書き方はとても単純です。こんな感じです。

```
#
# '#' で始まる行はコメント。
# 空行は無視。
#
# フォルダ   "あだ名"
# (あだ名は無くてもよい)
#
%inbox  "受信箱"
+trash  "ゴミ箱"
+draft  "草稿"
%#mh/Backup@my.imap.example.com "送信済み"
# グループの定義
Emacsen{
    %#mh/spool/wl            "Wanderlust ML"
    %#mh/spool/elips         "ELIPS ML"
    %#mh/spool/apel-ja       "APEL (日本語) ML"
    %#mh/spool/xemacs-beta   "XEmacs ベータ"
    -fj.news.reader.gnus@other.nntp.example.com
    *-fj.editor.xemacs,-fj.editor.mule,-fj.editor.emacs "fj の Emacsen"
}
#
# 行末に '/' がつくと、そのフォルダに含まれるサブフォルダ全てが
# ひとつのグループとなる (アクセスグループ)。
#
%#mh/expire@localhost /
# MH のフォルダ全てをひとつのグループにする例。
+ /
```

一行にひとつ、読みたいフォルダを書きます。各フォルダの定義については次の章で詳しく説明します。

‘グループ名{’ と ‘}’ で囲まれた部分は一つのグループとなります。ひとつのグループはフォルダモードでは開閉できるディレクトリのように見えます。いくつかのフォルダをまとめて整理するのに便利です。

注意すべきなのは、‘グループ名{’ と ‘}’ は 1 行を占領して書く必要があることです (これはパーサがダサいからです)。

グループには、2 つの種類があります。一つは、上の例の ‘Emacsen’ のように直接自分で好きなフォルダをグループとして定義するタイプです。

もう一つは、上の例の ‘+ /’ のような 『アクセスグループ』 です。これは、あるフォルダに含まれるサブフォルダ全てをまとめて一つのグループとするものです。(その動作はフォルダのタイプによって異なります。例えば ‘+ /’ なら MH のサブディレクトリすべてがひとつのグループとなります。)

実際に試してみて、確認してからの方がわかりやすいかもしれません。ちょっと書いて試してみてから、またフォルダの定義をやり直すと良いでしょう。

2.6 Wanderlust の起動

インストール、および設定がうまくいっていれば、

```
M-x wl
```

で起動できます。初期化の後、フォルダ一覧を表示するフォルダモードが現れます。

C-u M-x wl のように prefix argument つきで実行すると、フォルダのチェックを省略して起動します。

2.7 概観

Wanderlust では基本的に、次に挙げるバッファを互いに行き来しながらメッセージを取り扱います。それぞれの詳細については以下の各章で説明します。

‘フォルダバッファ’
: フォルダの一覧を表示します。フォルダを選択して、そのフォルダのサマリに入ることができます。また、購読するフォルダを追加したり、編集することもできます。

‘サマリバッファ’
: フォルダに含まれるメッセージの一覧を表示します。ここではメッセージを選択してその内容を表示したり、メッセージに対して返信をすることができます。また、メッセージを削除したり、別のフォルダに移動させたりすることができます。

‘メッセージバッファ’
: メッセージの内容を表示します。パートを保存したり、外部プログラムで開くことができます。

‘ドラフトバッファ’
: メッセージの編集を行います。

3 Wanderlust で扱えるフォルダたち

以下では Wanderlust で扱えるフォルダについて説明します。

Wanderlust は ELMO のインタフェースを利用しているため、ELMO モジュールがサポートしていれば、どんなフォルダでも利用できます。

バージョン 2.15.9 現在、用意されているフォルダは、IMAP, NNTP, LocalDir(MH), Maildir, News Spool, Archive, POP, Shimbun, Search, Multi, Filter, Pipe, File, Access, Internal の 15 種類です。

3.1 IMAP フォルダ

RFC 2060 で規定された IMAP4rev1 を利用してメールを読むためのフォルダです。

書式:

‘%’ メールボックス名 [‘:’ ユーザ名 [‘/’ 認証法]][‘@’ ホスト名][‘:’ ポート番号][‘!’]

認証法には login（エンコードしてパスワードを送信）か
cram-md5（CRAM-MD5 による認証）か
digest-md5（DIGEST-MD5 による認証）か
clear（または nil。生パスワードを送信）のいずれかを指定。

default 値:

ユーザ名 -> 変数 elmo-imap4-default-user の値。
初期設定は 環境変数 USER か、LOGNAME か、
(user-login-name) の返り値。
認証法 -> 変数 elmo-imap4-default-authenticate-type の値。
初期設定は login。
ホスト名 -> 変数 elmo-imap4-default-server の値。
初期設定は ‘localhost’。
ポート番号-> 変数 elmo-imap4-default-port の値。
初期設定は 143。

メインで使用する IMAP サーバを変数 elmo-imap4-default-server に指定すると、いちいちフォルダ名にホスト名を書かずに済みます。例えばファイアウォールを越えなければならない場合でも ‘foo%imap@gateway’ のように指定できます。

```
;; 例: imap4.exaple.org をメインで使用する IMAP サーバとして設定
(setq elmo-imap4-default-server "imap4.example.org")
```

フォルダ名の最後に ‘!’ が付いていると、SSL (Secure Socket Layer) を利用してコネクションを張ります。‘!!’ だと、STARTTLS により SSL コネクションを張ります。

変数 elmo-imap4-default-stream-type の値が ssl なら、‘!’ を付けなくても SSL を使います。starttls なら ‘!!’ を意味します。これらの場合、通常の接続をするにはフォルダ名の最後に ‘!direct’ を付けます。

```
;; 例: SSL を利用してコネクションを張る
(setq elmo-imap4-default-stream-type 'ssl)
```

認証法として、login、cram-md5 もしくは digest-md5 を指定した場合、パスワードをエンコードして送信します。ただし、サーバ側がパスワードをエンコードして受け取る能力

が無い場合は、確認の後、clear (生パスワードを送る) に切替えます。変数 elmo-imap4-force-login が non-nil ならば、確認無しに clear に切替えます (初期設定は nil)。

```
;; 例: 生パスワードで認証
(setq elmo-imap4-default-authenticate-type 'clear)
```

例:

```
%inbox     -> IMAP のメールボックス、"inbox"
%#mh/inbox -> IMAP のメールボックス、"#mh/inbox"

%inbox:hoge -> IMAP のメールボックス、"inbox" へユーザ "hoge" でアクセス。

%inbox:hoge/clear@server1
            -> server1 上の IMAP のメールボックス "inbox" へ
               ユーザ "hoge" で、生パスワードを送って ('clear で)
               アクセス。
```

3.1.1 日本語メールボックス名の扱い (Modified UTF7)

Emacs が UTF-7 に対応していれば，変数 elmo-imap4-use-modified-utf7 に non-nil の値を設定すれば (デフォルトは Emacs 23 以降では t、それ以外では nil)、日本語 (や、その他の英語以外の言語) でメールボックス名を指定することができます。

Emacs 23 以降では追加のパッケージは不要です。それ以外の Emacsen では、Mule-UCS を別にインストールする必要があります。Mule-UCS は以下の Emacsen で動作します。

- Emacs 20.3 以降
- XEmacs 21.2.37 以降

Mule-UCS は以下から 2004 年のスナップショットが入手可能です。

```
http://www.jpl.org/ftp/pub/tmp/
```

3.2 NNTP フォルダ

ネットニュースを読むためのフォルダです。一つのニュースグループを一つのフォルダとして扱います。

書式:

```
'-' ニュースグループ名 [':' ユーザ名]['@' ホスト名][':' ポート番号]['!']
```

default 値:

```
ホスト名   -> 変数 elmo-nntp-default-server の値。初期設定は "localhost"。
ユーザ名   -> 変数 elmo-nntp-default-user の値。初期設定は nil。
ポート番号-> 変数 elmo-nntp-default-port の値。
              初期設定は 119。
```

ユーザ名が non-nil の場合は AUTHINFO による認証を行ないます。フォルダ名の最後に '!' が付いていると、SSL を利用してコネクションを張ります。'!!' だと、STARTTLS により SSL コネクションを張ります。

変数 elmo-nntp-default-stream-type の値が ssl なら、'!' を付けなくても SSL を使います。starttls なら '!!' を意味します。これらの場合、通常の接続をするにはフォルダ名の最後に '!direct' を付けます。

例:

```
-fj.rec.tv            -> ニュースグループ、‘fj.rec.tv’。
-fj.rec.tv@newsserver -> ‘newsserver’ 上のニュースグループ、‘fj.rec.tv’。
```

3.3 MH フォルダ

MH 形式 (1 ファイル 1 メール) で保存されたメールを読むためのフォルダです。

書式:

‘+’ ディレクトリ名

ディレクトリ名は、通常、変数 elmo-localdir-folder-path (初期設定は ~/Mail) からの相対パスですが、‘/’ や ‘~’ で始まっていれば絶対パスと見做します (ドライブレターも同様です)。

メッセージが保存されるファイルのファイル名には、メッセージ番号を使用します。

例:

```
+inbox         -> ~/Mail/inbox
+from/teranisi -> ~/Mail/from/teranisi
+~/test        -> ~/test
```

3.4 Maildir フォルダ

Maildir 形式 (1 ファイル 1 メール) で保存されたメールを読むためのフォルダです。

書式:

‘.’ ディレクトリ名

ディレクトリ名は、通常、変数 elmo-maildir-folder-path (初期設定は ~/Maildir) からの相対パスですが、‘/’ や ‘~’ で始まっていれば絶対パスと見做します (ドライブレターも同様です)。

Maildir は、cur, new, tmp のディレクトリを含んでいます。実際にメッセージが存在するのは、指定ディレクトリ直下の cur ディレクトリです。指定ディレクトリ直下の new ディレクトリに存在するメッセージファイルは、アクセス時に cur ディレクトリへ移動されます。また、tmp ディレクトリにあり、かつ 36 時間以上アクセスが無いメッセージファイルは削除されます。

この動作は http://cr.yp.to/proto/maildir.html に従っています。
(日本語訳は http://man.qmail.jp/jman5/maildir.html)

例:

```
.              -> ~/Maildir
.inbox         -> ~/Maildir/inbox
.from/teranisi -> ~/Maildir/from/teranisi
.~/test        -> ~/test
```

3.5 News Spool フォルダ

Mew/IM が提唱する、ローカルに保存されたニュース記事を読み書きするためのフォルダです。NNTP 経由ではなく、gnspool などを使って取り寄せているような場合にそのスプールを直接読む、という使い方も想定しています。

書式:

‘=’ ディレクトリ名

ディレクトリ名は、変数 elmo-localnews-folder-path (初期設定は ~/News) で指定したディレクトリのサブディレクトリを指します。ディレクトリの区切りは ‘.’ でも可です。

例:

```
=fj/os/os2          -> ~/News/fj/os/os2
=fj.os.bsd.freebsd -> ~/News/fj/os/bsd/freebsd
```

3.6 アーカイブフォルダ

Info-ZIP や LHA などで圧縮されたアーカイブファイルを一つのフォルダとして扱います。

書式:

‘$’ ディレクトリ名 [‘;’ アーカイバタイプ ‘;’ プレフィクス]

ディレクトリ名は、通常、変数 elmo-archive-folder-path (初期設定は ~/Mail) からの相対パスですが、‘/’ や ‘~’ で始まっていれば絶対パスと見做します (ドライブレターも OK)。ange-ftp 表記も ange-ftp, efs が使える環境では OK です。

フォルダの実体としての書庫ファイルは、上述のディレクトリにある elmo-archive-basename(初期値は elmo-archive) になります。ただし、ディレクトリでなくファイルであった場合、そのファイルをフォルダと見做します。拡張子はアーカイバ毎に自動的に (動的に) 選択されます。

アーカイバタイプを省略した場合、変数 elmo-archive-default-type (初期設定は zip) が参照されます。

プレフィクスは、書庫がディレクトリ構造を持っている場合に、そのディレクトリ部分を指定します。これは主にアーカイブサービスや tar + gzip + uuencode によるダイジェスト配送を提供している ML の書庫ファイルを展開することなくマウントするためのもの、つまり閲覧時の便宜のためのものです。

例えば ML サーバが fml の場合、msend.tar.gz は spool/1 のような構造なので、‘spool’ を指定します。

例:

```
$teranisi          -> ~/Mail/teranisi/elmo-archive.zip
$bsd/freebsd;lha  -> ~/Mail/bsd/freebsd/elmo-archive.lzh
$/foo@server:~/bar;zoo     -> ‘server’ 上の ~/bar/elmo-archive.zoo
$d:/msend.tar.gz;tgz;spool -> d:/msend.tar.gz
$ml;zip/         -> ~/Mail/ml 以下のアーカイブフォルダからなる
                    アクセスグループ
```

3.6.1 アーカイブフォルダが対応している (対応可能な) アーカイバ

デフォルトで以下のアーカイバに対応します。

```
LHA, Info-ZIP/UNZIP, ZOO, RAR  ;; フルスペック
GNU TAR(tgz, tar) ;; デフォルトでは閲覧専用
```

複数ファイルを 1 プロセスで一つの書庫へまとめることができるアーカイバであれば、必要な変数を追加定義するだけで使える可能性があります (ARJ/UNARJ, ARC は、手許で使ってないので定義していません。TAR は元ファイルを消す (mv) ことができない点で真面目にサポートしていません)。複数ファイルを一つにまとめられない点で gzip, bzip, bzip2 は使えません。標準出力へ解凍できないアーカイバにも標準では対応しません。

3.6.2 各 OS でのアーカイバに関する特記事項

フルスペックで読み書き可能なことを確認しているアーカイバは、以下のとおりです ('*' 印のものは処理速度などの点で最も適しているもの)。

```
[OS/2]  Warp4.0J(w/o VoiceType)+Fx00505/emx0.9c(fix04)/PMMule,EmacsPM
         LHA  OS/2 version Rel.2.06b    Feb 18, 1998
        *UnZip 5.32 of 3 November 1997, by Info-ZIP.
        *Zip 2.2 (November 3rd 1997).
         Zoo archiver, zoo 2.1 $Date: 91/07/09 02:10:34 $
         GNU tar version 1.10 - AK 2.58 (DBCS/SJIS) 981216(homy) 版
         gzip 1.2.4 (18 Aug 93) + bzip2 パッチ (by 飯田さん)

[UN|X]  FreeBSD 2.2.7-RELEASE, Linux 2.0.30, Solaris2.6, HP-UX 9.07
         LHa for UNIX  V 1.14c
         UnZip 5.32 of 3 November 1997
         Zip 2.2 (November 3rd 1997)
         GNU tar 1.12 (1.11.x は不可)
         gzip 1.2.4 (18 Aug 93)

[Win32] Win.98/Meadow
         Lha32 version 1.28
         Zip 2.2
         UnZip 5.40
         GNU tar 1.11.8 + 1.5(WIN32)
         GZIP 1.2.4
         RAR 2.06
```

※ LHA に関する注意

OS/2 の場合、Peter Fitzsimmons 氏作の LH/2 には対応しません。平松版をお使いください。Win32 では DOS 版でなく、LHa32 でないと動かないとのことです。

※ GNU tar に関する注意

GNU tar は書庫からの削除に問題があるものが多いので、特に注意してください。書庫が破壊される危険性が高いので、フルスペックで読み書きする前に`--delete -f` を充分テストしておいてください。なお、上記のものでは今のところ問題は報告されていません。

3.6.3 TIPS

快適に移行するには、`wl-summary-archive` を実行する (9.5 節「Archive」(77 ページ) を参照) か、Expire 機能 (9.1 節「Expire」(71 ページ) を参照) と組み合わせると良いでしょう。ただし、Expire 機能で作成したアーカイブフォルダを扱う場合は、変数 `elmo-archive-treat-file` を non-nil に設定する必要があります。なお、OS/2 上でのテストでは、Mule2.3(19.28) と Emacs20.2 では処理速度に圧倒的な違いがあります。快適に使うには Emacs20 をお勧めします (re-search の速度の問題だとすると 19.3x 以上かどうかが境になるでしょう)。

また、一つの書庫ファイルが多くのファイルを含んでいるとアーカイバ起動時のオーバーヘッドが加速度的に増加する (特に LHA の場合) ため、150 通程度、最大でも 200 通までにしておくと、ストレスなく読み書きできるでしょう。

なお、当然のことながら

```
(setq wl-fcc "$backup")
(setq wl-trash-folder "$trash;lha")
```

も可能です:-)。

3.6.4 アーカイブフォルダに関する変数

`elmo-archive-default-type`
: デフォルトのアーカイバタイプをシンボルで指定します。初期値は `zip` です。

`elmo-archive-`*type*`-method-alist`
: アーカイバの *type* (実際には '`lha`', '`zip`', '`zoo`', '`tgz`' などの文字列が入る) 毎の、各種メソッドを記述します。連想リストの各要素は以下のようになります。

```
(action . (exec-name options))   ;; 外部プログラムとオプション
(action . function)              ;; 関数
```

現在のところ、有効な *action* は

```
'ls, 'cat ('cat-headers)         ;; 最低限必要 (閲覧のみ)
'mv ('mv-pipe), 'rm ('rm-pipe)   ;; 上とセットでフルスペック
'cp ('cp-pipe)                   ;;
```

です。括弧内のものは、無くても構いません (あれば優先的に使います)。

`elmo-archive-suffix-alist`
: アーカイバタイプ (シンボル) 毎に対応する書庫の拡張子を記述します。

`elmo-archive-file-regexp-alist`
: 書庫のリスト閲覧時の出力からファイル名を取得するための正規表現を、アーカイバタイプ (シンボル) 毎に記述します。

`elmo-archive-method-list`
: 有効にしたい *type* の elmo-archive-*type*-method-alist (*type* はアーカイバのシンボル) をリストで記述します。

`elmo-archive-lha-dos-compatible`
: この変数が non-nil であれば DOS 版 (吉崎氏オリジナル) の LHA とオプション互換と見做します。初期値では OS/2 と Win32 のみ `t` です。

`elmo-archive-cmdstr-max-length`
: elmo-archive からは (標準状態では) アーカイバをシェルを経由せずに起動します。elisp レベルでのコマンド文字列の総バイト数には制限はないとのことなので、多くのパラメータを一度に与えて動かせるかどうかは OS レベルの問題になります。これは、例えば数百通単位のメッセージを一度に消去できるかどうかの問題、と読み替えてください。

  OS/2 ではシェルを介さずに発行できるコマンド文字列は 8190 バイトまでなので、余裕を見てデフォルトを 8000 にしています。OS/2 REXX やシェルスクリプトなどを噛ます場合、シェルの実装に依存することに注意してください。

  なお、アーカイバが処理対象となるファイルのリストを標準入力から受け付ける (前述の `rm-pipe`, `mv-pipe`, `cat-headers` action を指定している) 場合、1 プロセスで処理することができます。

3.7 POP フォルダ

RFC 1939 で規定されている POP3 を利用してメールを読むためのフォルダです。

書式:

‘&’ [ユーザ名][‘/’ 認証法][‘:’ 番号の振り方][‘@’ ホスト名][‘:’ ポート番号][‘!’]

認証法には、‘user’ (生パスワードを送信して認証) と ‘apop’ (APOP で認証) の 2 種類があります。

番号の振り方には、‘uidl’ (UIDL コマンドによる番号付け) か ‘list’ (LIST コマンドによる番号付け) のいずれかを指定します。

default 値:

```
ユーザ名     -> 変数 elmo-pop3-default-user の値。
                初期設定は 環境変数 USER か、LOGNAME か、
                (user-login-name) の返り値。
認証法       -> 変数 elmo-pop3-default-authenticate-type の値。
                初期設定は "user"。
番号の振り方-> 変数 elmo-pop3-default-use-uidl の値による。
                t なら UIDL を用いる。初期設定は t。
ホスト名     -> 変数 elmo-pop3-default-server の値。
                初期設定は "localhost"。
ポート番号   -> 変数 elmo-pop3-default-port の値。
                初期設定は 110。
```

例:

```
&hoge@localhost -> localhost へユーザ ‘hoge’ でアクセス。
&hoge@popserver:109 -> ホスト ‘popserver’ のポート 109 番へ
                       ユーザ ‘hoge’ でアクセス。
```

APOP を利用するには、md5.el が必要です。(XEmacs では必要ありません。) md5.el は本パッケージの utils/sasl/lisp/ か Emacs/W3 パッケージ (http://www.cs.indiana.edu/elisp/w3/docs.html)、または LCD archive から入手可能です (GPL2)。

フォルダ名の最後に ‘!’ が付いていると、SSL を利用してコネクションを張ります。

変数 elmo-pop3-default-stream-type の値が ssl なら、‘!’ を付けなくても SSL を使います。starttls なら ‘!!’ を意味します。これらの場合、通常の接続をするにはフォルダ名の最後に ‘!direct’ を付けます。

3.8 新聞フォルダ

emacs-w3m (http://emacs-w3m.namazu.org/) を用いて WWW 上の新聞やニュースサイト、メーリングリストアーカイブなどを読むためのフォルダです。

w3m および emacs-w3m がインストールされている必要があります。

書式:

‘@’ モジュール名 ‘.’ フォルダ名

モジュール名、および **フォルダ名** に入る値については emacs-w3m に付属の README.shimbun.ja を参照してください。

例:

```
@airs.wl  -> wanderlust ML アーカイブ (モジュール sb-airs.el 使用)
@asahi/   -> sb-asahi.el モジュールに含まれるフォルダのアクセスグループ
```

3.8.1 新聞フォルダに関する変数

`elmo-shimbun-update-overview-folder-list`
: 初期設定は `all`。メッセージをフェッチしたときに overview を更新する新聞フォルダを指定します。全ての新聞フォルダで overview を更新したい場合には、`all` を設定します。フォルダ名の正規表現からなるリストを設定する事で、更新するフォルダを限定する事も出来ます。

  例:

```
(setq elmo-shimbun-update-overview-folder-list
      '("^@airs\\." "^@namazu\\."))
```

  サマリの表示は、自動的に更新されます。

3.9 RSS フォルダ

RSS や Atom フィードに含まれるメッセージを閲覧する為のフォルダです。

書式:

‘rss:’ *RSS* または *Atom* の *URL*

例:

```
rss:https://github.com/wanderlust/wanderlust/commits/master.atom
```

サポートする RSS や Atom は全て自動的に扱われます。より細かい設定が必要な場合は、新聞フォルダ (3.8 節 「Shimbun Folder」(15 ページ) を参照) を使ってください。

メッセージのキャッシュは持たないので、メッセージはフィードから無くなるとフォルダからも直ちに消失します。保存する必要がある場合はパイプフォルダ (3.13 節 「Pipe Folder」(21 ページ) を参照) と組み合わせてください。

```
|rss:http://lwn.net/headlines/newrss|+lwn
```

RSS や Atom 及び OPML をアクセスグループとして扱うことができ、その場合、関連するフィードはサブフォルダ内に現れます。

3.10 検索フォルダ

外部プログラムを用いて、検索式にマッチしたメッセージを集めた仮想的なフォルダを構成します。

書式:

‘[’ 検索式 ‘]’ [検索対象 [‘!’ 検索方式]]

検索式と検索対象の指定方法は、検索方式によって異なります。

3.10.1 対応している検索方式

デフォルトでは、以下の検索方式に対応しています。検索方式を省略した場合は、`elmo-search-default-engine` で指定された検索方式が使用されます。

3.10.2 namazu

namazu (`http://www.namazu.org/`) を使って、インデックスに登録された文書を検索します。

検索式は、namazu の検索式です。詳しくは、namazu に付属の文書を参照して下さい。

検索対象には、使用するインデックスを指定します。インデックスのあるディレクトリのパスをそのまま指定するか、以下の説明にある別名を指定する事が出来ます。インデックスを省略した場合は、elmo-search-namazu-default-index-path で指定された値が使用されます。

例:

```
[wanderlust]           -> デフォルトのインデックスから "wanderlust"
                          にマッチするものを探す
[semi flim]~/Mail/semi -> ディレクトリ "~/Mail/semi" にあるインデックス
                          から "semi flim" を探す
```

3.10.2.1 複数キーワードの入力

フォルダ名入力時に空白を入力したい場合、*C-q SPC* でできます。

3.10.2.2 インデックスのエイリアス名

インデックスにはエイリアス名 (別名) を定義できます。

```
(setq elmo-search-namazu-index-alias-alist
      '(("cache" . "~/.elmo/cache")
        ("docs" . "~/documents")))
```

の様に設定することによりインデックスの別名を定義できます。上記例であれば、

```
[wanderlust]cache
```

と入力すれば、~/.elmo/cache にある namazu インデックスを対象として、キーワード 'wanderlust' で検索を行います。

3.10.2.3 複数インデックスの指定

elmo-search-namazu-default-index-path, elmo-search-namazu-index-alias-alist の値としてインデックスのリストを指定することも可能です。リストを指定すると、複数のインデックスを対象とした検索となります。

例えば、

```
(setq elmo-search-namazu-index-alias-alist
      '(("all" . ("~/.elmo/cache" "~/documents"))
        ("cache" . "~/.elmo/cache")))
```

のように指定すると、

```
[wanderlust]all
```

と入力すれば、~/.elmo/cache と ~/documents のインデックスを対象として、キーワード 'wanderlust' で検索を行います。

3.10.3 grep

grep を使って、検索対象で指定されたディレクトリにあるファイルを検索します。

検索式には、grep の正規表現を指定します。検索対象とするディレクトリを省略する事は出来ません。

例:

```
[wanderlust]~/Mail/inbox!grep
    -> ディレクトリ "~/Mail/inbox" から
       "wanderlust" にマッチするものを探す

["[sr]emi"]~/Mail/semi!grep
    -> 検索式が ‘]’ を含む正規表現である場合は、
       ‘"’ で囲む必要があります
```

3.10.4 mu

mu (http://www.djcbsoftware.nl/code/mu/) を使って、文書を検索します。

例:（Mu cheatsheet より）

```
[Helsinki]
    -> messages about Helsinki (in message body, subject, sender, ...)

[to:Jack subject:jellyfish tumbleweed]
    -> messages to Jack with subject jellyfish containing the word tumbleweed

[size:2k..2m date:20091201..20093112 flag:attach from:bill]
    -> messages between 2 kilobytes and a 2Mb,
       written in December 2009 with an attachment from Bill

[subject:soc* flag:unread]
    -> サブジェクトが 'soc' で始まる (soccer, society, socrates, ...)
       未読のメッセージを探す

[mime:image/*]
    -> 画像が添付されたメッセージを探す

['foo bar']
    -> "foo bar" というフレーズを含むメッセージを探す
```

3.10.5 notmuch

notmuch (http://notmuchmail.org/) を使って、文書を検索します。

例（notmuch マニュアルより）

```
[term1]
    -> 'term1' を含むメッセージを探す

[-term1]
    -> 'term1' を含まないメッセージを探す

[term1 term2]
    -> 'term1' と 'term2' の両方を含むメッセージを探す

['foo bar']
    -> "foo bar" というフレーズを含むメッセージを探す
```

3.11 マルチフォルダ

複数のフォルダを仮想的に一つに見えるようにするフォルダです。

書式:

‘*’ フォルダ *1* [‘,’ フォルダ *2*] ... [‘,’ フォルダ *N*]

‘*’ の後に、‘フォルダ *1*, フォルダ *2*, . . . , フォルダ *N*’のように ‘,’ (コンマ) で区切って、一つに見えるようにしたいフォルダ群を指定します。

例:

```
*-fj.editor.xemacs,-fj.editor.mule,-fj.editor.emacs
-> -fj.editor.xemacs, -fj.editor.mule, -fj.editor.emacs が一つの
   フォルダとして見える。

*+inbox,-fj.rec.tv,%inbox
-> +inbox, -fj.rec.tv, %inbox が一つのフォルダとして見える。
```

3.12 フィルタフォルダ

指定した条件を満たすメッセージのみを含む仮想的なフォルダです。

書式:

‘/’ 条件 ‘/’ フォルダ

条件には、以下を書けます。

1. 部分フィルタ : ‘first:数字’, ‘last:数字’

   first: 全メッセージの先頭から数字の数だけメッセージを切り出します。last: 全メッセージの末尾から数字の数だけメッセージを切り出します。

   例:

   ```
   /last:10/-fj.os.linux -> -fj.os.linux の最近の 10 個のメッセージ
                            のみを表示するフォルダ
   /first:20/%inbox      -> %inbox の最初の 20 個を表示するフォルダ
   ```

2. 日付フィルタ : ‘since:日付’, ‘before:日付’

   since: 日付より最近のメッセージのみを取り出します。(日付を含みます) before: 日付より以前のメッセージのみを取り出します。(日付を含みません)

日付には以下が書けます。

```
yesterday ->  昨日
lastweek  ->  先週の今日
lastmonth ->  先月の今日
lastyear  ->  去年の今日
数字 daysago -> 数字 日前（e.x. 3daysago）
日-月の略名-年 ->  日付そのものの指定（ex. 1-Nov-1998）
```

例:

```
/since:3daysago/+inbox → 最近３日間の +inbox 中のメッセージ。
/before:yesterday/+inbox → 昨日より以前の +inbox 中のメッセージ。
```

3. フィールドフィルタ：‘フィールド名:文字列’

   メッセージのフィールドの中身が文字列にマッチするメッセージを取り出します。フィールド名、文字列に大文字小文字の区別はありません。

   例:

```
/from:teranisi/+inbox -> +inbox で、From: フィールドに
                         "teranisi" という文字列を含むメッセージのフォルダ
/body:なんとか/%inbox -> %inbox で、本文に "なんとか"
                         という文字列を含むメッセージのフォルダ
```

4. フラグフィルタ：‘flag:フラグ名’

   フラグ名で指定されたフラグを持つメッセージを取り出します。

   指定出来るフラグ名には、以下のものがあります。

```
unread    -> 未読
important -> 重要
answered  -> 返信済み
forwarded -> 転送済み
digest    -> 未読またはグローバルフラグ付き
any       -> 未読、返信済み、転送済み、グローバルフラグ付き
```

   また、グローバルフラグとして設定したフラグも同様に設定することが出来ます。グローバルフラグとは、wl-summary-set-flags (F キー) により付与できる任意の名前を持つフラグです。フラグの名前は自由に定義することができます。デフォルトでは、‘important’ フラグが用意されています。グローバルフラグが付与されたメッセージは、‘’flag’ フォルダのサブフォルダを閲覧することで、まとめて読むことができます。3.14 節 「Internal Folder」(22 ページ) を参照してください.

   例:

```
/flag:digest/%inbox      -> %inbox のうち、未読または重要な
                            メッセージのフォルダ
/flag:wl/+ML/Wanderlust -> +ML/Wanderlust のうち、wl という
                            グローバルフラグが付いたメッセージのフォルダ
```

5. 複合条件

   条件部分が ‘!’ で始まると否定の条件指定となります。複数の条件を ‘|’ で区切って指定すると、OR 条件となります。同様に、‘&’ で区切ると AND 条件を指定できます

(AND 条件は OR 条件よりも優先して評価されます)。さらに、‘(’, ‘)’, で囲うと条件式をグルーピング指定できます。

また、条件の省略記法として ‘tocc’ を用意しています。‘/tocc:xxxx/’ は、‘/to:xxxx|cc:xxxx/’ に展開されます。‘/!tocc:xxxx/’ は、‘/!to:xxxx&!cc:xxxx/’ に展開されます。

例:

```
/from:teranisi&!to:teranisi/+inbox
                        -> +inbox で From: フィールドに "teranisi" を含
み、
                           To: フィールドに "teranisi" を含まない
                           メッセージのフォルダ

/tocc:"Yuuichi Teranishi"/+inbox -> +inbox で、To: フィールドか
                            Cc: フィールドに "Yuuichi Teranishi" を含む
                            メッセージのフォルダ

/(from:yt|from:teranisi)&subject:報告/+inbox
                        -> +inbox で、From: フィールドが "yt" か
                           "teranisi" を含み、かつ Subject が "報告" を
含む
                           メッセージのフォルダ
```

注意

文字列部分に、空白文字、‘"’、‘/’、‘)’、‘|’、‘&’ を含みたい場合は、文字列全体を ‘"’ で括る必要があります。(‘"’ で括られている文字列中に ‘"’ を含む場合は、‘\’ でエスケープする必要があります)。これらの文字を含まない場合でも ‘"’ で括るのは問題ありません。

応用編

```
*%inbox,/from:teranisi/%inbox@server
       -> %inbox、および、
          %inbox@server の中で From フィールドが "teranisi" のメッセージ
群、
          をいっぺんに表示するフォルダ。

/last:100//to:teranisi/*+inbox,%inbox
       -> +inbox と %inbox 中のメッセージのうち、
          To: フィールドが "teranisi" にマッチする
          メッセージの最近の 100 個を表示するフォルダ。

/from:hogehoge//last:20//tocc:teranisi/%#mh/inbox@localhost
       -> %#mh/inbox@localhost の中で、To か Cc に "teranisi" が含まれる
          メッセージの最近の 20 個のうち、From が "hogehoge" のものを
          表示するフォルダ。
```

3.13 パイプフォルダ

フォルダ閲覧時に、自動的にメッセージの取り込みを実行するフォルダです。

書式:

‘|’ 取り込み元 ‘|’ 取り込み先

フォルダ閲覧時、**取り込み元**から**取り込み先**へ自動的にメッセージが移動します。POP を利用してメッセージをローカルにダウンロードして閲覧したい場合は、

```
|&username@popserver|+inbox
```

のように指定すると、フォルダの表示を更新するときに ‘&username@popserver’ から ‘+inbox’ へ、メッセージを自動的に取り込みます。

また、二つ目の ‘|’ を ‘|:’ にして、

‘|’ 取り込み元 ‘|:’ 取り込み先

とすると、取り込み元から取り込み先にメッセージをコピーします。次にそのフォルダにアクセスした際には差分だけを取り込みます。POP サーバにメールを残したまま、メッセージを ‘+inbox’ にコピーして閲覧したい場合は、

```
|&username@popserver|:+inbox
```

のようにします。

例:

```
|%inbox|%myinbox   -> %inbox から %myinbox へメッセージを取り込んで
                      閲覧するフォルダ。
|*&user@popserver1,&user@popserver2|+inbox
  -> &user@popserver1 と &user@popserver2 から +inbox へメッセージを
     取り込んで閲覧するフォルダ。
|-gnu.emacs.sources|:+sources
  -> -gnu.emacs.sources から +sources へメッセージをコピーして閲覧する
     フォルダ。
```

メッセージが移動した後、フック elmo-pipe-drained-hook が呼ばれますので、ダウンロード後に実行したい関数を登録しておくと良いでしょう。

3.14 内部フォルダ

Wanderlust の内部に保持したメッセージを閲覧するためのフォルダです。

書式:

```
‘’flag’ [‘/’ グローバルフラグ]
‘’sendlog’
‘’cache/00’ 〜 ‘’cache/1F’
```

‘’flag’ というフォルダは、**グローバルフラグ**で指定したフラグの付いたメッセージを集めた仮想フォルダです。

デフォルトで定義されている**グローバルフラグ**として ‘important’ フラグがあります。このフラグは、後述の重要マーク (‘$’) を付けることでメッセージに付加されます。重要マークをメッセージにつけておいて、あとから見直したいときなどに便利です。**グローバルフラグ**を省略した場合は、この ‘important’ フラグを指定したものと看做されます。

また、後述のサマリモードにおいて、自由にグローバルフラグを定義してメッセージに付加することも出来ます。5.1 節 「Usage of Summary Mode」(33 ページ) を参照してください.

このフォルダでメッセージを削除すると、そのメッセージについていた**グローバルフラグ** が削除されます。また、このフォルダにメッセージを追加すると、そのメッセージには**グローバルフラグ** が付けられます。

‘’sendlog’ は ~/.elmo/sendlog に記載されているメッセージのうちキャッシュしているものを集めた仮想フォルダです。自分自身への cc を付け忘れた時などに有用かもしれません。これを利用するには、wl-draft-use-cache を non-nil に設定して、送信したメッセージをキャッシュするようにしてください。

‘’cache/00’ 〜 ‘’cache/1F’ はネットワーク経由で読んだメッセージのキャッシュにアクセスするためのフォルダです。00 〜 1F には、キャッシュディレクトリ (~/.elmo/cache) のサブディレクトリ名を指定します。

3.15 ファイルフォルダ

ローカルファイルシステムのファイルを閲覧するためのフォルダです。一つのディレクトリを一つのフォルダとして扱います。

書式:

‘file:’ ディレクトリのパス

例:

```
file:~/work     -> ~/work
file:/etc       -> /etc
```

3.16 アクセスフォルダ

指定されたフォルダの配下のサブフォルダを仮想的に一つのフォルダとして扱えるようにするフォルダです。サブフォルダの増減は自動的に反映されます。

書式:

‘access:’ ルートフォルダ

例:

```
access:%INBOX -> %INBOX 以下の IMAP のメールボックスの全て
access:'cache -> 全ての 'cache フォルダ
```

4 フォルダモード

起動して最初に現れるのがフォルダモード (フォルダの一覧表示) です。

フォルダモードでは、読みたいフォルダの選択、購読フォルダの編集を行ないます。

4.1 読みたいフォルダの選択

4.1.1 使用方法 (TIPS)

4.1.1.1 新規数、未読数のチェック

フォルダモードの見ためはこんな感じになるはずです。(XEmacs ではもうちょっとかっこよく見えるはずです ;-))

```
[-]Desktop:14186/35580/67263
   受信箱:3/10/10
   ゴミ箱:2/7/10
   草稿:0/0/3
   送信済み:0/9/348
   [-]Emacsen:0/34/4837
     Wanderlust ML:0/0/558
     ELIPS ML:0/0/626
     tm (日本語) ML:0/0/821
     XEmacs ベータ:0/29/255
     Mew:0/0/998
     Mule-Win32:0/0/1491
     fj の Emacsen:0/5/88
```

各行は、

フォルダ名:未同期 (新規) 数/未読数/総数

を表示しています。チェックしたいフォルダの上にカーソルを合わせて *s* を押すと、これらの数を最新のものに更新します。たくさん新規メッセージがあると色が変わります。

フォルダモード全体は 'Desktop' というグループフォルダになります。グループフォルダはリターンキーで開閉できます。グループフォルダに対する操作は、そのグループフォルダに属する全ての子フォルダに対して適用されます。例えば、'[-]Emacsen' のところにカーソルを合わせて *s* を押すと、これに含まれる 6 つのフォルダの表示を最新のものに更新します。

4.1.1.2 フォルダの選択

フォルダの行にカーソルを置いてリターン (スペース) キーを押すとそのフォルダの内容を表示するサマリモードに移動します。

このとき、変数 `wl-stay-folder-window` が non-nil ならサマリに移動したときにフォルダバッファの右にサマリのバッファが現れます。

4.1.2 キーバインド

フォルダモードでのフォルダ選択に関するコマンドのキーバインドは以下の通りです。

SPC
RET 現在カーソルがある行のフォルダのサマリ表示へ移動します。prefix argument をつけると、スティッキーサマリに入ります。グループフォルダにカーソルがある場合は、グループフォルダの開閉を行ないます。アクセスグループでは、prefix argument つきでグループフォルダを開くと、内容を最新のリストに更新します。アクセスグループが階層構造になっている場合は再帰的に更新します。(wl-folder-jump-to-current-entity)

M-RET 現在カーソルがあるアクセスグループの内容を最新のリストに更新します。アクセスグループが階層構造になっている場合は再帰的に更新します。(wl-folder-update-recursive-current-entity)

w 新規ドラフトを用意します。(wl-draft)

W 現在カーソルがあるフォルダが NNTP フォルダなら、'Newsgroups:' フィールドを補ってドラフトを用意します。メーリングリストのリファイル先なら、メーリングリストのアドレスを推測して、'To:' フィールドを補ってドラフトを用意します。変数 wl-subscribed-mailing-list を設定しておく必要があります。(wl-folder-write-current-folder)

C-c C-o ドラフトバッファがあれば移動します。 複数のドラフトバッファが存在する場合は、次々と切り替えていきます。また、prefix argument をつけることにより、バッファが存在していない場合は、ドラフトフォルダからファイルを (存在すれば) 読み込みます。(wl-jump-to-draft-buffer)

s 現在カーソルがある行のフォルダの未同期メッセージ数を更新します。(wl-folder-check-current-entity)

S 現在カーソルがある行のフォルダのサマリを更新します。(wl-folder-sync-current-entity)

r s リージョンにあるフォルダの未同期メッセージ数を更新します。(wl-folder-check-region)

r S リージョンにあるフォルダのサマリを更新します。(wl-folder-sync-region)

Z ~/.addresses 等の状態を最新のものに更新します。(wl-status-update)

P ひとつ上の未読があるフォルダ (もしくはグループ) に移動します。(wl-folder-prev-unread)

N ひとつ下の未読があるフォルダ (もしくはグループ) に移動します。(wl-folder-next-unread)

p ひとつ上のフォルダへ移動します。(wl-folder-prev-entity)

n ひとつ下のフォルダへ移動します。(wl-folder-next-entity)

J 指定したフォルダへカーソルを移動します。(wl-folder-jump-folder)

I 現在カーソルがある行のフォルダに対して、wl-summary-incorporateにより、新着記事をプリフェッチします。グループフォルダにカーソルがある場合は、再帰的に実行します。(wl-folder-prefetch-current-entity)

c 現在カーソルがある行のフォルダのメッセージを全て読んだことにします。グループフォルダにカーソルがある場合は、再帰的に実行します。(wl-folder-mark-as-read-all-current-entity)

f 最初の未読があるフォルダのサマリへ移動します。(`wl-folder-goto-first-unread-folder`)

E ゴミ箱を空にします。(`wl-folder-empty-trash`)

F キューにあるメッセージを送信します。(`wl-folder-flush-queue`)

V 指定した条件と合致するメッセージのみを含む仮想フォルダ (フィルタフォルダ) へ移動します。(`wl-folder-virtual`)

? 指定した条件と合致するメッセージを含むフォルダを探します。(`wl-folder-pick`)

o 全ての未読があるフォルダが含まれるグループを開きます。(`wl-folder-open-all-unread-folder`)

x サマリバッファのマークを実行します。5.5 節 「Sticky Summary」(37 ページ) を参照してください. (`wl-execute-temp-marks`)

/ カーソル行のグループの開閉をトグルします。(`wl-folder-open-close`)

[全てのグループを開きます。(`wl-folder-open-all`)

] 全てのグループを閉じます。(`wl-folder-close-all`)

q Wanderlust を終了します。(`wl-exit`)

z Wanderlust をサスペンドします。(`wl-folder-suspend`)

M-s
C-x C-s 現在のフォルダの状態をセーブします。(`wl-save`)

M-t Wanderlust のオフラインモード/オンラインモードをトグルします。(`wl-toggle-plugged`)

C-t Wanderlust のサーバ・ポート別のオフライン/オンラインを変更します。(`wl-plugged-change`)

4.1.3 カスタマイズ変数

`wl-folders-file`
: 初期設定は `~/.folders`。購読するフォルダを記述するファイルの名前です。

`wl-folder-info-save`
: 初期設定は `t`。次に立ち上げ直したときに前回の未読数などの結果を使い回しするかどうか、です。

`wl-stay-folder-window`
: 初期設定は `nil`。Non-nil ならサマリに移動したときにフォルダバッファの右にサマリのバッファが現れます。

`wl-folder-window-width`
: 初期設定は 20。`wl-stay-folder-window` が non-nil のときに残すフォルダバッファのウィンドウの幅です。

`wl-folder-use-frame`
: 初期設定は `nil`。Non-nil ならフォルダ一覧用に新しいフレームを開きます。

wl-folder-many-unsync-threshold
: 初期設定は 70。未同期数がたくさんかどうかの閾値。この値を越えると色が変わります。

wl-highlight-folder-by-numbers
: フォルダバッファにおける各行のハイライトの形式を指定します。初期値は t で、行全体にメッセージ数に応じた色を付けます。nil ではフォルダの状態に応じた色を付けます。また、数字 (例えば 1) にしておくと、メッセージ数とフォルダの状態の両方に応じたハイライトが行なわれます。

wl-folder-desktop-name
: 初期設定は 'Desktop'。トップのグループの名前です。

wl-folder-petname-alist
: 初期設定は nil。フォルダの本名とあだ名の cons のリストです。

wl-folder-access-subscribe-alist
: 初期設定は nil。アクセスグループのリスト取得で自動的に subscribe、もしくは unsubscribe するフォルダを指定します。

  リストの各要素は

  ```
  (アクセスグループの正規表現 . (購読フラグ フォルダの正規表現 ...))
  ```

  となっています。**購読フラグ**が non-nil の場合は**フォルダの正規表現**にマッチしたフォルダのみ表示し、**購読フラグ**が nil の場合は**フォルダの正規表現**にマッチしたフォルダは表示しません。ただし、**購読フラグ**が non-nil でも既に unsubscribe 済のフォルダは表示しません。また、**フォルダの正規表現**は複数個記述できます。

  例:

  ```
  '(("^-fj$" . (t   "^-fj\\.\\(comp\\|editor\\|mail\\)"
                    "^-fj\\.\\(net\\|news\\|os\\|rec\\)"))
    ("^-$" . (t   "^-\\(fj\\|tnn\\|japan\\|gnu\\|comp\\)"))
    ("^\\+ml$" . (nil "^\\+ml$" "^\\+ml/tmp")))
  ```

wl-folder-hierarchy-access-folders
: 階層構造に作成するアクセスグループの正規表現からなるリストです。

  例えば、以下のように wl-folder-hierarchy-access-folders を設定すると、

  ```
  (setq wl-folder-hierarchy-access-folders
        '("^-[^\\.]*$" "^-comp.unix$" "^-comp.unix.bsd$"))
  ```

  次のような階層構造になったアクセスグループを作ります。

```
[-]-:912/912/3011
  [-]-fj:674/674/1314
    -fj.comp.announce:0/0/2
    -fj.comp.dev.cdrom:0/0/0
    ...
  [+]-japan:238/238/1688
  [-]-comp:0/0/4
    [-]-comp.unix:0/0/0
      -comp.unix.admin:0/0/0
      -comp.unix.dos-under-unix:0/0/0
      -comp.unix.programmer:0/0/0
      [-]-comp.unix.bsd:0/0/23
        -comp.unix.bsd.freebsd.announce:0/0/0
  ...
```

ただし、'-' を開いただけでは 1 つ目の階層 ('-fj', '-japan', '-tnn', ...) を作るだけです。2 つ目の階層 ('-fj.comp.announce', ..., '-comp.unix', ...) 以下のフォルダは、あらためてそのグループを開いた時に作られます。

4.2 購読フォルダの編集

前に述べた通り、購読するフォルダは ~/.folders に保持されますが、~/.folders を直接編集するまでもなく、フォルダモードからもフォルダの追加/削除/グループの定義などを編集できます。

4.2.1 使用方法 (TIPS)

4.2.1.1 フォルダの追加

m a で新規に購読するフォルダを追加します。存在しないフォルダを入力すると、新規に作成してよいか聞いてきます。*m g* でグループが追加されます。このグループにフォルダを追加するには、まずこのグループを開いた状態にします。そして次の行で挿入のコマンドを実行すると、そのグループにフォルダを追加できます。

4.2.1.2 フォルダの編集

C-k でカット (切り取り)、*C-y* でペースト (張り付け) ができます。つまり、通常の文書編集の行編集と同じようにフォルダの位置を変更可能です。

4.2.1.3 マルチフォルダの作成方法

1. *m q*で wl-fldmgr-cut-entity-list を消します。
2. *C-k* でフォルダを切り取り、または *M-c* でフォルダをコピーします。
3. *m m* を実行するとマルチフォルダが作成されます。

4.2.1.4 あだ名 (petname) やフィルタの削除

あだ名やフィルタを削除するためには、ミニバッファで ""(*NULL*) を入力します。

4.2.1.5 空グループへの追加

m g などでグループを作成した後、このグループにフォルダを追加するには、まずこのグループを開いた状態にします。そして次の行で追加や挿入のコマンドを実行するとグルー

プにフォルダを追加できます。一方、上の行のグループが閉じた状態のときは、そのグループと同じレベルに挿入されます。言葉で説明するのは難しいので実際に実行してみると良いでしょう。つまり、カーソル位置より上にあるグループの開閉状態により挿入される位置が異なるのです。

4.2.1.6 セーブ時の言語コード

`wl-folders-file` をセーブするときは `wl-mime-charset` の言語コードになります。

4.2.1.7 フィルタフォルダの作成

カーソル上のフォルダをフィルタ付きに変更するには、フィルタフォルダ作成コマンドを用います。もしカーソル上のフォルダを残したまま新たにフィルタフォルダを作成する場合は、まずコピーしてからフィルタを作成し、その後コピーしたフォルダを挿入します。フィルタフォルダ作成時には一度に複数の (多段の) フィルタが指定できます。先頭のフィルタをすべて削除するためには、“”(*NULL*) を入力します。

4.2.1.8 フォルダの並び替え

グループ内のフォルダを並び替える際には、`wl-fldmgr-sort-function` で指定した関数を用います。初期設定は `wl-fldmgr-sort-standard` になっています。これはアルファベット順に並び替え、グループを最初にする関数です。並び替えの対象になるのは指定したグループのみで、下位のグループは並び替えません。つまり、再帰的には行わないということです。

4.2.1.9 アクセスグループ内の表示しないフォルダの指定

アクセスグループを開くと通常全てのフォルダを表示しますが、表示しないフォルダを指定することもできます。以下の操作はアクセスグループ内でのみ有効です。

コマンド `wl-fldmgr-unsubscribe` (*u*) はカーソル位置のフォルダの表示 (subscribe)・非表示 (unsubscribe) 設定をトグルします。これに対して `wl-fldmgr-unsubscribe-region` (*U*) は指定範囲のフォルダを非表示にします。

`wl-fldmgr-unsubscribe` はトグルしますが、`wl-fldmgr-unsubscribe-region` だと通常トグルにならないことに注意して下さい。リージョンの場合トグルにするよりどちらかに設定させる方が使いやすいと考え、このようにしています。しかし、上記 2 つの関数とも prefix argument の値が正ならフォルダを非表示、負なら表示、0 ならトグルします。

またキーには割り当てていませんが、フォルダを表示に設定するだけの `wl-fldmgr-subscribe` と `wl-fldmgr-subscribe-region` も用意しています。使用する場合は適当なキーに割り当ててください。

さらに、アクセスグループ内で `wl-fldmgr-cut` と `wl-fldmgr-cut-region` を実行すると、それぞれ `wl-fldmgr-unsubscribe` と `wl-fldmgr-unsubscribe-region` を実行したのと同じ効果が得られます。違いは cut の場合は画面からも消去することです。

4.2.1.10 アクセスグループ内の操作

アクセスグループ内でも削除や挿入を行えます。といっても実際には unsubscribe するかどうかを設定しているだけなので、変更が可能なのは、当然そのアクセスグループに属しているフォルダだけです。つまり、挿入は subscribe し、削除は unsubscribe したことと同じです。[1] 従って、Wanderlust 以外でフォルダを変更した場合、そのフォルダを実際に追

[1] 現在は、指定範囲を表示しないフォルダにするとき、指定範囲の削除を行った方が高速です。

加したり削除したりするには、アクセスグループを *C-u RET* で更新してください。4.1 節「Selecting Folder」(24 ページ) を参照してください.

また、挿入/削除/並び替えなどを行ったあとのフォルダの並びは保持されます。もし `wl-force-fetch-folders` を設定するか、*C-u RET* でグループを開くかすると、存在しないフォルダは削除され、新たに作成されたフォルダは先頭に追加されます。

4.2.2 キーバインド

フォルダの編集に関する主なコマンドのキーバインドは以下の通りです。一応 *m* で始まるキーに全て割り当てて、主な機能のみ 1 ストロークで使用できるようにしています。

m a フォルダ一覧にフォルダを追加します。存在しないフォルダを入力した場合には確認の後、新規に作成します。(`wl-fldmgr-add`)

+
m g グループを作成します。(`wl-fldmgr-make-group`)

m A アクセスグループを作成します。(`wl-fldmgr-make-access-group`)

m d フォルダの実体とその msgdb を削除します。nntp などの削除できないフォルダでは msgdb のみ削除します。(`wl-fldmgr-delete`)

R
m R フォルダ、もしくはグループ名を変更します。フォルダを変更する場合は msgdb のパスも変更します。(`wl-fldmgr-rename`)

***
m m マルチフォルダを作成します (コピー、削除されたフォルダを結合します)。(`wl-fldmgr-make-multi`)

|
m f フィルタフォルダを作成します (選択したフォルダに filter を付けます)。(`wl-fldmgr-make-filter`)

M-c
m c フォルダをコピーする (グループはコピー不可)。(`wl-fldmgr-copy`)

M-w
m W 指定範囲のフォルダのコピー。(`wl-fldmgr-copy-region`)

C-k
m k フォルダの削除 (切り取り)。フォルダの実体は削除しません。(`wl-fldmgr-cut`)

C-w
m C-w 指定範囲のフォルダの削除 (切り取り)。(`wl-fldmgr-cut-region`)

C-y
m y コピー、削除したフォルダ (cut-list) を挿入する (ペースト、貼り付け)。(`wl-fldmgr-yank`)

m p フォルダにあだ名 (petname) を付ける。(`wl-fldmgr-set-petname`)

m q コピー、削除したフォルダ情報 (cut-list) を消す。(`wl-fldmgr-clear-cut-entity-list`)

m s グループ内のフォルダを並び替える (そのグループ階層のみ)。(`wl-fldmgr-sort`)

m C-s wl-folders-file にセーブする。(`wl-fldmgr-save`)

[以下の操作はアクセスグループに対してのみ有効]

u
m u フォルダの表示/非表示の設定。(`wl-fldmgr-unsubscribe`)

U
r u 指定範囲にあるフォルダの表示/非表示の設定。(`wl-fldmgr-unsubscribe-region`)

l
m l 現在有効なフォルダのみ一覧表示する。(`wl-fldmgr-access-display-normal`)

L
m L 非表示のフォルダも含めて全てのフォルダを一覧表示する。(`wl-fldmgr-access-display-all`)

4.2.3 カスタマイズ変数

`wl-interactive-save-folders`
: 初期設定は `t`。フォルダ変更の操作を行った場合、Wanderlust 終了時もしくは Emacs 終了時にセーブするか確認を行います。`nil` だと確認なしでセーブします。

`wl-fldmgr-make-backup`
: 初期設定は `t`。Non-nil なら、セーブする際に `~/.folders.bak` にバックアップを取ります。

`wl-fldmgr-sort-function`
: 初期設定は `wl-fldmgr-sort-standard`。sort 時に使用する関数を指定します。初期設定は、アルファベット順に並べグループは最初にする関数です。(`wl-fldmgr-sort-group-first` が non-nil の場合)。

`wl-fldmgr-sort-group-first`
: 初期設定は `t`。Non-nil なら、`wl-fldmgr-sort-standard` で並び替えるときグループを最初に置く。`nil` なら、グループも含めてアルファベット順に並び替えます。

`wl-folder-check-async`
: 初期設定は `t`。Non-nil の場合、フォルダの新規メッセージ数のチェックを非同期に行ないます。ニュースグループのチェックが大幅に速くなります。

`wl-folder-check-fast`
: 初期設定は `nil`。Non-nil の場合、フォルダの新規メッセージ数のチェックを行うたびにフォルダの表示を更新しません。

`wl-folder-notify-deleted`
: 初期設定は `nil`。Non-nil ならメッセージが削除されたときにフォルダモードで未読数をチェックすると負の値が表示されます。値が `sync` ならば、メッセージが削除されていたときにフォルダの内容と同期を取ります。

`wl-fldmgr-add-complete-with-current-folder-list`
: 初期設定は `nil`。Non-nil なら `wl-fldmgr-add` の補完候補を `elmo-folder-list-subfolders` で得ます。

4.2.4 その他

フォルダの編集に関して、以下のような仕掛けや制限があります。

1. 削除やコピーを行うと、スタックのように変数 `wl-fldmgr-cut-entity-list` にたまっていきます。ペースト (張り付け) は削除やコピーした単位で (リージョンでコピーしたなら、そのリージョン内のフォルダを一度に) 吐き出します。
2. ‘`Desktop`’ グループの削除、コピーはできません。また、‘`Desktop`’ 外への挿入はできません。
3. グループのコピーはできません。
4. アクセスグループ内での操作は、そのアクセスグループに実際に属しているフォルダのみが対象になります。
5. 全体で同じ名前のグループは複数作れません。
6. 同じグループ (階層) に同じフォルダは作れません。階層が違えば複数作れます。異なるフォルダに同じあだ名は指定できません。

5 サマリモード

フォルダモードで読みたいフォルダを選択すると、サマリモードに移動します。サマリモードは、メッセージの一覧を表示するモードです。

5.1 使用方法 (TIPS)

5.1.1 サマリの表示内容

サマリモードには、以下のようにメッセージの一覧が表示されます。

```
377  09/16(水)11:57 [+1: 北目さん       ] Bug?
381  09/17(木)00:16 [+3: 奥西さん       ] elmo-lha.el -- LHA interface
384  09/17(木)01:32 [+1: てらにし       ] wl-0.6.2
389 N09/18(金)01:07 [+2: てらにし       ] wl-0.6.3
```

各行は、メッセージの

メッセージ番号、一時的マーク、永続的マーク、日付け、差出人、サブジェクト

を順に表示しています。表示形式の変更については、サマリ行の形式の節を参照して下さい。5.6 節 「Summary View」(38 ページ) を参照してください.

メッセージ番号はそのフォルダ中にあるメッセージに対するラベルです。News フォルダでは article 番号、IMAP フォルダでは UID、MH フォルダではファイル名になります。

一時的マーク/**永続的マーク**については、あとで詳しく説明します。

日付けは、‘**月/日 (曜日) 時:分**’ のように表示されます。曜日を日本語ではなく英語表記したい場合は、`wl-summary-weekday-name-lang` を ‘`en`’ に設定してください。

差出人は、スレッドの深さ分インデントされて表示されます。**差出人**は、アドレス帳にあだ名があればあだ名で表示します。あだ名表記を止めたい場合は、`wl-use-petname` を `nil` に設定してください。

差出人の部分にある ‘`+2`’ のような数字は、そのメッセージに対する返事の数を表します。例えば ‘`+2`’ なら返事が 2 通あることを示します。

サブジェクトは、メッセージの ‘`Subject:`’ フィールドです。同じスレッドで、かつ親と同じ ‘`Subject:`’ を持つメッセージの場合には、**サブジェクト** を表示しません。メーリングリストなどの通し番号表示は無視します。‘`Subject:`’ が無い場合には、変数 `wl-summary-no-subject-message` の内容を表示します。

5.1.2 一時的マーク

一時的マークは、メッセージを操作するためマークです。

一時的マークには、‘`*`’, ‘`d`’, ‘`D`’, ‘`o`’, ‘`O`’, ‘`i`’, ‘`~`’ があります。

‘`*`’ まとめ処理用マークです。*m* で始まるコマンドで ‘`*`’マークのついたメッセージ対して一括した処理ができます。

‘`d`’ 処分するメッセージに付くマークです。*d* を押すと付きます。

‘`D`’ 削除するメッセージに付くマークです。*D* を押すと付きます。

‘`o`’ リファイルするメッセージに付くマークです。*o*を押すと、リファイル先を聞いて来ます。それに答えると、リファイル先のフォルダ情報が付け加わります。

‘O’ コピーするメッセージに付くマークです。*O*を押すと、コピー先を聞いて来ます。それに答えると、コピー先のフォルダ情報が付け加わります。

‘i’ プリフェッチ予約されたメッセージに付くマークです。*i*を押すと付きます。

‘~’ 再送予約されたメッセージに付くマークです。~を押すと、再送先アドレスを聞いて来ます。それに答えると、再送先のアドレス情報が付け加わります。

x を押すと、それぞれの一時的マークに対応したアクションを実行します。

5.1.3 永続的マーク

永続的マークは、メッセージの状態を示すマークです。

永続的マークには、‘!’, ‘N’, ‘n’, ‘U’, ‘u’, ‘A’, ‘a’, ‘F’, ‘f’, ‘$’ があります。

‘N’ 新規メッセージに付きます。

‘n’ 新規メッセージに付きます。‘N’ とは異なり、‘n’ のメッセージはキャッシュされています。

‘U’ 未読メッセージに付きます。

‘u’ 未読メッセージに付きます。‘U’ とは異なり、‘u’ のメッセージはキャッシュされています。

‘!’ 既読メッセージに付きます。マーク無しとは異なり、‘!’ のメッセージはキャッシュされていません。

‘A’ 返信済みメッセージに付きます。

‘a’ 返信済みメッセージに付きます。‘A’ とは異なり、‘a’ のメッセージはキャッシュされています。

‘F’ 転送済みメッセージに付きます。

‘f’ 転送済みメッセージに付きます。‘F’ とは異なり、‘f’ のメッセージはキャッシュされています。

‘$’ グローバルフラグの設定されたメッセージに付きます。このマークは Emacs を終了しても保存されるため、あとで返事を書きたい場合など、覚えておきたい重要なメッセージに付けておくと便利です。‘$’ の付いたメッセージは、(実際のメッセージが消されたとしても) ‘’flag’ フォルダで閲覧できます。グローバルフラグは、*$* または、*F* で付けることが出来ます。

‘なし’ 既読メッセージにはマークが存在しません。

‘N’, ‘U’, ‘!’, ‘A’, ‘F’ は、そのメッセージがキャッシュされていないことを示しています。これらのマークが付いていない場合、つまり、そのメッセージがキャッシュされている場合は、ネットワークに接続されていなくても IMAP フォルダのメッセージや NNTP フォルダのニュース記事を読むことができます。

永続的マークのうち、`wl-summary-expire-reserve-marks` で指定したマークの付いたメッセージは、後で説明する (Wanderlust の機能としての) expire の対象から除外されます。第 9 章 「Expire and Archive」(71 ページ) を参照してください.

5.1.4 メッセージの読み進めかた

サマリモードでは、基本的にスペースキーを押すだけでメッセージを読み進めることができます。サマリの表示内容をフォルダの最新の状態に合わせる (同期する) には、*s* を押します。

N で次の未読、*n* で次のメッセージを表示します。*j* を押すと現在表示中のメッセージのバッファに移動します。マルチパートの操作はメッセージバッファに移動してから行います。第 6 章 「Message」(53 ページ) を参照してください.

5.1.5 メッセージ番号を詰める

サマリで *M-x wl-summary-pack-number* とすることでメッセージ番号を詰めることができます。ただし対応しているフォルダの種類は MH Folder, News Spool Folder, Maildir Folder のみです。

5.2 スレッドの操作

例えば、

```
384  09/17(木)01:32 [+1: てらにし        ] wl-0.6.2
```

は一つのスレッド (一つの話題の流れ) を示しています。この行で / を押すと、スレッドが開いて次のような表示になります。

```
384  09/17(木)01:32 [ てらにし           ] wl-0.6.2
388  09/17(木)22:34  └[ 村田さん           ]
```

(388 番のメッセージは 384 番のメッセージに対する返事です。) もう一回 / を押すと、スレッドを閉じます。prefix argument つきで / を押すと、全ての子スレッドを開きます。

[でサマリの全てのスレッドを開き、*]*で全てのスレッドを閉じます。

そのスレッドに属するメッセージに対して一括して処理を行うためには *t* から始まるコマンドを用います。5.8 節 「Key Bindings of Summary」(40 ページ) を参照してください.

5.2.1 スレッドの繋ぎなおし

手動でスレッドの繋ぎなおしができます。サマリで該当するメッセージにカーソルを合わせて *M-w* (wl-summary-save-current-message) し、新しい親メッセージのところで *C-y* (wl-summary-yank-saved-message) を実行してください。

5.3 キャッシュ

5.3.1 キャッシュファイル

ネットワーク経由のメッセージはキャッシュされます。これにより、ネットワーク流量が節約され、またオフライン操作が可能となります。キャッシュは、ディレクトリ ~/.elmo/cache に保持されます。キャッシュをクリアするには、*M-x elmo-cache-expire-by-size* と入力してください。最近使われてないキャッシュから順に、一定のディスク容量になるまで消します。

5.3.2 バッファキャッシュと先読み機能

一度読んだメッセージは一定数バッファに保持されます。これにより、次にそのメッセージを表示する際の動作を高速にします。保持するバッファの数は wl-message-buffer-cache-size で指定します。

また、メッセージを読んでいる間に次のメッセージを取得しておく先読み機能があります。

この先読みするフォルダは次の 2 つの変数により指定できます。

`wl-message-buffer-prefetch-folder-type-list`
: 初期設定は `'(imap4 nntp)`。先読み機能を有効にしたいフォルダタイプのシンボルリストを指定します。初期設定では、IMAP4 と NNTP フォルダで先読み機能が有効になります。もし localdir と IMAP フォルダが混在したマルチフォルダでは IMAP のメッセージだけが先読みの対象となります。この変数は `wl-message-buffer-prefetch-folder-list` よりも優先されます。すべてのフォルダで先読み機能を有効にしたい場合には、t を指定します。

`wl-message-buffer-prefetch-folder-list`
: 初期設定は `nil`。先読み機能を有効にするフォルダをフォルダ名 (正規表現) のリストで指定します。

`wl-message-buffer-prefetch-depth`
: 初期設定は 1。先読み機能するメッセージの数。

`wl-message-buffer-prefetch-idle-time`
: 初期設定は 1 (単位:秒)。先読み機能する時間間隔。

`wl-message-buffer-prefetch-threshold`
: 初期設定は 30000 (bytes)。この値を越えるサイズのメッセージは、先読みしません。`wl-message-buffer-prefetch-threshold`を `nil` に設定すると、先読みするメッセージのサイズをチェックしません。

`wl-auto-prefetch-first`
: 初期設定は `nil`。Non-nil であればフォルダに移動した時に最初のメッセージを自動的に先読みします。

5.4 自動リファイル

メッセージのヘッダ情報から任意のフォルダへ振り分ける自動リファイル機能が *C-o* (`wl-summary-auto-refile`) で使用できます。自動リファイルは msgdb の overview 情報を元に振り分けします。標準では 'From:' 'Subject:' 'To:' 'Cc:' が含まれています。これ以外の拡張項目で振り分けたいときには、

```
(setq elmo-msgdb-extra-fields
      '("x-ml-name"
        "reply-to"
        "sender"
        "mailing-list"
        "newsgroups"))
```

として拡張項目を msgdb に含めるようにしてください。既に取得したメッセージの拡張項目を取り込むには *s all* などで msgdb を作り直す必要があります。

次に振り分けするルールを設定します。自動振り分けは，変数 `wl-refile-rule-alist` の値に基づいて行われます。`wl-refile-rule-alist` は、

```
（フィールド（正規表現 . 振り分け先）
             （正規表現 . 振り分け先）
             ...）
```

のようなルールのリストとなっています。各ルールは、『**フィールド**』の値が『**正規表現**』にマッチするときに『**振り分け先**』に振り分けるという意味を持ちます。前方に指定されたルールが優先されます。

『**フィールド**』部分には、フィールド名の文字列を指定します。また、フィールド名の文字列のリストを指定することもできます。この場合、リストのいずれかのフィールドの値がマッチしたときに振り分けを行います (OR 条件となります)。

『**正規表現**』にはフィールドの値にマッチさせる正規表現文字列を指定します。『**振り分け先**』には振り分け先のフォルダ名の文字列を指定します。『**振り分け先**』部分には、再びルールを書くこともでき、その場合そのルールのフィールドの値の条件と現在のルールのフィールドの値の条件が満たされたときに振り分けを行ないます (AND 条件となります)。

また、『**正規表現**』でマッチした部分文字列を用いて、『**振り分け先**』を指定することもできます。実際には次のようにして参照します。

'`\&`' マッチした文字列全体を参照します。

'`\`*N*' *N* 番目の '`\(...\)`' にマッチした文字列を参照します。(*N* は数字)

以下は `wl-refile-rule-alist` の一例です。

```
(setq wl-refile-rule-alist
      '(("x-ml-name"
         ("^Wanderlust" . "+wl")
         ("^Elisp" . "+elisp"))
        (("To" "Cc")
         ("\\([a-z]+\\)@gohome\\.org" . "+\\1"))
        ("From"
         ("me@gohome\\.org" . ("To" ("you@gohome\\.org" .
                                     "+from-me-to-you"))))))
```

C-o (`wl-summary-auto-refile`) で条件にマッチしたメッセージにリファイルマークが付加されます。*x* でリファイルを実行します。

`wl-summary-auto-refile-skip-marks` を設定することにより、自動リファイルの対象とならないメッセージを指定することができます。標準では '`N`' '`U`' '`!`' が設定されており、これらの永続的マークのついたメッセージは、自動リファイルしません。つまり標準では未読のメッセージを自動リファイルしないことになります。すべてのメッセージを自動リファイルの対象にするには、

```
(setq wl-summary-auto-refile-skip-marks nil)
```

の様に `wl-summary-auto-refile-skip-marks` を `nil` にします。

5.5 スティッキーサマリ

スティッキーサマリは、*q* で現在のサマリを終了してもバッファが消えない、その名の通りスティッキー (しつこい) サマリです。

フォルダモードから *Shift RET*、もしくはサマリ から *G* してサマリに入ると、スティッキーサマリが作られます。また、通常のサマリで *M-s* (`wl-summary-stick`) を実行すると現在のサマリがスティッキーサマリになります。

スティッキーサマリのバッファ名は、'`Summary:`**フォルダ名**' となります。 *C-x b* (`switch-to-buffer`) などで、適当にバッファを切替えればいつでもスティッキーサマリ

を参照できます。また、サマリモードで *C-c C-n* (wl-summary-previous-buffer) や *C-c C-p* (wl-summary-next-buffer) するとサマリを巡回できます。

スティッキーサマリでは、*q* や *g* の際に元のバッファを残します。スティッキーサマリを終了するには *C-u q* でサマリを抜けるか、*C-u g* で別のサマリに移動して下さい。その他の操作に関しては通常のサマリと同様です。

変数 wl-summary-always-sticky-folder-list にフォルダ名 (正規表現) のリストを設定することにより、フォルダ移動時に自動的にスティッキーサマリとすることもできます。

5.6 サマリ行の形式

サマリ行の形式は、ある程度自由に変更できます。

サマリ行の形式を変更するには wl-summary-line-format を設定します。wl-summary-line-format-spec-alist で定義された書式を用います。以下に例を示します。

```
;; 番号 一時マーク 永続マーク 日付 枝 [ (子の数) 差出人 ] 件名
(setq wl-summary-line-format "%n%T%P%M/%D(%W) %t%[%17(%c %f%) %] %s")
```

数字は桁数を表します。負の数の場合は、右詰めになります。また、最初の数字を '0' にすると、桁数が足りない場合に ' ' の代わりに '0' を追加して桁数を揃えます。

例:

```
%5n   -> '1    '
%-05n -> '00001'
```

デフォルトの wl-summary-line-format-spec-alist で用意されている書式記号のうち主なものは以下の通りです。

```
%n  メッセージ番号
%T  一時的マーク
%P  永続的マーク
%Y  年
%M  月
%D  日
%W  曜日
%h  時
%m  分
%t  スレッドの枝
%[  [ (繋ぎ直した子の場合は <)
%]  ] (繋ぎ直した子の場合は >)
%f  差出人
%s  件名
%S  サイズ
%c  +子の数: (スレッドを開いた時のみ表示)
%C  [+子の数] (スレッドを開いた時のみ表示)
%#  メーリングリストの情報 ('(' ML 名 [ ' ' ML 番号 ] ')')
%l  メーリングリストでの番号
%@  最初の MIME パートが multipart/mixed の場合に '@'
%~  直前の項目が空でない場合に ' '
```

wl-summary-line-format に一時的マーク ('%T') と永続的マーク ('%P') を含める場合には、必ず一定の位置に表示されるように定義しなければなりません。例えば、スレッドに

よって長さが変わる ‘%t’ のうしろに ‘%T’ や ‘%P’ を表示するようにしてしまうと、正しくマーク処理を行なえなくなる可能性があります。

また，‘%数字 (’, ‘%)’ で囲んだ範囲は，’数字’ の桁数になります (入れ子も可)。複数の書式記号に対する幅を設定するときに使います。例えば、上記例の

```
%17(%c %f%)
```

という指定は、『「子の数」と「差出人」をつなげた文字列を 17 桁にする』という意味です。

また、wl-folder-summary-line-format-alist を設定すると、サマリ行の形式をフォルダ毎に指定することができます。以下の例のように、フォルダ名の正規表現とサマリ行の書式を指定して下さい。

```
(setq wl-folder-summary-line-format-alist
      '(("^%" . "%T%P%M/%D(%W)%h:%m %t%[%17(%c %f%) %] %s")
        ("^+" . "%n%T%P%M/%D %[ %17f %] %t%C%s")))
```

5.6.1 差出し人の表示形式について

書式記号 ‘%f’ は、wl-summary-from-function で指定した関数が返す値を表示します。関数 wl-summary-default-from を用いる場合 (デフォルト)、通常は差出人を表示しますが、フォルダ名が wl-summary-showto-folder-regexp にマッチし、かつ差出人が自分であるメッセージについては、宛先を表示します。また、 wl-use-petname が Non-nil の場合にはペットネームを用いて表示します。

例えば、‘+backup’ では自分が出したメッセージについては宛先を表示したい場合、次のように設定します。

```
(setq wl-summary-showto-folder-regexp "^\\+backup$")
```

5.7 一時マークとその処理

一時マークとそれに対応する処理手順を wl-summary-mark-action-list で定義することができます。初期設定では、refile (‘o’), copy (‘O’), dispose (‘d’), delete (‘D’), prefetch (‘i’), resend (‘~’) が定義されています。

wl-summary-mark-action-list の各要素は

```
(‘MARK’ ‘SYMBOL’
 ‘ARGUMENT-FUNCTION’ ‘SET-MARK-FUNCTION’ ‘EXEC-FUNCTION’
 ‘FACE’ ‘DOC-STRING’)
```

の組です。ここで ‘MARK’ は定義する一時マークの文字列で、‘SYMBOL’ は定義するアクションの名前です。‘ARGUMENT-FUNCTION’ は次に説明する ‘SET-MARK-FUNCTION’ に与える引数を設定するための関数で、その引数としては

```
(‘ACTION’ ‘NUMBER’)
```

が与えられます。‘ACTION’ には ‘SYMBOL’ と同じものが入り、‘NUMBER’ にはメッセージ番号が入ります。‘SET-MARK-FUNCTION’ はマークを設定する際に呼ばれる関数で、引数は

```
(‘NUMBER’ ‘MARK’ ‘DATA’)
```

です。‘NUMBER’ は対象となるメッセージの番号、‘MARK’ は一時マークの文字列、‘DATA’ は ‘ARGUMENT-FUNCTION’ で与えられるものです。

'EXEC-FUNCTION' はアクションを実行する際に呼ばれる関数で、その引数は 'MARK-INFO' からなるリストです。ここで 'MARK-INFO' は

('NUMBER' 'MARK' 'DATA')

からなるリストです。'FACE' はハイライトに用いる face です。

5.8 キーバインド

サマリモードのキーバインドは以下の通りです。

SPC 現在カーソルがある行のメッセージをメッセージバッファに表示します。(wl-summary-read)

. 現在カーソルがある行のメッセージをデフォルトの表示形式で再表示します。prefix argument つきならばキャッシュが存在しても無視して再読み込みし直します。*C-u C-u .* のように 2 回の *C-u* を付けて実行した場合、現在の表示形式を維持して再読み込みし直します。(wl-summary-redisplay)

< 最初のメッセージを表示します。(wl-summary-display-top)

> 最後尾のメッセージを表示します。(wl-summary-display-bottom)

BS
DEL 前のページを表示します。(wl-summary-prev-page)

RET カーソル行のメッセージが表示中であればメッセージを一行上にスクロールします。表示中でなければ、表示します。(wl-summary-next-line-content) prefix argument をつけて実行した場合には、逆にメッセージを一行下にスクロールします。(wl-summary-prev-line-content) また、prefix argument が数値であった場合には、指定したメッセージ番号を持つメッセージへジャンプします。

-
M-RET カーソル行のメッセージが表示中であればメッセージを一行下にスクロールします。表示中でなければ、表示します。(wl-summary-prev-line-content)

/ カーソル行のスレッドの開閉をトグルします。prefix argument つきならば、カーソル行の子スレッドを全て開きます。(wl-thread-open-close)

[全てのスレッドを開きます。(wl-thread-open-all)

] 全てのスレッドを閉じます。(wl-thread-close-all)

g 違うフォルダに移動します。(wl-summary-goto-folder)

c 全てのメッセージを読んだことにします。(wl-summary-mark-as-read-all)

a 現在カーソルがある行のメッセージへの返事用のドラフトを用意します。(wl-summary-reply)

A 現在カーソルがある行のメッセージへの返事用のドラフトを本文を引用して用意します。(wl-summary-reply-with-citation)

C 現在のカーソルがある行のメッセージが自分が出したニュース記事の場合、その投稿をキャンセルします。(wl-summary-cancel-message)

E 現在カーソルがある行のメッセージの内容を持つドラフトを用意します。もし、prefix argument つきで実行すれば Supersedes メッセージを作成します。ただし、NNTP フォルダでかつ自分が投稿したメッセージに限ります。(wl-summary-reedit)

M-E カーソル行のメッセージが、自分のところに返ってきたエラーメッセージならば、もう一回送るためのドラフトを用意します。(wl-summary-resend-bounced-mail)

f 現在カーソルがある行のメッセージを他の人に転送するドラフトを用意します。(wl-summary-forward)

$ 'important' フラグをつけます。既に 'important' フラグがあればフラグを取り除きます。prefix argument をつけて実行した場合は、*F* と同様に付加するグローバルフラグを尋ねてきます。(wl-summary-mark-as-important)

F ミニバッファで入力した任意のグローバルフラグをつけます。Emacs 21 以降なら ',' で区切って複数のフラグを同時に設定出来ます。prefix argument をつけて実行した場合は、既についているグローバルフラグを取り除きます。(wl-summary-set-flags)

y
e 現在カーソルがある行のメッセージを保存します。(wl-summary-save)

n 一つ下のメッセージへ移動します。wl-summary-skip-mark-list にある一時マークが付いたメッセージへは移動しません。オフラインモードのときには、キャッシュされていないメッセージにも移動しません。(wl-summary-next)

p 一つ上のメッセージへ移動します。wl-summary-skip-mark-list にある一時マークが付いたメッセージへは移動しません。オフラインモードのときには、キャッシュされていないメッセージにも移動しません。(wl-summary-prev)

N 下方向にある未読もしくは '$'マークつきのメッセージへ移動します。オフラインモードのときには、キャッシュされていないメッセージに移動しません。ただし、まとめ処理マーク '*' がついたメッセージがある場合は、まとめ処理マーク '*' がついたメッセージに優先的に移動します。この挙動は wl-summary-move-orderの値に応じて変わります。(wl-summary-down)

P 上方向にある未読もしくは '$'マークつきのメッセージへ移動します。オフラインモードのときには、キャッシュされていないメッセージに移動しません。ただし、まとめ処理マーク '*' がついたメッセージがある場合は、まとめ処理マーク '*' がついたメッセージに優先的に移動します。この挙動は wl-summary-move-orderの値に応じて変わります。(wl-summary-up)

w 新規ドラフトを用意します。(wl-summary-write)

W 現在のサマリがニュースグループの場合、'Newsgroups:' フィールドを補ってドラフトを用意します。現在のサマリがメーリングリストの場合、メーリングリストのアドレスを推測して 'To:' フィールドを補ってドラフトを用意します。wl-subscribed-mailing-list を設定しておく必要があります。(wl-summary-write-current-folder)

H 現在カーソルがある行のメッセージを、全てヘッダ情報を表示するかどうかを切り換えて再表示します。prefix argument つきならばキャッシュが存在しても

無視して再読み込みし直します。*C-u C-u H* のように 2 回の *C-u* を付けて実行した場合、現在のヘッダの表示形式をサマリのデフォルト値として設定します。(wl-summary-toggle-all-header)

M 現在カーソルがある行のメッセージを、MIME 解析の指定を切り換えて再表示します。切り換えは、wl-summary-display-mime-mode-list に指定された順に行われます。また、数値の prefix argument つきで実行した場合、以下のように直接切り換えることが出来ます。

1: MIME 解析を有効にします
2: ヘッダのみ MIME 解析を有効にします
3: MIME 解析を無効にします

C-u C-u M のように 2 回の *C-u* を付けて実行した場合、現在の指定をサマリのデフォルト値として設定します。(wl-summary-toggle-mime)

C-c C-f 現在カーソルがある行のメッセージのヘッダの省略表示をトグルします。(wl-summary-toggle-header-narrowing)

B 現在カーソルがある行のメッセージが MIME でカプセル化された複数のメッセージ含む場合、それらを現在のフォルダにほどきます。書き込み不可のフォルダに居る場合や prefix argument をつけて実行した場合には、展開先のフォルダを尋ねます。(wl-summary-burst)

@ サマリ行のメールの 'From:' (発信者) を ~/.addresses にインタラクティブに追加します。既に登録されている場合は変更/削除もできます。prefix argument つきで実行すると、任意の入力アドレスを登録/変更/削除できます。(wl-summary-edit-petname)

Z ~/.addresses 等の状態を最新のものに更新します。(wl-status-update)

| 現在のメッセージの内容を他のプロセスにパイプ経由で引き渡します。(wl-summary-pipe-message)

現在のメッセージの内容を印刷します。Emacs 20 以降 では ps-print を使います。白黒プリンタでは、wl-ps-print-buffer-function を ps-print-buffer に設定したほうが良いかもしれません。

```
(setq wl-ps-print-buffer-function 'ps-print-buffer)
```

(wl-summary-print-message)

q 現在のフォルダを脱出します。(wl-summary-exit)

j 現在表示中のメッセージのバッファに移動します。(wl-summary-jump-to-current-message)

J 他のメッセージにジャンプします。(wl-summary-jump-to-msg)

I サマリの表示を更新した後、wl-summary-incorporate-marks に含まれるマークを持つメッセージをプリフェッチします。(wl-summary-incorporate)

M-j 入力した 'Message-ID:' を持つメッセージの行にジャンプします。elmo-use-database が non-nil なら、現在のサマリ以外からも候補を検索します。(wl-summary-jump-to-msg-by-message-id)

^ 現在のメッセージの親メッセージに移動します。(wl-summary-jump-to-parent-message)

! カーソルがある行のメッセージを読まなかったことにします。(wl-summary-mark-as-unread)

s メッセージの一覧表示の更新レンジの入力を受け付けた後、それに基づいてメッセージの一覧表示を更新します。更新レンジとしては以下のいずれかを指定できます。

all	現在の msgdb の内容を破棄し、全情報を取り寄せ直します。 killed メッセージは取り寄せ直しません。
all-entirely	現在の msgdb の内容を破棄し、全情報を取り寄せ直します。 killed メッセージも取り寄せ直します。
update	保持している msgdb の情報と最新情報の差分を更新します。 killed メッセージは更新しません。
update-entirely	保持している msgdb の情報と最新情報の差分を更新します。 killed メッセージも更新します。
rescan	現在の msgdb に基づいて再表示を行います。
rescan-noscore	現在の msgdb に基づいて再表示を行います。 スコアにより消えたメッセージも表示します。
rescan-thread	現在の msgdb に基づいて再表示を行います。 スレッドの情報も再構築します。
cache-status	キャッシュの状態をマークに反映します。
mark	マークの状態を最新にします。
no-sync	何もしません。
first:数字	フィルタフォルダに移動します。
last:数字	フィルタフォルダに移動します。

(wl-summary-sync)

S サマリの表示順序を並び替えます。'date' (日付)、'from' (発信者)、'number' (メッセージ番号)、'subject' (サブジェクト)、'size' (メッセージのサイズ)、'list-info' (メーリングリストの名前とメッセージ番号) のいずれかを元に並び替えます。prefix argument つきならば、選んだ順序の逆順でサマリを並び替えます。(wl-summary-sort)

T スレッド表示をトグルします。その状態は Wanderlust を終了しても保存されます。新規に作成されたサマリに対してのデフォルトの状態は wl-summary-default-view や wl-summary-default-view-alist で指定することができます。現在のスレッド表示状態はモードラインに表示されます。'{S}' はスレッド・オフ (Sequence) の状態、'{T}' はスレッド・オン (Thread) の状態を示します。(wl-summary-toggle-thread)

l フォルダモードのバッファの表示をトグルします。(wl-summary-toggle-disp-folder)

v メッセージのバッファの表示をトグルします。(wl-summary-toggle-disp-msg)

V 与えられた条件を持つメッセージのみを持つ仮想フォルダ (フィルタフォルダ) へ移動します。prefix argument つきで実行すると仮想フォルダから脱出します。(wl-summary-virtual)

TAB さっき表示したメッセージに飛びます。(wl-summary-goto-last-displayed-msg)

? 与えられた条件を持つメッセージにまとめ処理マーク '*' をつけます。通常は、既についている '*' マークはそのままで、単に追加されますが、prefix argument をつけて実行した場合には、以前のマークを削除して置き換えます。(wl-summary-pick)

R カーソル行のメッセージを読んだことにします。(wl-summary-mark-as-read)

x サマリバッファのすべてのメッセージに対して、一時的マークに対応するアクションを実行します。(wl-summary-exec)

*** カーソル行のメッセージにまとめ処理用マークをつけます。(wl-summary-target-mark-line) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

o カーソル行のメッセージにリファイルマークをつけます。(wl-summary-refile) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

C-o 自動リファイルを実行します。(wl-summary-auto-refile)

O カーソル行のメッセージにコピーマークをつけます。(wl-summary-copy) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

M-o カーソル行のメッセージに、直前にリファイルしたフォルダと同じフォルダ宛に、リファイルマークをつけます。(wl-summary-refile-prev-destination)

d カーソル行のメッセージに処分マークをつけます。処分の結果は wl-dispose-folder-alist で制御でき、デフォルトでは wl-trash-folder にリファイルされます。(wl-summary-dispose) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

D カーソル行のメッセージに強制削除マークをつけます。(wl-summary-delete) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

i カーソル行のメッセージにプリフェッチ予約マークを付けます。(wl-summary-prefetch) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

~ カーソル行のメッセージに再送予約マークを付けます。(wl-summary-resend) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

u カーソル行のメッセージに一時的マークがあれば取り除きます。(wl-summary-unmark)

U 指定した一時的マークをすべて取り除きます。(wl-summary-unmark-all)

r R 指定リージョンにあるメッセージを全て読んだことにします。(wl-summary-mark-as-read-region)

r $ 指定リージョンにあるメッセージ全てに 'important' フラグをつけます。すでに 'important' フラグがあればそのフラグを取り除きます。(wl-summary-mark-as-important-region)

r F 指定リージョンにあるメッセージ全てにミニバッファで入力した任意のグローバルフラグをつけます。(`wl-summary-set-flags-region`)

r ! 指定リージョンにあるメッセージを全て読まなかったことにします。(`wl-summary-mark-as-unread-region`)

r x 指定リージョンにあるメッセージに対して、一時的マークに対応するアクションを実行します。(`wl-summary-exec-region`)

*r ** 指定リージョンにあるメッセージにまとめ処理用マークをつけます。(`wl-summary-target-mark-region`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

r o 指定リージョンにあるメッセージにリファイルマークをつけます。(`wl-summary-refile-region`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

r O 指定リージョンにあるメッセージにコピーマークをつけます。(`wl-summary-copy-region`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

r d 指定リージョンにあるメッセージに処分マークをつけます。(`wl-summary-dispose-region`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

r D 指定リージョンにあるメッセージに強制削除マークをつけます。(`wl-summary-delete-region`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

r i 指定リージョンにあるメッセージにプリフェッチ予約マークを付けます。(`wl-summary-prefetch-region`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

r u 指定リージョンにあるメッセージにマークがあれば削除します。(`wl-summary-unmark-region`)

r y 指定リージョンにあるメッセージを保存します。(`wl-summary-save-region`)

t R カーソル行があるメッセージを先頭とするスレッドを読んだことにします。prefix argument つきならばカーソル行があるメッセージを含むスレッド全てを読んだことにします。(`wl-thread-mark-as-read`)

t $ カーソル行があるメッセージを先頭とするスレッドに '`important`' フラグをつけます。既に '`important`' フラグがあれば取り除きます。prefix argument つきならばカーソル行があるメッセージを含むスレッド全体に適用します。(`wl-thread-mark-as-important`)

t F カーソル行があるメッセージを先頭とするスレッドにミニバッファで入力した任意のグローバルフラグをつけます。prefix argument つきならばカーソル行があるメッセージを含むスレッド全体に適用します。(`wl-thread-set-flags`)

t ! カーソル行があるメッセージを先頭とするスレッドを読まなかったことにします。prefix argument つきならばカーソル行があるメッセージを含むスレッド全てを読まなかったことにします。(`wl-thread-mark-as-unread`)

t x カーソル行があるメッセージを先頭とするスレッドのメッセージに対して、一時的マークに対応するアクションを実行します。prefix argument つきならばカーソル行があるメッセージを含むスレッド全てに適用します。(`wl-thread-exec`)

*t ** カーソル行があるメッセージを先頭とするスレッドにまとめ処理用マークをつけます。prefix argument つきならばカーソル行があるメッセージを含むスレッド全体に適用します。(`wl-thread-target-mark`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

t o カーソル行があるメッセージを先頭とするスレッドにリファイルマークをつけます。prefix argument つきならばカーソル行があるメッセージを含むスレッド全体に適用します。(`wl-thread-refile`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

t O カーソル行があるメッセージを先頭とするスレッドにコピーマークをつけます。prefix argument つきならばカーソル行があるメッセージを含むスレッド全体に適用します。(`wl-thread-copy`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

t d カーソル行があるメッセージを先頭とするスレッドに処分マークをつけます。prefix argument つきならばカーソル行があるメッセージを含むスレッド全体に適用します。(`wl-thread-dispose`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

t D カーソル行があるメッセージを先頭とするスレッドに強制削除マークを付けます。(`wl-thread-delete`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

t i カーソル行があるメッセージを先頭とするスレッドにプリフェッチ予約マークを付けます。(`wl-thread-prefetch`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

t u カーソル行があるメッセージを先頭とするスレッドのメッセージにマークがあれば削除します。prefix argument つきならばカーソル行があるメッセージを含むスレッド全体に適用します。(`wl-thread-unmark`)

t y カーソル行があるメッセージを先頭とするスレッドを保存します。prefix argument つきならばカーソル行があるメッセージを含むスレッド全体を保存します。(`wl-thread-save`)

m R まとめ処理用マーク '`*`' のついたメッセージを読んだことにします。(`wl-summary-target-mark-mark-as-read`)

m $ まとめ処理用マーク '`*`' のついたメッセージに '`important`' フラグをつけます。すでに '`important`' フラグがあれば取り除きます。(`wl-summary-target-mark-mark-as-important`)

m F まとめ処理用マーク '`*`' のついたメッセージにミニバッファで入力した任意のグローバルフラグをつけます。(`wl-summary-target-mark-set-flags`)

m ! まとめ処理用マーク '`*`' のついたメッセージを読まなかったことにします。(`wl-summary-target-mark-mark-as-unread`)

m o まとめ処理用マーク '`*`' のついたメッセージにリファイルマークをつけます。(`wl-summary-target-mark-refile`) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

m O まとめ処理用マーク ‘*’ のついたメッセージにコピーマークをつけます。(wl-summary-target-mark-copy) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

m d まとめ処理用マーク ‘*’ のついたメッセージに処分マークをつけます。(wl-summary-target-mark-dispose) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

m D まとめ処理用マーク ‘*’ のついたメッセージに強制削除マークを付けます。(wl-summary-target-mark-delete) 5.7 節 「Mark and Action」(39 ページ) を参照してください.

m i まとめ処理用マーク ‘*’ のついたメッセージにプリフェッチ予約マークを付けます。(wl-summary-target-mark-prefetch) 5.7 節「Mark and Action」(39 ページ) を参照してください.

m y まとめ処理用マーク ‘*’ のついたメッセージを保存します。(wl-summary-target-mark-save)

m u 全ての一時的マークを消します。(wl-summary-delete-all-temp-marks)

m a 全てのメッセージにまとめ処理用マーク ‘*’ を付けます。(wl-summary-target-mark-all)

m t まとめ処理用マーク ‘*’ を現在のスレッドにつけます。(wl-summary-target-mark-thread)

m T まとめ処理用マーク ‘*’ を、現在マークされたメッセージを含むスレッド全体に拡大します。(wl-summary-target-mark-threads)

m r 指定されたリージョンにまとめ処理用マーク ‘*’ をつけます。(wl-summary-target-mark-region)

m A まとめ処理用マーク ‘*’ のついたメッセージを引用して返事を書くドラフトを用意します。(wl-summary-target-mark-reply-with-citation)

m f まとめ処理用マーク ‘*’ のついたメッセージをフォワードするドラフトを用意します。(wl-summary-target-mark-forward)

m U まとめ処理用マーク ‘*’ のついたメッセージをまとめて uudecode します。(wl-summary-target-mark-uudecode)

m ? まとめ処理用マーク ‘*’ のついたメッセージの内、条件にマッチするメッセージの ‘*’ マークのみを残します。(wl-summary-target-mark-pick)

m # まとめ処理用マーク ‘*’ のついたメッセージを印刷します。(wl-summary-target-mark-print)

m | まとめ処理用マーク ‘*’ のついたメッセージそれぞれを、指定した他のプロセスにパイプ経由で引き渡します。(wl-summary-target-mark-pipe)

M-t Wanderlust のオフラインモード/オンラインモードをトグルします。(wl-toggle-plugged)

C-t Wanderlust のサーバ・ポート別のオフライン/オンラインを変更します。(wl-plugged-change)

C-c C-o ドラフトバッファがあれば移動します。 複数のドラフトバッファが存在する場合は、次々と切り替えていきます。また、prefix argument をつけることにより、バッファが存在していない場合は、ドラフトフォルダからファイルを (存在すれば) 読み込みます。(`wl-jump-to-draft-buffer`)

M-w カーソル行のメッセージのコピー。(`wl-summary-save-current-message`)

C-y カーソル行のメッセージを親メッセージとして、`wl-summary-save-current-message` で保存されたメッセージをスレッドに繋ぎます。(`wl-summary-yank-saved-message`)

C-x C-s 現在のサマリをセーブします。(`wl-summary-save-status`)

5.9 カスタマイズ変数

`wl-summary-move-order`
: 初期設定は `unread`。メッセージを読み進めるときに、何を優先するかを指定します。新規メッセージを優先したいときは `new` を設定します。未読をメッセージを優先したいときは `unread` を設定します。`nil` なら単純に次のメッセージに進みます。

`wl-auto-select-first`
: 初期設定は `nil`。Non-nil ならフォルダに移動した時に最初のメッセージを自動的に表示します。

`wl-auto-select-next`
: 初期設定は `nil`。サマリに未読メッセージがなくなった時の動作を指定します。

  `nil`： フォルダモードに戻るか尋ねる
  `'unread`： 次の未読ありのフォルダに行くか尋ねる。
  移動先のフォルダで `crosspost` 処理や `Score` 機能などにより未読がなくなれば、さらに次のフォルダに行くか尋ねる。
  `'skip-no-unread`： `unread` と同様。
  ただし、移動した先で未読がなくなった場合には、自動的に次のフォルダに行く。
  上記以外： 次の未読ありのフォルダに行くか尋ねる。

  スペースキーでひたすら読み進めたい人は `'skip-no-unread` にしておくのが便利かもしれません。

`wl-thread-insert-opened`
: 初期設定は `nil`。Non-nil なら最初から thread が開かれた状態でサマリに表示されます。

`wl-thread-open-reading-thread`
: 初期設定は `t`。Non-nil なら、閉じた状態の thread へ移動した時に自動的に thread を開きます。

`wl-summary-exit-next-move`
: 初期設定は `t`。Non-nil なら、サマリを終了するときに次のフォルダに移動します。

`wl-folder-move-cur-folder`
: 初期設定は `nil`。Non-nil ならサマリで他のフォルダに移動するとフォルダモードでのカーソル位置も合わせて移動します。

`wl-summary-weekday-name-lang`
: サマリの曜日表示の言語を指定します。‘en’ なら英語、‘fr’ ならフランス語、‘de’ ならドイツ語となります。値を変更した後には、サマリを rescan して下さい。

`wl-summary-fix-timezone`
: 初期設定は `nil`。サマリの日時表示を指定したタイムゾーンに直します。`nil` ならば、デフォルトのタイムゾーンを参照して直します。デフォルトのタイムゾーンとは、システムにあらかじめ設定されたタイムゾーンか、環境変数 ‘TZ’ の値です。

`wl-use-petname`
: 初期設定は `t`。Non-nil ならサマリの From 部分にあだ名を表示します。

`wl-break-pages`
: 初期設定は `t`。Non-nil なら、‘^L’ で改ページしてメッセージを表示します。

`wl-summary-from-function`
: サマリの差出人表示の整形に使う関数を指定します。初期設定は `wl-summary-default-from` です。

`wl-summary-no-from-message`
: 初期設定は ‘nobody@nowhere?’。メッセージに ‘From:’ が無かった場合にサマリに表示する文字列です。

`wl-summary-subject-function`
: サマリのサブジェクト表示の整形に使う関数を指定します。初期設定は `wl-summary-default-subject` で、サブジェクト先頭のリスト名など部分をカットします。サブジェクトをそのまま表示するには以下のように設定します。

```
(setq wl-summary-subject-function 'identity)
```

`wl-summary-no-subject-message`
: 初期設定は ‘(WL:No Subject in original.)’。メッセージに ‘Subject:’ が無かった場合にサマリに表示する文字列です。

`wl-summary-default-view`
: 初期設定は `'thread`。新規に作成されたサマリの状態をスレッド表示なら `'thread`、番号順なら `'sequence` のいずれかで指定します。

`wl-summary-use-frame`
: 初期設定は `nil`。Non-nil ならサマリ表示用に新しいフレームを開きます。

`wl-use-folder-petname`
: 初期設定は以下のリスト。

```
(modeline)
```

フォルダのあだ名 (petname) を使用する場所 (シンボル) のリストを指定します。指定できるシンボルは次の通りです。

`modeline` サマリのモードラインを petname で表示します。

`ask-folder`
: `wl-auto-select-next` が non-nil のとき、移動先フォルダを petname で表示します。

read-folder
: wl-summary-read-folder におけるフォルダ入力の際、petname でも補完ができます。

wl-summary-move-direction-toggle
: 初期設定は t。Non-nil なら最後に実行された *p*, *P*, *n*, *N* の結果で、次のメッセージが上か下かを切り替える。読んでいる方向を意識したいときは t にすると良いでしょう。

wl-summary-width
: 初期設定は 80。サマリの表示幅を設定された値に切り詰めます。nil なら表示幅を切り詰めません。

wl-summary-print-argument-within-window
: 初期設定は nil。Non-nil なら wl-summary-width が nil であってもウィンドウの右端に揃えて、アクション引数を表示します。

wl-summary-indent-length-limit
: 初期設定は 46。設定された値以上サマリをインデントしません。nil ならサマリのインデントを無制限にします。値を nil にするときは、wl-summary-width も nil に設定するとよいでしょう。

wl-summary-max-thread-depth
: 初期設定は 30。設定された値以上の深さのスレッドを分割します。

wl-summary-recenter
: 初期設定は t。Non-nil ならば表示したときに表示中のメッセージのサマリ行をウィンドウの中央付近に移動します。

wl-summary-max-thread-depth
: 初期設定は 30。スレッドの深さがこの値より大きくなるとスレッドを分割します。

wl-summary-divide-thread-when-subject-changed
: 初期設定は nil。Non-nil ならサブジェクトが変わったときにスレッドを切ります。

wl-summary-search-via-nntp
: 初期設定は confirm。

  Non-nil なら wl-summary-jump-to-msg-by-message-id で、メッセージを見つけることができなかったときに、wl-summary-jump-to-msg-by-message-id-via-nntp を呼び、elmo-nntp-default-server で指定されたサーバから検索します。このとき elmo-nntp-default-user, elmo-nntp-default-port, elmo-nntp-default-stream-type で指定した条件でサーバと接続します。

  confirm なら elmo-nntp-default-* で指定されたサーバから検索するか、サーバを指定するかを確認します。この場合、サーバのホスト名、もしくは ‘-:username@servername:119!’ のように NNTP フォルダ形式での指定が可能です。

wl-summary-keep-cursor-command
: 初期設定は以下のリスト。

```
(wl-summary-goto-folder wl-summary-goto-last-visited-folder)
```

既に存在するサマリへ移動したときに更新せず、カーソル位置を保存するコマンドのリストです。

`elmo-folder-update-threshold`
: 初期設定は 500。この値よりサマリの更新数が多い場合、一部分だけ更新するかどうか、質問します（`elmo-folder-update-confirm` が non-nil の場合）。

`elmo-folder-update-confirm`
: 初期設定は `t`。Non-nil ならば `elmo-folder-update-threshold` による判定を行います。

`wl-summary-always-sticky-folder-list`
: 初期設定は `nil`。フォルダ名 (正規表現) のリストを設定することにより、自動的にスティッキーサマリとするかどうかをフォルダ毎に指定します。

`wl-summary-reserve-mark-list`
: 初期設定は以下のリスト。

```
("o" "O" "D" "d" "i")
```

このリストにある一時マークは、消さない限り上書きされません。

`wl-summary-skip-mark-list`
: 初期設定は以下のリスト。

```
("D" "d")
```

このリストにある一時マークの付いたメッセージは、カーソル移動の際にスキップされます。

`elmo-message-fetch-threshold`
: 初期設定は 30000 (bytes)。この値を越えるサイズのメッセージを表示する時に、確認を求めます（`elmo-message-fetch-confirm` が non-nil の場合）。

`elmo-message-fetch-confirm`
: 初期設定は `t`。Non-nil ならば `elmo-message-fetch-threshold` による判定を行います。

`wl-prefetch-threshold`
: 初期設定は 30000 (bytes)。`wl-prefetch-threshold` を越えるサイズのメッセージは、`wl-prefetch-confirm` が non-nil の場合、プリフェッチ時に確認を求めます。`wl-prefetch-threshold`を `nil` にすると、`wl-prefetch-confirm` の値にかかわらずプリフェッチを実行します。

`wl-prefetch-confirm`
: 初期設定は `t`。Non-nil ならば `wl-prefetch-threshold` を越えるサイズのメッセージをプリフェッチしようとしたときに、確認を求めます。

`elmo-imap4-use-cache`
: 初期設定は `t`。Non-nil なら、IMAP4 で読んだメッセージをキャッシュします。

`elmo-nntp-use-cache`
: 初期設定は `t`。Non-nil なら、NNTP で読んだメッセージをキャッシュします。

`elmo-pop3-use-cache`
: 初期設定は `t`。Non-nil なら、POP3 で読んだメッセージをキャッシュします。

elmo-shimbun-use-cache
: 初期設定は t。Non-nil なら、新聞フォルダで読んだメッセージをキャッシュします。

wl-summary-resend-use-cache
: 初期設定は nil。Non-nil はら、オフライン状態でもキャッシュを使って再送します。キャッシュを用いて再送すると、必ずしも意図したメッセージが使われるとは限らないことに留意してください。

wl-folder-process-duplicates-alist
: 初期設定は nil。重複したメッセージが同じフォルダにある場合の動作を指定します。各項目は、フォルダ名の正規表現と動作からなります。動作としては以下のものが指定できます。

```
nil : 重複メッセージに対し，何もしない．
hide : 重複メッセージをサマリに表示しない．
read : 重複メッセージを既読にする．
```

例えば以下のように設定します (マルチフォルダで重複メッセージを隠す場合)

```
(setq wl-folder-process-duplicates-alist
                  '(("^\\+draft$" . nil) ("^\\+trash$" . nil)
                    ("^\\*.*" . hide) (".*" . read)))
```

wl-summary-flag-alist
: 初期設定は以下の通り。

```
((important "orange"))
```

サマリでのフラグつきメッセージの色とマークを指定します。マークを省略した場合は、wl-summary-flag-mark に指定されたマークが使用されます。一つのメッセージに複数のグローバルフラグが設定されている場合には、このリストの前方にあるフラグが優先されます。

設定例:

```
(setq wl-summary-flag-alist
      '((important "purple")
        (todo "red")
        (business "green" "B")
        (private "blue" "X")))
```

wl-summary-display-mime-mode-list
: 初期設定は以下のリスト。

```
(mime as-is)
```

wl-summary-toggle-mime はこのリストの順に MIME 解析の指定を切り換えます。MIME 解析の方式には以下のものが指定できます。

```
mime        : MIME 解析を行います
header-only : ヘッダのみ MIME 解析を行います
as-is       : MIME 解析を行いません
```

6 メッセージバッファ

メッセージバッファは SEMI の MIME-View mode です。操作の方法、キーバインドはそれぞれのドキュメントを参照してください。*a MIME user interface for GNU Emacs* の “MIME-View” 節 を参照してください. また、メッセージバッファにおいて、`?` でヘルプが参照できます。

メッセージの先頭で *p*、メッセージの最後尾で *n* を押すと、サマリモードに戻ります。また、*l* を押すと、サマリモードバッファの表示をトグルします。

6.1 キーバインド

l サマリバッファを表示するかどうかをトグルします。(`wl-message-toggle-disp-summary`)

Button-2 マウスポインタの位置に ‘`Message-ID:`’ があったときに対応するメッセージが見つかれば表示します。(`wl-message-button-refer-article`)

マウスポインタの位置のヘッダが省略表示されていた場合には、全ての内容を表示します。もう一度押すと再度省略表示します。

Button-4 (ホイールつきマウスのホイールを上へ)

メッセージを上にスクロールします。メッセージの末尾まで到達すると、次のメッセージに移動します。(`wl-message-wheel-down`)

Button-5 (ホイールつきマウスのホイールを下へ)

メッセージを上にスクロールします。メッセージの先頭まで到達すると、前のメッセージに移動します。(`wl-message-wheel-up`)

D カーソル位置のパートを削除します。実際には、変更したメッセージを今居るフォルダに追加してから、古いメッセージをゴミ箱に移動するので、メッセージ番号は変化します。(`wl-message-delete-current-part`)

6.2 カスタマイズ変数

`wl-message-window-size`

初期設定は `(1 . 4)`。cons セルで、car 値: cdr 値がサマリのウィンドウ幅 : メッセージのウィンドウ幅となります。

`wl-message-ignored-field-list`

初期設定は `nil`。メッセージバッファに表示しないヘッダフィールドを正規表現のリストで指定します。`nil` なら、`mime-view-ignored-field-list` の値を使います。

`wl-message-visible-field-list`

初期設定は `nil`。メッセージバッファに必ず表示したいヘッダフィールドを正規表現のリストで指定します。`wl-message-ignored-field-list` よりも優先されます。`nil` なら、`mime-view-visible-field-list` の値を使います。

`wl-message-sort-field-list`

初期設定は’("Return-Path" "Received" "^To" "^Cc" "Newsgroups" "Subject" "^From")。メッセージバッファに表示するヘッダフィールドの順番を正規表現のリストで指定します。

`wl-message-truncate-lines`
: 初期設定では `default-truncate-lines` の値を使います。Non-nil ならメッセージバッファで長い行の折り返しをしません。

`wl-message-use-header-narrowing`
: 初期設定は `t`。Non-nil ならヘッダの省略表示を有効にします。

`wl-message-header-narrowing-fields`
: 初期設定は以下のリスト。

```
("to" "cc")
```

省略表示するフィールドを指定します。

`wl-message-header-narrowing-lines`
: 初期設定は 4。指定行数以上ある場合に省略表示します。

`wl-message-header-narrowing-string`
: 初期設定は '`...`'。ヘッダの内容を省略した時に表示する文字列を指定します。

`wl-message-auto-reassemble-message/partial`
: 初期設定は `nil`。Non-nil なら MIME メディアタイプが message/partial のメッセージを表示する際に、自動的に結合して表示します。

7 ドラフトバッファ

サマリモードやフォルダモードで w を押すと新規ドラフトバッファが用意されます。このバッファでは、メールやニュース記事の (新規) 編集、送信を行います。

W を押すと、(可能であれば) 宛先を推測してドラフトバッファを用意します。

7.1 使い方 (TIPS)

基本は Emacs 標準のメールモードです。

7.1.1 送信の為のパラメータ

メッセージの送信に使うサーバの情報に沿って以下の変数を設定して下さい。

`wl-smtp-posting-server`
: メール送信時の SMTP サーバ名です。

`wl-smtp-posting-port`
: メール送信時の SMTP ポート番号です。設定していない場合はデフォルトの SMTP ポート番号 (25) を使います。

`wl-nntp-posting-server`
: ニュース投稿時の NNTP サーバ名です。設定していない場合は `elmo-nntp-default-server` を使います。

`wl-nntp-posting-port`
: ニュース投稿時の NNTP サーバのポート番号です。設定していない場合は `elmo-nntp-default-port` を使います。

必要に応じて以下の変数も設定して下さい。詳細はカスタマイズ変数の節を参照して下さい。7.3 節 「Variables of Draft Mode」(61 ページ) を参照してください.

`wl-smtp-posting-user`
: SMTP AUTH による認証を行なうときのユーザ名です。

`wl-smtp-authenticate-type`
: SMTP AUTH による認証を行なうときの認証方式です。設定していない場合は認証を行いません。

`wl-smtp-authenticate-realm`
: SMTP AUTH による認証を行なうときのレルム (realm) を指定します。設定していない場合はレルムの指定を行いません。

`wl-smtp-connection-type`
: SMTP のコネクションをどのように張るかを指定します。

`wl-nntp-posting-user`
: ニュース投稿時に AUTHINFO による認証を行なうときのユーザ名です。

`wl-nntp-posting-stream-type`
: NNTP のコネクションをどのように張るかを指定します。

7.1.2 ヘッダの編集

実際に送信操作を行なうまででであれば、‘--text follows this line--’ より上のヘッダ領域は自由に編集することができます。

初期状態では ‘To:’ にカーソルがあります。送信相手のアドレスを入力してください。アドレスを入力する際には、TAB で補完ができます。

メッセージの宛先を指定するために、以下のフィールドが利用できます。それぞれ自分で書き加えて下さい。フィールド名を入力する際には、TABで補完ができます。

‘Newsgroups:’
: ニュース記事として出す場合の投稿先を指定します。

‘Cc:’
: メッセージのコピー (Carbon Copy) を送付するアドレス。

以下のヘッダは、送信されるメッセージからは削除されます。

‘Bcc:’
: メッセージのコピー (Blind Carbon Copy) を送付するアドレス。

‘Fcc:’
: 送信したメッセージを保存するフォルダを指定します。

‘Ecc:’
: 送信したメッセージをカプセル化して転送します。

以下の変数により、最初に用意するヘッダを追加できます。

wl-fcc
: 初期設定は nil。Non-nil なら、設定された値をドラフトの ‘Fcc:’ として最初から挿入します。関数が指定されている場合、関数の返り値 (string) が挿入されます。

wl-bcc
: 初期設定は nil。Non-nil なら、設定された値をドラフトの ‘Bcc:’ として最初から挿入します。

7.1.3 メッセージの編集と送信

説明するまでもないですが、メッセージ本文の編集は基本的に通常の文章作成と同様に行ないます。本文は ‘--text follows this line--’ の行より下に記述します。(注意：‘--text follows this line--’ の行はいじらないで下さい。)

マルチパートの編集は SEMI の MIME 編集モードを利用しています。編集の方法については各パッケージ付属のドキュメントを参照してください。*a MIME user interface for GNU Emacs* の “MIME-Edit” 節 を参照してください. また、ドラフトバッファにおいて、*C-c C-x ?* でヘルプが参照できます。

編集中のドラフトをセーブすると、wl-draft-folder で設定したドラフトフォルダに追加されます。*C-c C-z* (wl-draft-save-and-exit) により、後から編集できるようにそれをセーブした上でドラフトバッファを離脱できます。ドラフトフォルダに入って *E* (wl-summary-reedit) を押せば、編集を再開できます (5.8 節 「Key Bindings of Summary」(40 ページ) を参照)。

メッセージの編集が完了したら、*C-c C-c* で送信できます。

7.1.4 メッセージの動的な変更

wl-draft-config-alist を設定すると、ヘッダやその他の情報に基づいて、自動的に他のヘッダや本文を変更することができます。

変数 wl-draft-config-alist について説明します。この変数の初期設定は nil です。

例えば以下のように設定すると、wl-draft-send-and-exit や wl-draft-send を実行した時にヘッダが変更されます。wl-interactive-send を non-nil に設定しておくと、送信前に変更点が確認できるので安心です。

```
(setq wl-draft-config-alist
      '(((string-match "aaa\\.example\\.com$" (system-name))
         ;; 式が non-nil なら適用する
         (wl-smtp-posting-server . "mailserver-B")
         (wl-nntp-posting-server . "newsserver-B")
         ;; 一時的な変数の設定
         )
        ("^To: .*user@aaa\\.bbb\\.example\\.com"
         ;; ドラフトバッファのヘッダにマッチすれば適用する
         ("Organization" . (format "Go %s" my-webpage)))
                          ;; elisp 式が書ける (eval しているだけ)
         (top . "  ○○です。\n")     ;; 本文先頭へ文字列を挿入します
         (bottom . "\n以上です。\n") ;; 本文末尾へ文字列を挿入します
        ))
```

wl-draft-config-alist は次の形式になっています。

```
'((ヘッダの正規表現 または elisp 式
   ("Field" . 値 (elisp 式))
   (variable . 値 (elisp 式))
   (サブ関数 . 値 (elisp 式))
   関数
   ...)
  (ヘッダの正規表現 または elisp 式
   ("Field" . 値 (elisp 式))
   ...))
```

[サブ関数] には、デフォルトで以下の 13 個を用意しています。

```
'header:      ヘッダの末尾に指定した文字列を挿入します。
'header-top:  ヘッダの先頭に指定した文字列を挿入します。
'header-file: ヘッダの末尾に指定したファイルを挿入します。
'x-face:      指定したファイルの内容を持つ 'X-Face:' フィールドを
              挿入します。
'top:         本文の先頭に指定した文字列を挿入します。
'top-file:    本文の先頭に指定したファイルを挿入します。
'body:        本文を指定した文字列に置き換えます。
              nil を指定すると本文の文字列を削除します。
'body-file:   本文を指定したファイルの内容で置き換えます。
'bottom:      本文の末尾に指定した文字列を挿入します。
'bottom-file: 本文の末尾に指定したファイルを挿入します。
'part-top:    現在のマルチパートの先頭に指定した文字列を挿入します。
'part-bottom: 現在のマルチパートの末尾に指定した文字列を挿入します。
'template:    指定したテンプレートを適用します。
              (次項 テンプレートの挿入参照)
```

また、これらを定義しているのは wl-draft-config-sub-func-alist であり、定義を変更したり自分で作った関数を追加することができます。関数の書き方はここで説明するより直接コードを見た方が分かりやすいでしょうから、省略します。

各要素の 1 番目には **ヘッダの正規表現** か elisp 式 を指定します。elisp 式の場合は評価した値が non-nil の場合に適用されます。

また、デフォルトでは複数の要素がマッチまたは non-nil になった場合に、その全てが適用されます。ここで、変数 wl-draft-config-matchone を t にしてあれば、最初にマッチした 1 つだけを適用することもできます。

要素の 2 番目には cons か関数の list を指定します。cons にはヘッダの Field か変数、サブ関数を指定します。Field を指定した場合はその Field を変更し、変数を指定した場合は一時的にその変数値を変更します。

値には文字列や変数の他、elisp 式をそのまま記述することもできます。もし、Field の値が nil ならばその Field を削除します。

つづいて、次の例を見て下さい。

```
(setq wl-draft-config-alist
      '((reply                          ;; (1)
         "X-ML-Name: \\(Wanderlust\\|emacs-mime-ja\\|apel-ja\\)"
         ;; 返信元バッファのヘッダにマッチすれば適用する
         (body . "  こんにちは\n")
         (template . "default")
         )))
```

この例 (1) のように、ヘッダの正規表現の前に reply をつけると wl-summary-reply などでドラフトを作成した場合、返信元のヘッダにマッチすれば適用されるようになります。ただし、wl-draft を実行したときなど返信元のバッファがない場合は無視されます。

親フォルダの名前を利用したい場合には、バッファローカル変数 wl-draft-parent-folder を利用できます。次の例では、ドラフトを開く時に居たサマリバッファのフォルダ名によって From を変更します。

```
(setq wl-draft-config-alist
      '(((string-match \".*@domain1$\" wl-draft-parent-folder)
         (\"From\" . \"user@domain1\"))
        ((string-match \".*@domain2$\" wl-draft-parent-folder)
         (\"From\" . \"user@domain2\"))))
```

何も特別な設定をしなければ、wl-draft-config-alist は、wl-draft-send-and-exit か wl-draft-send が実行される直前に一度だけ適用されます。送信を取りやめた後に再度 wl-draft-config-alist を適用したい場合は、*C-c C-e* (wl-draft-config-exec) を実行してください。

wl-draft-send-and-exit、wl-draft-send の実行時に自動的に wl-draft-config-alist を適用したくない場合は、次のように設定して下さい。

```
(remove-hook 'wl-draft-send-hook 'wl-draft-config-exec)
```

ドラフトバッファの作成時に自動的に wl-draft-config-alist を適用したい場合は、次のように設定します。

```
(add-hook 'wl-mail-setup-hook 'wl-draft-config-exec)
```

サマリモードから 'E'(wl-summary-reedit) を押してメールを編集する時にも自動的に wl-draft-config-alist を適用したい場合は、次のように設定します。

```
(add-hook 'wl-draft-reedit-hook 'wl-draft-config-exec)
```

7.1.5 テンプレートの挿入

変数 wl-template-alist を設定し、ドラフトバッファで *C-c C-j* もしくは *M-x wl-template-select*と入力することで適用します。wl-template-alist の書式は wl-draft-config-alist とほぼ同じです。7.1.4 節「Dynamical Message Re-arrangement」(56 ページ) を参照してください.

```
(setq wl-template-alist
      '(("default"
         ("From" . wl-from)
         ("Organization" . "組織名")
         (body . "  ○○です。\n"))
        ("report"
         (template . "default")                    ;; (a)
         ("To" . "jousi@example.com")
         ("Subject" . "報告")
         (body-file . "~/work/report.txt")
         )
        ))
```

このようにヘッダの正規表現の代わりに 'default' や 'report' のように各要素の名前 (テンプレート名) をつけるだけです。各要素の定義は wl-draft-config-alist と同じなので、(a) のように別のテンプレートを呼ぶことも可能です。

コマンド wl-template-select を実行すると、テンプレートを選択できます。変数 wl-template-visible-select の値によって、それぞれ以下のように振舞います。

wl-template-visible-select が t (デフォルト) の場合、ドラフトバッファの下に適用後のバッファウィンドウが表示されるので、それを見ながら *n* と *p* で選択します。そしてリターンキーで決定すると実際にドラフトバッファに適用されます。*q* を押すと何も適用されません。なお、wl-template-buffer-lines でウィンドウの大きさを変更できます。

wl-template-visible-select が nil の場合、ミニバッファでテンプレート名を入力することで選択します。

wl-template-select を prefix argument つきで実行した場合には、wl-template-visible-select の値を逆にした動作になります。

また、wl-draft-config-alist の例のように例えば

```
(template . "default")
```

と書くことで 'default' のテンプレートを適用できます。

7.1.6 POP-before-SMTP によるメールの送信

POP-before-SMTP によるメールの送信ができます。そのために必要な設定は、メール送信に用いる関数をデフォルトの wl-draft-send-mail-with-smtp から変更する

```
(setq wl-draft-send-mail-function 'wl-draft-send-mail-with-pop-before-smtp)
```

と合わせて、必要に応じて以下の変数を設定してください。

`wl-pop-before-smtp-user`
: POP-before-SMTP で認証を行なうときの POP ユーザ名です。設定していない場合は `elmo-pop3-default-user` を使います。

`wl-pop-before-smtp-server`
: POP-before-SMTP で認証を行なうときの POP サーバ名です。設定していない場合は `elmo-pop3-default-server` を使います。

`wl-pop-before-smtp-authenticate-type`
: POP-before-SMTP で認証を行なうときの POP 認証方式です。設定していない場合は `elmo-pop3-default-authenticate-type` を使います。

`wl-pop-before-smtp-port`
: POP-before-SMTP で認証を行なうときの POP ポート番号です。設定していない場合は `elmo-pop3-default-port` を使います。

`wl-pop-before-smtp-stream-type`
: `ssl` なら SSL を利用して POP コネクションを張ります。`starttls` なら STARTTLS (RFC2595) を利用してコネクションを張ります。設定していない場合は `elmo-pop3-default-stream-type` を使います。

POP-before-SMTP 用の変数 (`wl-pop-before-smtp-*`) を設定していない場合には、POP フォルダの設定 (`elmo-pop3-default-*`) が用いられます。そのため、SMTP サーバと POP サーバの実体が同じで、デフォルト設定の POP フォルダ (‘`&`’ など) が利用可能なら、設定は不要です。

なお、POP-before-SMTP については以下を参照してください。

`http://www.iecc.com/pop-before-smtp.html`

7.2 キーバインド

C-c C-y 現在表示中のメッセージバッファ (カーソル位置のパート) の内容を引用します。リージョンが有効な場合には、リージョンを引用します (`transient-mark-mode` (GNU Emacs の場合) か `zmacs-regions` (XEmacs の場合) が Non-nil の時のみ)。また、prefix argument を付けると、yank で挿入されるテキストの内容 (クリップボードの内容) が引用されます。(`wl-draft-yank-original`)

C-c C-p 現在のドラフトの内容をプレビューします。MIME のマルチパートメッセージの確認に使うと便利です。(`wl-draft-preview-message`)

C-c C-s 現在のドラフトの内容を送信します。ドラフトバッファは消去しません。複数の人に少しずつ内容を変えてメッセージを送りたい場合に便利です。(`wl-draft-send`)

C-c C-c 現在のドラフトの内容を送信し、ドラフトバッファを消去します。(`wl-draft-send-and-exit`)

C-x C-s 現在のドラフトをセーブします。(`wl-draft-save`)

C-c C-k 現在のドラフトを破棄します。セーブしていた場合、‘`+draft`’ フォルダからも削除されます。(`wl-draft-kill`)

C-x k 現在のドラフトを破棄します。(`wl-draft-mimic-kill-buffer`)

C-c C-z 現在のドラフトをセーブし、ドラフトバッファを消去します。ドラフトの編集を中断したいときに便利です。(`wl-draft-save-and-exit`)

C-c C-r 指定されたリージョンをシーザ暗号でエンコード/デコードします。(`wl-caesar-region`)

C-l 現在のドラフトをハイライトしなおします。(`wl-draft-highlight-and-recenter`)

M-t Wanderlust のオフラインモード/オンラインモードをトグルします。(`wl-toggle-plugged`)

C-c C-o 他のドラフトバッファがあれば移動します。また、prefix argument をつけることにより、バッファが存在していない場合は、ドラフトフォルダからファイルを (存在すれば) 読み込みます。(`wl-jump-to-draft-buffer`)

C-c C-e `wl-draft-config-alist`を適用します。(`wl-draft-config-exec`)

C-c C-j テンプレートを選択します。(`wl-template-select`)

C-c C-a アドレスマネージャを起動します。11.2 節 「Address Manager」(86 ページ) を参照してください. (`wl-addrmgr`)

C-c C-d ポイントとマークの間の文を省きます (`wl-draft-elide-region`)。文章は切られて (killed) 変数 `wl-draft-elide-ellipsis` の値で置き換えられます。デフォルトの省略符号として使われる値は (‘`[...]`’) です。

7.3 カスタマイズ変数

`wl-subscribed-mailing-list`
初期設定は `nil`。参加しているメーリングリストのリスト。返事のドラフトを用意するときに ‘`Mail-Followup-To:`’ や ‘`Cc:`’ にこれらが含まれるときは自分のアドレスを除きます。また、これらが ‘`To:`’ か ‘`Cc:`’ に含まれるときには、そのアドレスでリファイル先を覚えます。

設定例:

```
(setq wl-subscribed-mailing-list
      '("wl@ml.gentei.org"
        "apel-ja@m17n.org"
        "emacs-mime-ja@m17n.org"))
```

`wl-insert-mail-followup-to`
初期設定は `nil`。Non-nil なら、‘`Mail-Followup-To:`’ フィールドをドラフトバッファに自動的に挿入します。

`wl-insert-mail-reply-to`
初期設定は `nil`。Non-nil なら、‘`Mail-Reply-To:`’ フィールドをドラフトバッファに自動的に挿入します。

`wl-auto-insert-x-face`
初期設定は `t`。Non-nil でかつ、エンコードされた X-Face 文字列を`~/.xface` (変数 `wl-x-face-file` の値) の内容に用意しておくと、ドラフトを準備するときに自動的に ‘`X-Face:`’ フィールドとして挿入されます。`nil` の場合は自動的には挿入しません。

`wl-insert-message-id`
: 初期設定は `t`。Non-nil なら、送信時に ‘`Message-ID:`’ フィールドを自動的に挿入します。

`wl-message-id-use-message-from`
: 初期設定は `t`。Non-nil なら、‘`Message-ID:`’ のドメインパートに ‘`From:`’ フィールド、もしくは `wl-from` の値を利用します。

`wl-local-domain`
: 初期設定は `nil`。`nil` ならば ‘`Message-ID:`’ のドメインパートには関数 `system-name` の返り値を設定します。

  `system-name` が FQDN (‘`smtp.gohome.org`’ のようなホストのフルネーム) を返さない場合は、この変数に **必ず** ホスト名を除いたドメイン名 (‘`gohome.org`’ など) を設定してください。もし、グローバルな IP アドレスを持たない場合は、`wl-message-id-domain`を設定してください。(‘`Message-ID:`’ のドメインがおかしいと、ネットニュースでふくろ叩きにあってしまう可能性があります。)

  また、この値を設定している場合、`system-name` にこの値を付加したホスト名を SMTP の HELO の引数として利用します。

`wl-message-id-domain`
: 初期設定は `nil`。Non-nil なら、‘`Message-ID:`’ のドメインパートに使用されます。グローバルな IP アドレスを持たない場合など、ドメインパートを決められない場合は、`wl-message-id-domain` に一意な文字列 (例えば、メールアドレスなど) を設定してください。

`wl-unique-id-suffix`
: 初期設定は ‘`.wl`’。Message-ID を生成する際に ‘`@`’ または ‘`%`’ の直前に現れる文字列を指定します。

`wl-draft-config-alist`
: 初期設定は `nil`。送信直前にドラフトメッセージを変更します。自動的に `wl-draft-config-alist` の内容が適用されるのは送信時に一度だけです。もし，手動で適用する場合は，*C-c C-e* (`wl-draft-config-exec`) を使用してください。このコマンドは何度でも適用できます。

`wl-template-alist`
: 初期設定は `nil`。ドラフトバッファで適用するテンプレートを設定します。

`wl-draft-config-matchone`
: 初期設定は `nil`。Non-nil なら `wl-draft-config-alist` の適用時に最初にマッチした要素のみを適用します。`nil` ならマッチしたものすべてを適用します。

`wl-template-visible-select`
: 初期設定は `t`。Non-nil なら別ウィンドウに適用後の状態を表示しながらテンプレートを選択します。

`wl-template-confirm`
: 初期設定は `nil`。Non-nil ならウィンドウを表示しながらテンプレートを選択する場合、リターンキーで選択する時に確認を行います。

`wl-template-buffer-lines`
: 初期設定は 7。`wl-template-visible-select`が non-nil の場合、適用後の状態を表示するウィンドウの大きさを指定します。

`wl-draft-buffer-style`
: 初期設定は `full`。(返信と転送の場合以外の) ドラフトバッファのウインドウの形態を指定します。`keep` とすると現在のウィンドウを使い、`full` とするとフレーム全体のウインドウを使います。`split` とすると現在のウィンドウを分割して使います。関数が指定された場合、ドラフトバッファを引数としてそれ呼び出します。

`wl-draft-reply-buffer-style`
: 初期設定は `split`。返信と転送の場合のドラフトバッファのウインドウの形態を指定します。`keep` とするとメッセージバッファのウィンドウを使い、`full` とするとフレーム全体のウインドウを使います。`split` とするとメッセージバッファのウィンドウを分割して使います。関数が指定された場合、ドラフトバッファを引数としてそれ呼び出します。

`wl-draft-use-frame`
: 初期設定は `nil`。Non-nil ならドラフト用に新しいフレームを開きます。

`wl-draft-reply-default-position`
: 初期設定は `body` 返信の場合のドラフトでのカーソルの初期位置を指定します。`body` とするとメッセージ本文の先頭、`bottom` とすると本文末尾、`top` とするとヘッダの先頭にカーソルを移動します。

`wl-draft-truncate-lines`
: 初期設定では `default-truncate-lines` の値を使います。Non-nil ならドラフトバッファで長い行の折り返しをしません。

`wl-from`
: 初期設定は変数 `user-mail-address` の値。設定された値をドラフトの '`From:`' フィールドとして最初から挿入します。

`wl-envelope-from`
: 初期設定は `nil`。設定した値を envelope from (MAIL FROM) に使用します。`nil` なら `wl-from` のアドレス部分を使用します。

`wl-user-mail-address-list`
: 初期設定は `nil`。ユーザのアドレスリストです。アドレスを複数持っている場合は設定してください。

`wl-reply-subject-prefix`
: 初期設定は '`Re: `'。返信時のドラフトの '`Subject:`' で、元記事の '`Subject:`' の先頭に付け加える文字列です。返信対象のメッセージバッファで評価される関数を指定することもできます。

`wl-forward-subject-prefix`
: 初期設定は '`Forward: `'。転送時のドラフトの '`Subject:`' で、元記事の '`Subject:`' の先頭に付け加える文字列です。転送対象のメッセージバッファで評価される関数を指定することもできます。

`wl-draft-reply-use-address-with-full-name`
: 初期設定は `t`。Non-nil なら返信アドレスの '`To:`', '`Cc:`' フィールドに相手のフルネームを挿入します。`nil` ならアドレスだけを挿入します。

`wl-draft-enable-queuing`
: 初期設定は `t`。オフライン送信するかどうかを示すフラグです。Non-nil ならオフライン送信します。

`wl-draft-use-cache`
: 初期設定は `nil`。送信するメッセージをキャッシュするかどうかを示すフラグです。Non-nil ならキャッシュします。ただし `wl-insert-message-id`が nil の場合はキャッシュしません。

`wl-fcc-force-as-read`
: 初期設定は `nil`。Non-nil なら '`Fcc:`' で保存したメッセージを既読にします。

`wl-auto-flush-queue`
: 初期設定は `t`。オンラインになったときに自動的にキューを送信するかどうかを示すフラグです。Non-nil なら自動的に送信します (一応 `y-or-n-p` で確認します)。手動でキューを送信するには、フォルダモードで *F* を押してください。

`wl-ignored-forwarded-headers`
: 初期設定は '`\\(received\\|return-path\\|x-uidl\\)`'。転送時に削除するヘッダフィールド名を正規表現で指定します。

`wl-ignored-resent-headers`
: 初期設定は '`\\(return-receipt\\|[bdf]cc\\)`'。再送時に削除するヘッダフィールド名を正規表現で指定します。

`wl-draft-always-delete-myself`
: Non-nil なら、自分宛てのメールに返信する場合、常に '`To:`', '`Cc:`' から自分のメールアドレスを削除します。

`wl-draft-delete-myself-from-bcc-fcc`
: Non-nil で、'`To:`', '`Cc:`' が 変数 `wl-subscribed-mailing-list` に含まれている場合、'`Bcc:`' , '`Fcc:`' をつけません.

`wl-draft-send-mail-function`
: 初期設定は `wl-draft-send-mail-with-smtp`。メール送信に使う関数です。POP-before-SMTP を利用する場合は `wl-draft-send-mail-with-pop-before-smtp` に設定します。

`wl-smtp-posting-server`
: 初期設定は `nil`。メール送信時の SMTP サーバ名です。

`wl-smtp-posting-port`
: 初期設定は `nil`。メール送信時の SMTP ポート番号です。`nil` ならデフォルトの SMTP ポート番号 (25) を使います。

`wl-smtp-posting-user`
: 初期設定は `nil`。SMTP AUTH による認証を行なうときのユーザ名です。

`wl-smtp-authenticate-type`
: 初期設定は `nil`。SMTP AUTH による認証を行なうときの認証方式を文字列で指定します。値として `plain`, `cram-md5`, `digest-md5`, `login` などが指定できます。`nil` なら認証を行いません。

`wl-smtp-authenticate-realm`
: 初期設定は `nil`。SMTP AUTH による認証を行なうときのレルム (realm) を文字列で指定します。レルムの指定は DIGEST-MD5 等の認証方式で必要な場合があります。`nil` の場合はレルムの指定を行いません。

wl-smtp-connection-type
: 初期設定は nil。SMTP のコネクションをどのように張るかをシンボルで指定します。nil ならデフォルトの接続型式を利用します。starttls なら STARTTLS (RFC3207) を利用してコネクションを張ります。ssl なら SSL を利用します。

wl-nntp-posting-server
: 初期設定は nil。ニュース投稿時の NNTP サーバ名です。nil なら elmo-nntp-default-server を使います。

wl-nntp-posting-user
: 初期設定は nil。ニュース投稿時に AUTHINFO による認証を行なうときのユーザ名です。nil なら elmo-nntp-default-user を使います。それでも nil なら AUTHINFO による認証を行ないません。

wl-nntp-posting-port
: 初期設定は nil。ニュース投稿時の NNTP サーバのポート番号。nil なら elmo-nntp-default-port を使います。

wl-nntp-posting-stream-type
: 初期設定は nil。nil なら elmo-nntp-default-stream-type を評価します。ssl ならニュース投稿時に SSL を利用します。starttlsなら STARTTLS (RFC2595) を利用してコネクションを張ります。

wl-nntp-posting-function
: 初期設定は elmo-nntp-post。ニュース投稿のための関数。

wl-nntp-posting-config-alist
: 初期設定は nil。以下の例のようにして、ニュース投稿時のサーバ選択方法を設定します。wl-nntp-posting-{server|user|port|function}より優先されます。

```
(setq wl-nntp-posting-config-alist
      '((",?gmane\\." . "news.gmane.org")
        (",?comp\\." .
         ((server . "news-server")
          (user . "newsmaster")
          (port . 119)
          (function . elmo-nntp-post)))
        (".*" . "default-news-server")))
```

wl-pop-before-smtp-user
: 初期設定は nil。POP-before-SMTP で POP を行なうときのユーザ名です。nil のままなら elmo-pop3-default-userを利用します。

wl-pop-before-smtp-server
: 初期設定は nil。POP-before-SMTP で POP を行なうときのサーバ名です。nil のままなら elmo-pop3-default-serverを利用します。

wl-pop-before-smtp-authenticate-type
: 初期設定は nil。POP-before-SMTP で POP を行なうときの認証方式です。nil のままなら elmo-pop3-default-authenticate-typeを利用します。

`wl-pop-before-smtp-port`
: 初期設定は `nil`。POP-before-SMTP で POP を行なうときのポート番号です。`nil` のままなら `elmo-pop3-default-port`を利用します。

`wl-pop-before-smtp-stream-type`
: 初期設定は `nil`。POP-before-SMTP で SSL を利用するかどうかを示すフラグです。`nil` のままなら `elmo-pop3-default-stream-type`を利用します。`ssl` なら SSL を利用します。`starttls`なら STARTTLS (RFC2595) を利用してコネクションを張ります。

`wl-draft-queue-save-variables`
: オフライン送信時にキューに格納したメッセージについて保存しておく変数をリストで指定します。

`wl-draft-sendlog`
: 初期設定は `t`。`t` なら `~/.elmo/sendlog` に送信ログを出力します。ログを出力するタイミングは以下の通りです (失敗の場合も)。

  - smtp, qmail による送信
  - fcc によるフォルダへの格納
  - queuing によるフォルダへの格納

  ただし、`im-wl.el` による送信では、`sendlog` には出力せずに `imput` のログ機能におまかせします。

`wl-draft-sendlog-max-size`
: 初期設定は 20000 (バイト)。`wl-draft-sendlog` が `t` の場合、保存したログの大きさが指定した大きさ以上になれば、ログをローテーションします。

`wl-use-ldap`
: 初期設定は `nil`。Non-nil なら LDAP を利用してアドレス補完します。

`wl-ldap-server`
: 初期設定は ‘`localhost`’。アドレス補完に用いる LDAP サーバ名です。

`wl-ldap-port`
: 初期設定は `nil`. アドレス補完に用いる LDAP サーバのポート番号です。

`wl-ldap-base`
: 初期設定は ‘`c=US`’。アドレス補完時の LDAP 検索の開始点 (base) を指定します。

`wl-draft-remove-group-list-contents`
: 初期設定は `t`。Non-nil ならメール送信の際に宛先から group-list の内容を削除します (group-list とは宛先に含まれる ‘`Group: foo@gohome.org, bar@gohome.org;`’のような記述を指します)。

8 オフライン処理

Wanderlust にはオンラインモードとオフラインモードがあります。

8.1 オフラインモード

Wanderlust にはオンラインモードとオフラインモードがあります。オフラインモードでは、ネットワーク経由でなければ読めないメッセージにはアクセスできません (キャッシュされていればアクセスできます)。

モードラインの '[ON]' という表示は、オンラインモードにあることを示しています。モードラインが '[--]' という表示になっているときはオフラインモードです。フォルダモード、サマリモードで *M-t* を押すとオフライン/ オンラインの切り替えができます。

オフラインモードではサマリモードの *n* と *p* の動作が変わり、キャッシュされていないメッセージへは移動しなくなります。

~/.wl などで変数 wl-plugged を nil に設定してから起動すると、起動時からオフラインモードとなります。

8.2 オフラインモードで実行できる操作

以下の操作は (対象となるメッセージがキャッシュされていれば) オフラインモードでも実行できます。(変数 elmo-enable-disconnected-operation (後述) が non-nil のとき。) 8.3 節 「Plugged Mode」(68 ページ) を参照してください, 8.4 節 「Off-line State settings」(69 ページ) を参照してください.

オフラインモードで行ったこれらの操作が、ネットワーク経由でサーバ上に反映されるのは、Wanderlust がオンラインモードになった瞬間です。

変数 elmo-enable-disconnected-operation が nil なら、これらのネットワークフォルダに関するオフライン処理を実行せず、オフライン中のリファイル/コピー等の操作は単にエラーになります。

8.2.1 メッセージの送信

オフライン状態でメール/ニュース記事の送信操作をすると、送信の予約がされます。(im-wl.el をお使いの場合は、関係ありません。) オフラインのときに送信予約されたメッセージはキューフォルダ '+queue' に溜ります。溜ったメッセージは、オンラインになったときに一気に送信されます。

オフラインのうちに '+queue' を訪れて、キューにあるメッセージの内容を確認できます。メッセージを削除することも可能です。(削除されたメッセージはオンラインになっても送信されません。)

8.2.2 リファイル/コピー (IMAP4)

オフライン状態のあいだに実行された IMAP フォルダに対するリファイル/コピー操作はキューに溜められ、オンラインになったときにサーバ側に反映されます。オフライン・リファイル/コピーの後、リファイル/コピー先のフォルダを訪れると、オフラインでもメッセージが追加されているように見えます。

オフライン・リファイルのキュー処理時の削除処理は安全を期してサーバ上のメッセージと 'Message-ID:' が一致した場合のみ実行されます。また、キュー処理時にリファイル/コピー先に指定したフォルダへメッセージを追加できなかった場合には、それらのメッセージを '+lost+found' フォルダに追加します。

8.2.3 フォルダ生成 (IMAP4)

IMAP フォルダの生成もオフライン状態で実行できます。オンラインになったときにフォルダ生成がサーバに反映されますが、このとき、何らかの原因でフォルダ生成が失敗してしまった場合、オフライン中に生成されたフォルダへリファイルされたメッセージは '+lost+found' フォルダに追加されます。

8.2.4 マーク付け (IMAP4)

IMAP フォルダにあるメッセージに対する未読/既読の情報、および、重要マーク '$' がついているかどうかも、オフライン中の変更がオンラインになったときにサーバに反映されます。

8.2.5 プリフェッチ

ネットワークフォルダ (IMAP, NNTP, POP3, shimbun) にあるメッセージに対して、プリフェッチの予約をします。プリフェッチを予約したメッセージには 'u' が付きますが、この時点ではキャッシュされておらず、オンラインになったときにサーバからプリフェッチされます。

8.3 サーバ・ポート別のオンライン、オフラインの切り替え

上記の *M-t* による操作ではネットワークの状態を一括して切り替えますが、サーバ・ポート別にオンラインとオフラインを切り替えることもできます。

フォルダモード、サマリモードで *C-t* を押すと以下のような wl-plugged-mode に入り、このモードで各ポートの plug 状態を変更します。

```
Queuing:[ON] AutoFlushQueue:[--] DisconnectedOperation:[ON]
[ON](wl-plugged)
  [--]hosta
    [--]smtp        +queue: 2 msgs (1,2)        ...sending queue
    [--]nntp(119)   +queue: 1 msg (3)           ...sending queue
  [ON]hostb
    [--]imap4/cram-md5(143) %#mh/wl(prefetch-msgs:3,mark-as-important:1)
                            %inbox(delete-msgids:1)    ...dop queue
    [ON]nntp(119)
    [ON]smtp
```

1 行目はオフライン操作に関係する次の 3 つの変数の状態を表示しています。それぞれのラベル欄で SPC や RET を押すことで変数の値を簡単に変更できるようになっています。

```
"Queuing"               wl-draft-enable-queuing
"AutoFlushQueue"        wl-auto-flush-queue
"DisconnectedOperation" elmo-enable-disconnected-operation
```

ここで、'[ON]' はその変数の値が t であることを、'[--]' は nil であることを示しています。

また、2 行目以降ではサーバとポートのオンラインとオフライン状態を表示し、'[ON]' はそのサーバやポートがオンラインであることを、'[--]' はオフラインであることを示しています (XEmacs と Emacs 21 ではアイコンで表示されます)。そしてそれぞれの行で SPC や RET を押すことで状態を切り替えることができます。

『sending queue』 はオフライン送信時に ‘+queue’ フォルダに格納されている送信待ちのメッセージを指し、『dop queue』 はオフラインで行ったリファイル/コピー等の操作を指すとします。

そしてもし、これらの sending queue や dop queue があればその状態が画面に表示されます。上記例では、sending queue には hosta の smtp 向けに 2 つ (queue フォルダの 1 番と 2 番) と、hosta の nntp 向けに 1 つ (3 番) のメッセージがあり、dop queue には ‘%inbox’ の操作が 1 つと、‘%#mh/wl’ の操作が 2 つあることを示しています。

このモードで 2 行目にある ‘(wl-plugged)’ を変更すると、wl-plugged 変数が変更され、これによりモードラインの indicator と全体の ポート plug 状態が ON/OFF されます。また、各サーバやポートの plug 状態を変更すると、elmo-plugged-condition (後述) の設定と各ポートの plug 状態により 2 行目の ‘(wl-plugged)’ が変化します。

8.4 起動時のオフライン状態設定

前述の通り、~/.wl などで変数 wl-plugged を nil に設定してから起動すると、起動時からオフラインモードにすることができます。さらに細かくサーバやポート毎にオフライン状態を設定することも可能です。併せて変数 wl-reset-plugged-alist も参照して下さい。

通常、起動時には ~/.folders と wl-smtp-posting-server, wl-nntp-posting-server などから各ポートの plug 状態が自動的に追加されますが、これらのポートの plug 状態を変更したり、上記以外のポートを追加したりする場合には wl-make-plugged-hook に変更する関数を記述します。

```
(add-hook 'wl-make-plugged-hook
          '(lambda ()
             (elmo-set-plugged plugged 値 (t/nil) server port)
                     ;; server,port の plug 状態を新規追加もしくは変更する
             (elmo-set-plugged plugged 値 (t/nil) server)
                     ;; port を省略すると server の全 port が変更される
                     ;; (port を省略して新規の追加はできない)
             ))
```

8.5 カスタマイズ変数

wl-plugged
: この値を nil に設定して Wanderlust を起動すると、起動時からオフラインモードとなります。

wl-queue-folder
: 初期設定は ‘+queue’。送信キューのメッセージが溜るフォルダ。

wl-auto-flush-queue
: 初期設定は t。オンラインになったときに自動的にキューを送信するかどうか。Non-nil なら自動的に送信します (一応 y-or-n-p で確認します)。手動でキューを送信するには、フォルダモードで *F* を押してください。

elmo-enable-disconnected-operation
: 初期設定は t。ネットワークフォルダに関するオフライン処理を実行するかどうか。Non-nil ならオフライン処理を実行します。

elmo-lost+found-folder
: 初期設定は ‘+lost+found’。オフライン・リファイル/コピーのキュー処理でメッセージの追加に失敗したときにメッセージを退避させるフォルダです。

elmo-plugged-condition
: 初期設定は one。wl-plugged の値は関数 elmo-plugged-p (引数なし) の戻り値により決定されます。この変数 elmo-plugged-condition は (elmo-plugged-p) の戻り値が t になる条件を各ポートの plug 状態により指定します。

```
'one         : 1 つ以上のポートが plugged なら plugged である
'all         : 全てのポートが plugged なら plugged である
'independent : ポートの plug 状態に関係なく wl-plugged (elmo-plugged)
               を参照する
function     : 関数 function の戻り値により変化する
 標準で用意されている関数
 'elmo-plug-on-by-servrs
             : 変数 elmo-plug-on-servers で指定したサーバの plug
               状態により変化する
 'elmo-plug-on-by-exclude-servers
             : 変数 elmo-plug-on-exclude-servers で指定した以
外の
               サーバの plug 状態により変化する
                  elmo-plug-on-exclude-servers のデフォルト値
は
                   '("localhost"
                     (system-name)
                     (system-name) からドメイン部を除いたもの)
                  である
例 1:
 (setq elmo-plugged-condition 'all)
例 2:
 (setq elmo-plug-on-servers '("smtpserver" "newsserver"))
 (setq elmo-plugged-condition 'elmo-plug-on-by-servers)
例 3:
 (setq elmo-plug-on-exclude-servers '("localhost" "myname"))
 (setq elmo-plugged-condition 'elmo-plug-on-by-exclude-servers)
```

wl-reset-plugged-alist
: 初期設定は t。Non-nil なら Wanderlust の起動時にサーバ・ポート別のプラグ状態を wl-plugged の値により初期化します。

  nil なら、Emacs が動作している間、前回終了した時点のプラグ状態を保持します。言い換えれば nil であっても Emacs を再起動すると初期化されます。

9 メッセージの自動削除とアーカイブ

9.1 メッセージの自動削除

Expire とは、指定した期間を過ぎた古いメッセージを削除する機能です。

しかし、wl-expire ではメッセージを単純に消すだけではなく、指定したアーカイブフォルダに移動することも出来ます。

9.2 使い方

wl-expire-alistを設定して、フォルダモードで *e*、もしくはサマリモードで *M-e* を押します。

9.2.1 wl-expire-alistの設定

次に wl-expire-alistの設定例を示します。この wl-expire-alist の書き方一つで expire の実施方法が大きく変わりますので、慎重に設定してください。最初のうちは wl-expire-use-logを t にセットしておくと良いでしょう。

```
(setq wl-expire-alist
      '(("^\\+trash$"   (date 14) remove)
                                ;; 削除する。
        ("^\\+tmp$"     (date 7) trash)
                               ;; wl-trash-folder にリファイルする。
        ("^\\+outbox$"  (number 300) "$outbox;lha")
                                   ;; 特定のフォルダにリファイルする。
        ("^\\+ml/tmp$"  nil)
                        ;; expire しない
        ("^\\+ml/wl$"   (number 500 510) wl-expire-archive-number1 t)
                          ;; 番号ごとにアーカイブする (番号は保持する)。
        ("^\\+ml/.*"    (number 300 310) wl-expire-archive-number2 t)
                          ;; 一定数ごとにアーカイブする (番号は保持する)。

        ("^\\+nikki$"   (date 30) wl-expire-archive-date)
                          ;; 年月ごとにアーカイブする (番号は保持しない)。

        ))
```

各リストの要素は

（フォルダの正規表現　削除メッセージの指定　削除先）

となっています。リストの先頭から**フォルダの正規表現**にマッチするかどうかを調べます。もし、**フォルダの正規表現**にマッチしないフォルダで、expire を実行しても何もしません。また、2,3 番目の要素のいずれかが nil であれば expire しません。

削除メッセージの指定には次のものを指定します。

(number *n1* [*n2*])
: フォルダにあるメッセージ数に応じて削除を行います。

  n1 は削除後のメッセージ数で、例えば値が 500 なら最新の 500 個を残して残りを削除することになります。

n2 は expire を実行するためのメッセージの総数で、省略すると *n1* + 1 になります。例えば値が 510 ならメッセージが 510 以上のときに expire を実行することになります。これは自動実行で expire を行うようにした場合、頻繁にメールが来るフォルダでは毎回 expire を実行するようになるので、メールを読むまでに時間がかかり、煩わしくなってしまいます。そこで *n2* を *n1* よりも大きめの値に設定することで、一定数溜まるまでは expire を実行しないようにできます。

また、`wl-summary-expire-reserve-marks`で指定したメッセージ (重要マークや新規・未読マークの付いたメッセージ) は削除しないようになっていますが、もし、`wl-expire-number-with-reserve-marks` が non-nil の場合、このようなメッセージも含めて 500 個になるように expire します。`nil` の場合は上記メッセージ以外で 500 になるように expire します。

`(date` *d1*`)` メッセージの日付により削除を行います。

d1 は現在より何日前のメッセージを削除するどうかであり、例えば値が 7 なら 7 日より前のメッセージを削除します。なお、この日付とはメッセージの '`Date:`' フィールドの日付であり、メッセージがフォルダに入った日付ではないことに注意してください。

もし、メッセージに '`Date:`' フィールドがなかったり、'`Date:`' フィールドが不正な値なら、expire されませんので手で削除するなりして下さい。

削除先には次のものを指定します。

`remove` 即メッセージを削除します。

`hide` メッセージをサマリから見えなくします (削除はされません)。

`trash` メッセージを `wl-trash-folder` に移動します。

string(folder)

メッセージを *string*で指定したフォルダに移動します。

アーカイブフォルダを指定すると便利ですが、'`$`' マークの付いた重要メッセージなどは移動されないので、下記の標準関数を使う方がより良いです。

function 指定の関数を呼び出します。

指定した関数には次の 3 つの引数、フォルダ名、削除するメッセージのリスト、そしてサマリの msgdb 情報が渡されます。また、関数名の後に関数独自の引数も指定できます。なお、この関数には `wl-summary-expire-reserve-marks` で指定したメッセージも含んだリストが渡されますので、独自に関数を作る場合は注意してください。

ここで指定できる関数には、標準で次の 4 つが用意されています。そのうち 3 つは指定した方法でアーカイブフォルダにメッセージを移動するもので、古いメッセージをフォルダから削除しながら別ファイルに圧縮して保存しておくことができます。残り 1 つはメッセージを MH フォルダに振り分けるものです。

`wl-expire-archive-number1`

削除対象のメッセージ番号に対するアーカイブフォルダにリファイルします。例えば、102 番であるなら `wl-00100.zip`、390 番であるなら `wl-00300.zip`、などのようにです。なお、`wl-expire-archive-files` を 200 にすると、`wl-00000.zip`, `wl-00200.zip`, `wl-00400.zip`, ... にリファイルしていきます。

リファイル先のアーカイブフォルダは削除元のフォルダ名に基づいて次のように決定されます。(このとき、アーカイブフォルダは elmo-archive-treat-file が non-nil の場合として扱われます)

フォルダタイプが localdir の場合

ArchiveDir/*foldername*-xxxxx.zip

例えば '+ml/wl' は '$ml/wl;zip' (~/Mail/ml/wl-00100.zip) となります。

フォルダタイプが localdir 以外の場合

ArchiveDir/*foldertype*/*foldername*-xxxxx.zip

例えば、'%#mh/ml/wl' は '$imap4/#mh/ml/wl;zip' (~/Mail/imap4/#mh/ml/wl-00100.zip) となります。

すなわち、localdir の場合は種別がパス名に含まれませんが、それ以外は種別がパス名に含まれるのです。また、wl-expire-archive-folder-prefix により、アーカイブフォルダに付ける prefix を制御できます。wl-expire-archive-folder-prefixの説明を良く見ておいてください。

wl-expire-archive-number2

指定した個数ごとにアーカイブフォルダにリファイルします。

'wl-expire-archive-number1' と異なる点はメッセージ番号に関係なくアーカイブフォルダが指定数に達するまでそのフォルダにリファイルする、という点です。なお、リファイル先のアーカイブフォルダは wl-expire-archive-number1 と同じように決定されます。

wl-expire-archive-date

メッセージの日付 (年月) ごとにアーカイブフォルダにリファイルします。

例えば、1998 年 12 月のメッセージは $folder-199812;zip にリファイルされます。なお、日付の部分以外のアーカイブフォルダ名は wl-expire-archive-number1と同じように決定されます。

また、上記の 3 つの標準関数では wl-expire-alist での第 1 引数に non-nil を指定すると、フォルダのメッセージ番号をそのまま保存できます。例えば、次のように関数名の後に続けて指定します。

```
("^\\+ml/wl$" (number 300 310) wl-expire-archive-number1 t)
```

引数を指定しない場合は、各アーカイブフォルダごとに 1 から順に番号を与えて保存されます。

wl-expire-localdir-date

メッセージの日付（年月）ごとに、例えば、'+ml/wl/1999_11/', '+ml/wl/1999_12/' といった MH フォルダにリファイルします。

9.2.2 重要メッセージや未読メッセージの扱い

削除先に remove や trash、フォルダ名、標準関数のいずれを指定した場合でも、wl-summary-expire-reserve-marksで指定したマークのメッセージ (以下、『reserve メッセージ』と呼びます) は残すようになっています。

この変数にはデフォルトで、重要マーク、新規マーク、未読マークが設定されているので、これらのマークのついたメッセージは削除されないことになります。ただし、この変数には一時的マークは指定できないため (すなわち削除されるため)、expire を実行する前に一時的マークは処理しておいてください。

9.2.3 自動実行

サマリに移動したときに自動的に expire を実行するには次のように設定します。ただし、この場合は確認せずに自動実行するため、フォルダの正規表現などに誤りがないかどうかを十分確認してから設定して下さい。

```
(add-hook 'wl-summary-prepared-pre-hook 'wl-summary-expire)
```

また、フォルダモードで各フォルダごとに expire を実行できるのはもちろん、グループ単位の実行も可能です。従って、'Desktop' グループを指定すれば wl-expire-alistにマッチする全てのフォルダで expire を実行します。

9.3 TIPS

9.3.1 作成したアーカイブフォルダの取り扱い

上記の標準関数 wl-expire-archive-number1 などで作成したアーカイブフォルダを扱う場合は、変数 elmo-archive-treat-file を non-nil に設定しておく必要があります。

9.3.2 動作確認

remove を指定する場合は、まず trash にして期待通りにメールが wl-trash-folder に移動されることを確認してから remove に変えるとよいでしょう。いきなり remove を指定するのは危険です。

また、wl-expire-archive-number1などの関数を利用する場合、まずは使用するアーカイバタイプ (zip や lha) などのフォルダを試しに作って、正しく追加できるかどうかを確認してください。たとえ、wl-expire-alist や elmo-archive の設定が正しくても、アーカイブプログラムが正しく動かなければどこにも保存されずにメッセージが消えてしまうかも知れません。

アーカイブフォルダの動作が確認でき、実際に expire を実行するようになれば、ログを活用してください。wl-expire-use-log を t にすると、~/.elmo/expired-logには以下のような記録が残ります。

```
delete  +ml/wl  (593 594 595 596 597 598 599)
move    +ml/wl -> $ml/wl-00600;tgz;wl  (600 601 602)
```

最初の項目は動作を示すもので、'delete', 'copy', 'move' があります。次が expire を実行したフォルダ名で、'copy' と 'move' の場合は '->'に続けてコピーもしくは移動先のフォルダ名が記録されます。最後の項目は、実際に削除や移動されたメッセージ番号のリストです ('copy' や 'move' の場合、移動後ではなく移動前のメッセージ番号です)。

9.3.3 reserve メッセージのリファイル

標準で用意されている 3 つの関数では、reserve メッセージはアーカイブフォルダにコピーしますが、元のフォルダからは削除しないようになっています。なお、重要マークなどは常に残るため、何度もコピーされることがないように~/.elmo/expired-alist に記録するようになっています。ただしこれは reserve メッセージが refile 対象になったときの話です。wl-summary-archive などでコピーされる場合は記録を残しません。

ログ機能を有効にしていた場合は、リファイル時には通常 'move' が記録されますが、reserve メッセージが含まれていると、'copy' と 'delete' に分けて記録されます。これは reserve メッセージを含めたメッセージをコピーした後、reserve メッセージを除いたメッセージを削除する、という処理を行っているためです。

9.4 カスタマイズ変数

wl-expire-alist
: 初期設定は nil。expire を行うフォルダと expire 方法の指定を行います。詳しくは上記の wl-expire-alistの設定をご覧下さい。

wl-summary-expire-reserve-marks
: 初期設定は以下のリスト。

```
(list wl-summary-flag-mark
      wl-summary-new-uncached-mark
      wl-summary-new-cached-mark
      wl-summary-unread-uncached-mark
      wl-summary-unread-cached-mark)
```

expire を行っても、フォルダには残しておくメッセージのマークを指定します。マークには永続的マークのみ指定できます。一時的マークは指定できません。

デフォルトのようにリストで指定するとそのマークのメッセージを残せる他、以下の指定もできます。

all
: 永続マークの付いたすべてのメッセージを残します。つまり、デフォルトで設定されているマーク以外に wl-summary-read-uncached-mark が含まれます。

none
: どんなマークの付いたメッセージであっても、通常の既読メッセージと同じ扱いをします。すなわち、'$' マークの付いた重要メッセージであっても削除されます。

wl-expire-archive-files
: 初期設定は 100。ひとつのアーカイブフォルダに保持するメッセージ数を指定します。

wl-expire-number-with-reserve-marks
: 初期設定は nil。Non-nil にすると、削除メッセージの指定で number を指定したとき、残しておくメッセージ数に wl-summary-expire-reserve-marks で設定されたメッセージを含めます。

wl-expire-archive-get-folder-function
: 初期設定は wl-expire-archive-get-folder。

削除先の標準関数でアーカイブフォルダ名を取得する関数を指定します。次の 3 つの変数により簡易なフォルダ名の変更できますが、もっと複雑な指定をしたい場合は新たに関数を作ってこの変数に設定します。

関数 wl-expire-archive-get-folderのカスタマイズ変数には次のものがあります。

- wl-expire-archive-folder-name-fmt
- wl-expire-archive-folder-type

- wl-expire-archive-folder-prefix

wl-expire-archive-folder-name-fmt
: 初期設定は '%s-%%05d;%s'。wl-expire-archive-number1および wl-expire-archive-number2で使用されるアーカイブのフォルダの format 形式の文字列を指定します。なお、2 度 format で指定するため、番号の部分は必ず '%%d' にしなくてはなりません。

  もし、変更する場合は wl-expire-archive-folder-num-regexp も合わせるようにしてください。

wl-expire-archive-date-folder-name-fmt
: 初期設定は '%s-%%04d%%02d;%s'。wl-expire-archive-date で使用されるアーカイブのフォルダの format 形式の文字列を指定します。なお、2 度 format で指定するため、番号の部分は必ず '%%d' にしなくてはなりません。また、メッセージの年と月を与えるため、'%%d' は 2 つ必要です。

  もし、変更する場合は wl-expire-archive-date-folder-num-regexp も合わせるようにしてください。

wl-expire-archive-folder-type
: 初期設定は zip。アーカイブフォルダのアーカイバタイプを指定します。

wl-expire-archive-folder-prefix
: 初期設定は nil。アーカイブフォルダに付ける prefix を指定します。ただし、アーカイブフォルダに prefix (ディレクトリ構造) を付ける仕様はおまけ機能ですので、取り扱いは慎重に行って下さい。最悪の場合，アーカイブファイルを壊す恐れがあります。

  nil
  : prefix は付きません。

  short
  : 例えば、'+ml/wl' では prefix 'wl' が付き、'$ml/wl-00000;zip;wl' となります。

  t
  : 例えば、'+ml/wl' では prefix 'ml/wl' が付き、'$ml/wl-00000;zip;ml/wl' となります。

wl-expire-archive-folder-num-regexp
: 初期設定は '-\\([-0-9]+\\);'。elmo-list-folders による複数のアーカイブフォルダ名から番号を取得するための正規表現を指定します。wl-expire-archive-folder-name-fmt に対応して設定してください。

wl-expire-archive-date-folder-num-regexp
: 初期設定は '-\\([-0-9]+\\);'。elmo-list-folders による複数のアーカイブフォルダ名から番号を取得するための正規表現を指定します。wl-expire-archive-date-folder-name-fmt に対応して設定してください。

wl-expire-delete-oldmsg-confirm
: 初期設定は t。Non-nil の場合、既に存在しているアーカイブフォルダの最大メッセージ番号よりも古いメッセージがあった場合に確認してから削除します。nil の場合は確認せずに削除します。

  なお、標準関数の引数に non-nil を指定して番号を保持するようにした場合のみ有効です。

wl-expire-use-log
: 初期設定は nil。Non-nil にすると、~/.elmo/expired-logに expire の実行結果を記録します。なお、ファイルに追加していく一方なので、適当に手で消す必要があります。

wl-expire-add-seen-list
: 初期設定は t。Non-nil の場合、expire によりメッセージをリファイルした場合、既読情報をリファイル先のフォルダに伝えるようにします。

  ただし、リファイル先のフォルダを Wanderlust 上から読まないと、~/.elmo/ 以下にある seen ファイルが大きくなっていくので、アーカイブフォルダなどに単に保存しておくだけなら nil に設定しておくと良いでしょう。nil に設定しても、リファイルしたアーカイブフォルダを読むときに新規メッセージ扱いされるだけで、expire などの動作には影響はありません。

wl-expire-folder-update-msgdb
: 初期設定は t。t の場合、フォルダモードで expire を実行するときに、サマリ情報を update してから expire を実行する。また、フォルダ名の正規表現のリストを指定した場合は、マッチしたフォルダのみサマリ情報を update する。

9.5 メッセージのアーカイブ

9.5.1 メッセージのアーカイブ

M-x wl-summary-archive でフォルダ全体をアーカイブフォルダにコピーします。既にアーカイブフォルダがある場合、新規メッセージのみ追加します。

wl-expire-alist と同じ様に、フォルダ名に応じてどのようにアーカイブするかを wl-archive-alist で指定します。例えば以下のようになります。

```
(setq wl-archive-alist
      '(("^\\+tmp$"     wl-archive-date)
        ("^\\+outbox$"  wl-archive-number2)
        (".*"           wl-archive-number1)))
```

各リストの要素は次のようになります。

（フォルダの正規表現　削除関数）

このようにフォルダの正規表現の後には関数しか指定できません。標準では次の 3 つの関数

- wl-archive-number1
- wl-archive-number2
- wl-archive-date

が用意されてます。名前からお解りの通り、次の点を除いて Expire 用に用意されている物と同じ動作をします。

- メッセージを削除しない
- 引数なしであってもメッセージ番号を保持する

フォルダの全メッセージを番号ごとや日付ごとにまとめてアーカイブしたい場合は、これらの関数を使用するとよいでしょう。また、expire を行う前のバックアップや動作を確

認するのにも有効です。もっとも、アーカイブ後に expire でリファイルすると、リファイルせずに削除するだけになります。

デフォルトではコピー先のアーカイブフォルダは `wl-expire-archive-get-folder-function` に従って自動的に決定されますが、prefix argument を付けて *C-u M-x wl-summary-archive* で実行すると、指定したフォルダにコピーすることができます。

しかし、単純に 1 つのアーカイブフォルダにコピーするだけなら、`wl-summary-copy-region` などで全メッセージをアーカイブフォルダにコピーすることでも可能なため、おまけの機能でしかありません (つまり、動作確認は不十分です)。

このアーカイブフォルダの決定方法は `wl-summary-expire` と同じものを用いているため、カスタマイズ変数の中で次に示すものが関係してきます。

- `wl-expire-archive-files`
- `wl-expire-archive-get-folder-function`
- `wl-expire-archive-folder-name-fmt`
- `wl-expire-archive-folder-type`
- `wl-expire-archive-folder-prefix`
- `wl-expire-archive-folder-num-regexp`

9.5.2 カスタマイズ変数

`wl-archive-alist`

初期設定は以下のリスト。

```
((".*" wl-archive-number1))
```

アーカイブフォルダにコピーする処理を行う関数を指定します。この関数には、フォルダ名、フォルダ内にあるメッセージのリスト、サマリの msgdb 情報、の 3 つの引数が渡されます。もちろんユーザが独自に作って指定することができます。

10 スコア

スコアとは、メッセージにスコア (値) をつけ、その値により既読マークを付けたりサマリから消したりする機能です。

この機能によって重要なメッセージにまとめ処理用マーク '*' や重要マーク '$'をつけたり、spam 記事などの読みたくないメッセージに既読マークをつけたりすることができます。

このスコア機能は Gnus のスコアとほぼ同等の機能を持ち、またスコアファイルの書式もほぼ同じです。ただし、幾つかは未対応であったり Wanderlust 特有の機能があったりします。*Gnus Manual* の "Scoring" 節 を参照してください.

10.1 スコアに関するコマンド

10.1.1 スコアファイルの指定方法

変数 `wl-score-folder-alist` にフォルダ名に対応したスコアファイル名かスコアを定義した変数を設定します。

```
(setq wl-score-folder-alist
      '(("^-.*"
         "news.SCORE"
         "my.SCORE")
        (".*"
         "all.SCORE")))
```

スコアファイル名のパスを省略した場合は、変数 `wl-score-files-directory` で指定したディレクトリにあるものとします。

また、`wl-score-folder-alist` の設定に関わらずデフォルトのスコアファイル `wl-score-default-file` (`all.SCORE`) は必ず読み込まれます (ファイルが存在していなくても構いません)。したがって、上記例の '`^-.*`' にマッチしたフォルダでは `news.SCORE`, `my.SCORE`, `all.SCORE` の 3 つのスコアファイルが読み込まれることになります。

10.1.2 スコアファイルの対象メッセージ

スコアはサマリの update 時に一時的に `wl-summary-score-marks` で指定したメッセージのみにつけられます。つまりサマリから抜けるとメッセージにつけられたスコアは消去され、デフォルトのスコア値に戻ります。

10.1.3 スコアファイルの作成

まずサマリバッファで適当なメッセージに移動してから *L* をタイプします。その後ミニバッファでの入力を求められますので、続けて *s*, *s*, *p* とタイプしてみてください。すると Subject の文字列が入力されている状態になりますので、適当に編集した後 `RET` を押します。

これで、入力した文字列と同じ '`Subject:`' を持つメッセージに対してスコア −1000 がつけられるようになります。つまり、このようなスコアファイルが自動的に作成されたことになります。

次に、同じサマリバッファで *h e* とタイプしてください。すると先ほど作成したスコアファイルが表示されていると思います。このバッファを『スコア編集バッファ』と呼びます。このスコア編集バッファで *C-c C-e* とタイプすると、ミニバッファで先ほどと同じような

入力を求められると思いますが、ここで a とタイプしてください。今度はすぐに "From" のスコアエントリが挿入されたはずです。このようにしてサマリバッファでもスコア編集バッファでもスコアファイルを簡単に作成することができます。

ところで、ミニバッファでの入力時に *s s p* または *a* とタイプしたように、キータイプ数が違っていたと思います。これは、wl-score-header-default-entry の設定によるものです。この変数ではヘッダに応じたデフォルトのスコアエントリを設定します。たとえば、"subject" ヘッダでは型と期限についての入力を求めますが、"from" ヘッダでは型は substring, 期限は永続に自動的に決定されます。ただし、スコアの値は prefix argument で強制的に変更することができます。また、ミニバッファでの入力時に ? とタイプすることでキーとそれに対応するヘッダや型を (help) を表示します。

では最後に、スコア編集バッファで *C-c C-c* と入力して下さい。これでスコアファイルを保存して編集モードを終了します。バッファの内容を消去してから *C-c C-c* すると編集中のスコアファイルを削除します。

10.1.4 TIPS

10.1.4.1 スコアファイルの選択

wl-summary-increase-score と wl-summary-lower-score とで追加するスコアファイルは wl-score-change-score-file で変更することができます。

10.1.4.2 スコアの加算

wl-summary-increase-score や wl-summary-lower-score、wl-score-edit-insert-entry で同じエントリを追加した場合、スコアが加算されます。

たとえば、*L a* でスコアが −1000 の 'from' エントリを作成した後、再度 *C-u 200 L a* でスコアが −200 の 'from' エントリを作成すると、スコアが −1200 のエントリが 1 つ作成されることになります。

10.1.4.3 Thread キーの作成

wl-summary-increase-score か wl-summary-lower-score で 'Thread'キーを作成すると、子スレッドの 'Message-ID' も全て追加されます。

10.1.4.4 Followup キーの作成

wl-summary-increase-score か wl-summary-lower-score で 'Followup'キーを作成すると、カーソル上のメッセージの 'Message-ID' も 'References' キーに追加されます。もし、wl-score-auto-make-followup-entry が non-nil であれば wl-score-expiry-days で指定した日にち以内の全 followup 対象のメッセージの 'Message-ID' が追加されます。

10.1.5 キーバインド

K 現在のメッセージのスコアを高くします。同時にスコアエントリがスコアファイルに追加されます。また、prefix argument でスコアの値を設定することができます。

L 現在のメッセージのスコアを低くします。同時にスコアエントリがスコアファイルに追加されます。また、prefix argument でスコアの値を設定することができます。

h R スコアを適用し直します。ただし、既にスコアがつけられているメッセージには、新たにスコアはつきません。

h c 現在選択しているスコアファイルを変更します。

h e 現在選択しているスコアファイルを編集します。スコアファイルが複数ある場合先に指定されたファイルが選択されます。

h f 任意のスコアファイルを編集し、このスコアファイルを選択します。

h F 読み込んだスコアファイルは一旦キャッシュされますが、そのキャッシュを消去します。Wanderlust 以外で直接スコアファイルを変更した場合は、キャッシュを消去して再読み込みする必要があります。

h m 既読マークを付ける (読んだことにする) スコア基準値を設定します。この値よりも小さなスコアが既読になります。

h x サマリから消去するスコア基準値を設定します。この値よりも小さなスコアが消去されます。消去といっても表示されないだけであり、サマリ情報やフォルダからは削除されません。消去されたメッセージは rescan-noscore により再び表示することができます。

10.1.6 スコア編集バッファのキーバインド

C-c C-k 編集中のファイルを破棄します。

C-c C-c 編集中のファイルを保存して、編集モードを終了します。

C-c C-p スコアを綺麗に表示し直します。

C-c C-d 紀元前 1 年 12 月 31 日からの日数を挿入します。期限付きのスコアを作るときに期限の要素 (3 番目) に使用します。

C-c C-s サマリバッファで選択しているメッセージのヘッダを挿入します。

C-c C-e サマリバッファで選択しているメッセージのスコアエントリを追加します。

10.1.7 カスタマイズ変数

`wl-summary-default-score`
初期設定は 0。スコアのデフォルト値を設定します。この値を元にスコアが加減されます。

`wl-summary-important-above`
初期設定は `nil`。この値より大きいスコアに対して重要マーク (‘`$`’) をつけます。`nil` の場合はマークを付けません。

`wl-summary-target-above`
初期設定は `nil`。この値より大きいスコアに対してまとめ処理用マーク (‘`*`’) をつけます。`nil` の場合はマークを付けません。

`wl-summary-mark-below`
初期設定は 0。この値より小さなスコアに対して既読マークをつけます (読んだことにします)。

`wl-summary-expunge-below`
初期設定は `nil`。この値より小さなスコアはサマリから消去します。`nil` の場合は消去しません。

`wl-summary-score-marks`
: 初期設定は以下のリスト

```
(list wl-summary-new-uncached-mark
      wl-summary-new-cached-mark)
```

スコアをつけるメッセージのマークを指定します。

`wl-use-scoring`
: 初期設定は `t`。Non-nil ならスコア機能を有効にします。

`wl-score-files-directory`
: 初期設定は `~/.elmo/`。スコアファイルをデフォルトのディレクトリを指定します。

`wl-score-interactive-default-score`
: 初期設定は 1000。スコアファイルでスコア要素が `nil` の時に用いるスコアを指定します。また、`wl-summary-increase-score` や `wl-summary-lower-score` でつけるスコア値でも用いられます。ただし、`wl-score-header-default-entry` のスコア値が `nil` の時。

`wl-score-expiry-days`
: 初期設定は 7。期限付きスコアを削除する日数を指定します。

`wl-score-update-entry-dates`
: 初期設定は `t`。Non-nil なら期限付きスコアを削除する機能を有効にします。

`wl-score-header-default-entry`
: `wl-summary-increase-score` や `wl-summary-lower-score`、`wl-score-edit-insert-entry` でスコアエントリを作成する場合の各ヘッダのデフォルト値を設定します。

`wl-score-simplify-fuzzy-regexp`
: スコアエントリの型で `fuzzy` を指定した場合、文字列から削除する正規表現を指定します。`Subject` で使用されることが多いので、デフォルトではメーリングリストプログラムでつけられる prefix を指定しています。

`wl-summary-rescore-partial-threshold`
: 初期設定は 200。sync-all や rescan が実行されたときに、この値を越えるメッセージがサマリにある場合、サマリの最後から指定された数のメッセージだけ、部分的にスコア付けが適用されます。

`wl-summary-auto-sync-marks`
: Non-nil ならば、サマリ同期時に未読/重要マークも同期します。未読マークは、IMAP4 サーバ上の未読情報が反映されます。重要マークは IMAP4 サーバ上の重要情報 (Flagged フラグがついているか)、および ‘’`flag`’ フォルダの内容が、反映されます。初期設定は `t`。

10.2 スコアファイル書式

スコアファイルの書式は Gnus と同じなので、Gnus で使用していたスコアファイルがそのまま利用できます。ただし、幾つかのキーは対応していなかったり Wanderlust 特有のキーがあったりしますので、完全に互換性があるわけではありません。*Gnus Manual* の “Score File Format” 節 を参照してください.

```
(("subject"
  ("for sale" -1000 nil s)
  ("儲け" -1000 nil s))
 ("from"
  ("spam@spamspamspam" -10000 nil s))
 ("followup"
  ("my@address" 3001 nil s))
 ("chars"
  (1000000 -10 nil >))
 (important 5000)
 (target 3000)
 (mark 0)
 (expunge -3000))
```

文字列 (STRING)
: キーが文字列である場合、マッチさせるヘッダの名前を指定します。このキーには次のものが指定できます。Subject, From, Date, Message-Id, References, To, Cc, Chars, Lines, Xref, Extra, Followup, Thread この中で、Chars はメッセージのサイズを指し、Extra, Followup, Thread については後述します。残りはキーと同じ名前のフィールドが対象となります。

  このキーの後にスコアエントリを任意の数だけ指定し、この各スコアエントリは次の 5 つの要素からなります。

  1. ヘッダにマッチする要素。lines と chars の場合は数字で、それ以外は文字列を指定します。
  2. スコア要素。1 番目の要素がマッチした場合、そのメッセージのスコアをこの値分増減させます。
  3. 期限の要素。nil なら永続 (permanent) 指定で、数字 (日数) なら一定期間 (wl-score-expiry-days) マッチしないと削除されます。この日数は紀元前 1 年 12 月 31 日から経過した日にちです。
  4. 型の要素。1 番目の要素をマッチさせる方法を指定します。キーによって指定できる型が異なります。

    『From, Subject, References, Message-Id』
    : これらの文字列のキーに対しては、r と R (正規表現) (regexp) や、s と S (文字列の一部) (substring)、e と E (正確な合致) (exact match)、それに f と F (あいまい) (fuzzy) が指定できます。R, S, E, F は大文字小文字を区別してマッチさせます。

    『Lines, Chars』
    : これらは数字の大小を指定します。その記号は次の 5 つです。
      <, >, =, >=, <=

    『Followup』
    : このキーは、Fromヘッダにマッチし、そのメッセージへの全てのフォローアップに対してスコアをつけます。たとえば、自分自身の記事へのフォローアップのスコアを増やしたりするのに便利です。

f を除いて From キーと同じ型が指定出来ます。また、自動的にスコアファイルに 'Followup' エントリが追加されます。

『Thread』 このキーは、Message-ID x で始まっている (サブ) スレッドにスコアを付ける場合に指定します。これは References ヘッダに x を持つそれぞれの記事に新しい 'Thread' エントリを自動的に追加します。これにより、全ての祖先の Message-ID を References に含んでいない場合でも、確実にスレッド全体のスコアを増減させることができます。

f を除いて References キーと同じ型が指定出来ます。また、自動的にスコアファイルに 'Thread' エントリが追加されます。

5. 拡張ヘッダの要素。キーが Extra である場合のみ意味を持ちます。Subject や From などの標準以外のヘッダにマッチさせたい場合にそのヘッダを指定します。ただし、指定したヘッダは elmo-msgdb-extra-fields にも設定する必要があります。したがって、拡張ヘッダが取得できないフォルダでは機能しません。

そしてこれらの全ての要素を当てはめた後の合計のスコアがそのメッセージのスコアとなります。

mark この値より小さいスコアのメッセージには既読マークをつけます。デフォルト値は wl-summary-mark-below で指定されます。

expunge この値より小さいスコアのメッセージはサマリから消去します。デフォルト値は wl-summary-expunge-below で指定されます。

mark-and-expunge
mark と expunge を同時に指定します。つまり、この値より小さいスコアのメッセージは既読マークをつけ、サマリから消去します。

target この値より大きいスコアのメッセージにはまとめ処理用マーク '*' をつけます。デフォルト値は wl-summary-target-above で指定されます。

important
この値より大きいスコアのメッセージには重要マーク '$' をつけます。デフォルト値は wl-summary-important-above で指定されます。

10.2.1 注意事項

extra キーはもちろん、lines と xref キーを使用する場合でも、elmo-msgdb-extra-fields を設定する必要があります。

```
(setq elmo-msgdb-extra-fields '("lines" "xref"))
```

その他、下記の制限事項があります。

- サマリ情報に含まれる 'References' フィールドには最後の 'Message-ID' しか存在しないため、referencesキーもその 'Message-ID' にしかマッチしない。

フォルダ種別により参照できるキーの一覧。

	chars	lines	xref	extra
localdir,localnews	○	△	△	△
nntp（xover 対応）	○	△	△	
（xover 非対応）		△	△	△
imap4	○	△	△	△
pop3		△	△	△

○: 参照できる
: 参照できない（無視される）
△: elmo-msgdb-extra-fields を設定すれば参照できる

11 アドレス帳

アドレス帳を利用することで、メールアドレスを簡単に入力したり，サマリの表示にペットネームを用いることが出来ます。

11.1 アドレス帳の定義

アドレスファイル ~/.addresses を作成し、自分用に編集します。~/.addresses に書かれたデータは、ドラフト作成時のアドレス補完データとして利用されるほか、サマリ表示での名前表示等にも用いられます。なお、起動した状態でサマリバッファから ~/.addresses にアドレスを追加/ 変更/削除することも可能です。

書き方はとても単純です。こんな感じです。

```
#
# '#' で始まる行はコメント。
# 空行は無視。
#
# メールアドレス  "あだ名"  "本名"
#
teranisi@gohome.org             "てらにし"      "寺西裕一"
foo@bar.gohome.org              "Foo さん"      "John Foo"
bar@foo.gohome.org              "Bar さん"      "Michael Bar"
```

一行が一人分の定義です。

実際には (デフォルト設定では) サマリ表示で**あだ名**、ドラフト作成時のアドレス情報として**本名**が使われます。試してみて、確認してからの方がわかりやすいと思われます。ちょっと書いて試してみてから、またアドレス帳の定義をやり直すのが良いでしょう。

また、変数 wl-alias-file に MH の alias file が指定されていれば、ドラフト作成時のアドレス情報として使われます。

さらに、変数 wl-use-ldap (初期設定は nil) を non-nil に設定すると、LDAP サーバの情報をドラフト作成時のアドレス情報として利用します。

LDAP を利用する場合は、wl-ldap-server、wl-ldap-port, wl-ldap-base も適切に設定して下さい。また、LDAP 対応の XEmacs 以外では、外部プログラムとして ldapsearch を利用しますので、ldapsearch へあらかじめコマンド実行パスを設定しておく必要があります。

11.2 アドレスマネージャ

C-c C-a とするとアドレスマネージャを起動されます。アドレスマネージャではアドレス帳の編集を行うことが出来るのに加えて、アドレスマネージャで指定したアドレスをドラフトバッファへ挿入することができます。

11.2.1 キーバインド

t　　'To:' マークをつけます。

c　　'Cc:' マークをつけます。

b　　'Bcc:' マークをつけます。

u マークを取り消します。

x 宛先マークがついている場合、それらのアドレスをドラフトバッファに反映してアドレスマネージャを終了します。ドラフトバッファが無い場合、それらを反映して新規にドラフトバッファを開きます。宛先マークがついていない場合は単にアドレスマネージャを終了します。

q アドレスマネージャを終了します。

a アドレス帳に新しい項目を追加します。

d アドレス帳の項目を削除します。

e アドレス帳の項目を編集します。

12 Quick Search

`wl-qs` provides an interface to quickly search your mail archive. It can use an external search engine (3.10 節 「Search Folder」(16 ページ)), Gmail search, or a filter folder (3.12 節 「Filter Folder」(19 ページ)).

`wl-qs` provides the command `wl-quicksearch-goto-search-folder`. Using it will first prompt for a search, and then jump to the search results.

12.1 Setup of `wl-qs`

To setup, configure the value of `wl-quicksearch-folder`. This should be the name of the folder you would like to search. For example, '`%[Gmail]/All Mail:username@imap.gmail.com`', '`.archive`' or '`[]`'. The latter is advised if you use a mail index, such as mu, notmuch or namazu, as it is quite fast.

12.2 Searching

To search your mail archive, use the command `wl-quicksearch-goto-search-folder`, which can be called using `'` in a Summary buffer or the Folder buffer. You will be prompted for a search, and then will immediately jump to the search results.

12.2.1 Search folder

If you use specified '`[`' as the value of `wl-quicksearch-folder`, you will be accessing a search folder (3.10 節 「Search Folder」(16 ページ)). You will be prompted for a search string. The syntax of the search will depend on the value of `elmo-search-default-engine`. Quotes will be escaped for you automatically and passed on to the search program.

(If you are using the grep search engine, you must specify a target folder. Your `wl-quicksearch-folder` should look like '`[]~/Mail/semi!grep`'.)

12.2.2 Gmail

If you use a gmail folder as your `wl-quicksearch-folder`, you will be prompted for a Gmail search query (`https://support.google.com/mail/answer/7190`). You may use any Gmail search operator; the search is handled by Gmail's server.

12.2.3 Filter folder

If you are using any other type of folder, you will be prompted for a query using the interactive query builder. When you have finished your query, you will be directed to a filter folder for your `wl-quicksearch-folder`.

13 Spam フィルタ

wl-spam は、外部 spam フィルタプログラムへのフロントエンドを提供します。Wanderlust 上でのメッセージに対する操作と連携して、フィルタプログラムへの登録や、spam の判定ができるようになります。

13.1 使い方

13.1.1 初期設定

wl-spam を使うには、まず ~/.wl に以下のように設定して下さい。

```
;; 'bogofilter' を使う場合。
;; ここで、使いたい spam フィルタの 'scheme' を設定して下さい。
;; 13.2 節 「Spam Filter Processors」(92 ページ) を参照してください.
(setq elmo-spam-scheme 'bogofilter)
(require 'wl-spam)
```

13.1.2 spam マーク

一時的マークに spam マーク ('s') が追加されます。このマークの付いたメッセージは、アクションの実行時に wl-spam-folder にリファイルされます。また、デフォルトの設定ではサマリでの通常の移動でスキップされるようになります。

spam マークは、後述する spam の判定処理で自動的に付く他、*k m* と押して任意に付けることも出来ます。

13.1.3 spam の判定

以下の方法で spam の判定を行うことが出来ます。

1. 自動リファイルの実行時に判定する。

   以下の設定例のように wl-auto-refile-guess-functions の任意の位置に wl-refile-guess-by-spam を挿入します。

   ```
   (setq wl-auto-refile-guess-functions
         '(wl-refile-guess-by-rule
           wl-refile-guess-by-spam))
   ```

   この例の場合、wl-refile-rule-alist で振り分け先が決まらなかった時に spam かどうかを判定するようにしています。

2. 特定のフォルダのサマリに移動した時に判定する。

   wl-spam-auto-check-folder-regexp-list に自動判定を行いたいフォルダ名の正規表現のリストを設定します。

   ```
   (setq wl-spam-auto-check-folder-regexp-list '("\\+inbox"))
   ```

   この例の場合、フォルダ名に '+inbox' を含むフォルダのサマリに移動した時に判定処理が実行されます。

3. elmo-split によるメッセージの振り分け時に判定する。

   elmo-split-rule の 'CONDITION' として指定出来る関数に spam-p が追加されます。この関数は、対象のメッセージが spam と判定された時に真となります。14.5 節「Split messages」(103 ページ) を参照してください.

また、判定結果を元に学習させることも出来ます。(ある程度学習が進んでから、この設定を有効にするとよいでしょう)

以下に例を示します。

```
(setq elmo-split-rule
      '(((spam-p) "+spam")
        ;; 判定結果を元に学習させる場合は代わりに下の条件を使う
        ;((spam-p :register t) "+spam")
        (t "+inbox")))
```

13.1.4 spam の学習

wl-spam は、メッセージをリファイルすることで、自動的に spam を学習します。

まず、wl-spam は Wanderlust の管理するフォルダをそこに含まれるメッセージの区分によって、以下の 4 つの領域に分類します。

'spam' spam と判定されたメッセージがあるフォルダ。(wl-spam-folder に設定されたフォルダ)

'good' non-spam と判定されたメッセージがあるフォルダ。

'undecide' 未判定のメッセージがあるフォルダ。'+inbox' 等、自身で振り分けていないメッセージがあるフォルダが該当します。(wl-spam-undecided-folder-regexp-list で設定)

'ignored' wl-trash-folder や wl-draft-folder 等、spam の処理とは関係のないフォルダ。(wl-spam-ignored-folder-regexp-list で設定)

メッセージをリファイルした時、そのメッセージの属する領域が変わった場合、前後の領域に従って 'spam' または、'non-spam' として学習します。

具体的には以下の通りです。

'undecide -> spam' spam として学習。

'good -> spam' spam としての学習に加えて、non-spam に行なった学習を削除します。

'undecide -> good' non-spam として学習。

'spam -> good' non-spam としての学習に加えて、spam に行なった学習を削除します。

上記以外のリファイルでは、学習は行われません。

13.1.5 キーバインド

k m カーソル行のメッセージに spam マーク ('s') を付けます。

k c カーソル行のメッセージをテストし、spam と判定された場合に spam マークを付けます。spam でないと判定された場合は spam マークを取り除きます。

k C `wl-spam-auto-check-marks` に含まれるマークを持つメッセージについて spam かどうかのテストを行います。spam と判定されたメッセージには、spam マークが付けられます。prefix argument をつけた場合は、マークにかかわらず全てのメッセージを対象とします。

k s カーソル行のメッセージを spam として登録し spam マークを付けます。

k S フォルダ内の全てのメッセージを spam として登録し spam マークを付けます。

k n カーソル行のメッセージを non-spam として登録し spam マークを取り除きます。

k N フォルダ内の全てのメッセージを non-spam として登録し spam マークを取り除きます。

r k m 指定リージョンにあるメッセージに spam マークを付けます。

r k c 指定リージョンにあるメッセージをテストし、spam と判定された場合に spam マークを付けます。spam でないと判定された場合は spam マークを取り除きます。

r k s 指定リージョンにあるメッセージを spam として登録し spam マークを付けます。

r k n 指定リージョンにあるメッセージを non-spam として登録し spam マークを取り除きます。

t k m カーソル行があるメッセージを先頭とするスレッドのメッセージに spam マークを付けます。prefix argument つきならばカーソル行があるメッセージを含むスレッド全てに適用します。

t k c カーソル行があるメッセージを先頭とするスレッドのメッセージをテストし、spam と判定された場合に spam マークを付けます。spam でないと判定された場合は spam マークを取り除きます。prefix argument つきならばカーソル行があるメッセージを含むスレッド全てに適用します。

t k s カーソル行があるメッセージを先頭とするスレッドのメッセージを spam として登録し spam マークを付けます。prefix argument つきならばカーソル行があるメッセージを含むスレッド全てに適用します。

t k n カーソル行があるメッセージを先頭とするスレッドのメッセージを non-spam として登録し spam マークを取り除きます。prefix argument つきならばカーソル行があるメッセージを含むスレッド全てに適用します。

m k まとめ処理用マーク '`*`' のついたメッセージに spam マーク ('`s`') を付けます。

m s まとめ処理用マーク '`*`' のついたメッセージを spam として登録し spam マークを付けます。

m n まとめ処理用マーク '`*`' のついたメッセージを non-spam として登録し spam マークを取り除きます。す。

13.1.6 カスタマイズ変数

`wl-spam-folder`
spam と判定されたメッセージを移動するフォルダ名を設定します。初期設定は、'`+spam`'。

`wl-spam-undecided-folder-regexp-list`
: spam か non-spam か未判定のメッセージがあると看做すフォルダを、フォルダ名の正規表現のリストで指定します。初期設定は、`'("inbox")`

`wl-spam-ignored-folder-regexp-list`
: 初期設定は以下の通り。

```
(list (regexp-opt (list wl-draft-folder
                        wl-trash-folder
                        wl-queue-folder)))
```

spam 判定に対して無効なフォルダを、フォルダ名の正規表現のリストで指定します。

`wl-spam-auto-check-folder-regexp-list`
: サマリに移動した時に自動的に spam 判定を行うフォルダを正規表現のリストで指定します。初期設定は、`nil`。

`wl-spam-auto-check-marks`
: 初期設定は以下のリスト。

```
(list wl-summary-new-uncached-mark
      wl-summary-new-cached-mark)
```

`wl-spam-auto-check-folder-regexp-list` による自動判定を含む、フォルダ全体に対する spam 判定の対象とするメッセージのマークを指定します。マークには永続マークのみ指定できます。一時的マークは指定できません。

デフォルトのようにリストで指定するとそのマークのついたメッセージだけを対象とする他、以下の指定もできます。

`all` 永続マークが何であっても spam 判定の対象とします。

13.2 対応している Spam Filter

デフォルトでは、以下の spam フィルタリングライブラリに対応しています。

13.2.1 bogofilter

bogofilter (`http://bogofilter.sourceforge.net/`) は、C 言語で実装された spam フィルタです。

bogofilter による spam フィルタを使用するには、`~/.wl` などに以下の設定を記述します。

```
(setq elmo-spam-scheme 'bogofilter)
```

13.2.1.1 カスタマイズ変数

`elmo-spam-bogofilter-program`
: 初期設定は、`bogofilter`。bogofilter の実行ファイルの名前を設定します。実行ファイルが、環境変数 `PATH` 上にない場合は、フルパスを設定する必要があります。

`elmo-spam-bogofilter-args`
: 初期設定は、`nil`。bogofilter の実行時に実行ファイルに渡される引数を指定します。

elmo-spam-bogofilter-database-directory
: 使用する統計データベースの存在するディレクトリを指定します。nil ならデフォルトの位置 (~/.bogofilter) が使用されます。初期設定は、nil

elmo-spam-bogofilter-max-messages-per-process
: 初期設定は、30。学習時にまとめ処理されるメッセージの数を指定します。

elmo-spam-bogofilter-debug
: 初期設定は、nil。non-nil に指定すると、bogofilter からの出力が"*Debug ELMO SPAM Bogofilter*"というバッファに出力されます。

13.2.2 spamfilter.el

spamfilter.el (http://www.geocities.co.jp/SiliconValley-PaloAlto/7043/) は、Emacs Lisp で実装された spam フィルタリングライブラリです。

インストール時に load-path 上に spamfilter.el があれば、自動的に対応モジュールがコンパイル/インストールされます。2.3 節 「Install」(4 ページ) を参照してください.

spamfilter.el を使用するには、~/.wl などに以下の設定を記述します。(もちろん、spamfilter.el 自体の設定も必要です)

```
(setq elmo-spam-scheme 'spamfilter)
```

13.2.2.1 カスタマイズ変数

elmo-spam-spamfilter-corpus-filename
: 初期設定は ~/.elmo/.spamfilter。コーパスファイルの名前を設定します。

13.2.3 bsfilter

bsfilter (http://bsfilter.org/) は、Ruby で実装された spam フィルタです。

bsfilter による spam フィルタを使用するには、~/.wl などに以下の設定を記述します。

```
(setq elmo-spam-scheme 'bsfilter)
```

13.2.3.1 カスタマイズ変数

elmo-spam-bsfilter-program
: 初期設定は bsfilter。bsfilter の実行ファイルの名前を設定します。実行ファイルが、環境変数 PATH 上にない場合は、フルパスを設定する必要があります。

elmo-spam-bsfilter-args
: 初期設定は、nil。bsfilter の実行時に実行ファイルに渡される引数を指定します。

elmo-spam-bsfilter-database-directory
: 使用する統計データベースの存在するディレクトリを指定します。nil ならデフォルトの位置 (~/.bsfilter) が使用されます。初期設定は、nil。

elmo-spam-bsfilter-debug
: 初期設定は、nil。non-nil を指定すると、bsfilter からの出力が"*Debug ELMO Bsfilter*"というバッファに出力されます。

elmo-spam-bsfilter-shell-program
: 初期設定は、ruby。bsfilter を起動するシェルの名前を設定します。シェルが環境変数 PATH 上にない場合は、フルパスを設定する必要があります。

elmo-spam-bsfilter-shell-switch
: 初期設定は、nil。bsfilter を起動するシェルに与える引数を指定します。

elmo-spam-bsfilter-update-switch
: 初期設定は、"--auto-update"。メッセージを学習する際に bsfilter に与える引数を指定します。

13.2.4 SpamAssassin

SpamAssassin (http://spamassassin.org/) は、Perl 言語で実装された、テキスト解析技術やブラックリストに基づくメールフィルタで、最もよく使われている spam フィルタの一つです。SpamAssassin は Bayesian フィルタを使用しており、spam と正当なメールについて学習させることで判定の正確性を向上することができます。

SpamAssassin を使用するには、~/.wl などに以下の設定を記述します。(もちろん、あらかじめ SpamAssassin がインストールされ、正常に動作することを確認してください)

```
(setq elmo-spam-scheme 'sa)
```

13.2.4.1 カスタマイズ変数

elmo-spam-spamassassin-program
: 初期設定は spamassassin。spamassassin の実行ファイルの名前を設定します。実行ファイルが、環境変数 PATH 上にない場合は、フルパスを設定する必要があります。

elmo-spam-spamassassin-learn-program
: 初期設定は sa-learn。SpamAssassin において Bayesian フィルタの学習を行なうプログラム、sa-learn の実行ファイルの名前を設定します。実行ファイルが、環境変数 PATH 上にない場合は、フルパスを設定する必要があります。

elmo-spam-spamassassin-program-arguments
: 初期設定は、'("-e")。spamassassin 実行時に与える引数を指定します。spam の判定結果を、プログラムのプロセス終了コードとして出力する引数を与える必要があります。例えば、spamassassin の替わりに spamc を用いる場合、'("-c") を設定しなければなりません。

elmo-spam-spamassassin-learn-program-arguments
: 初期設定は、nil。SpamAssassin の学習用プログラム sa-learn 実行時に与える引数を指定します。

elmo-spamassassin-debug
: 初期設定は、nil。t を指定すると、spamassassin からの出力が "*Debug ELMO SpamAssassin*"というバッファに出力されます。

13.2.5 SpamOracle

SpamOracle (http://pauillac.inria.fr/~xleroy/software.html#spamoracle) は、Objective Caml で実装された spam フィルタです.

SpamOracle を使用するには、~/.wlなどに以下の設定を記述します。(もちろん、あらかじめ SpamOracle がインストールされ、正常に動作することを確認してください)

```
(setq elmo-spam-scheme 'spamoracle)
```

13.2.5.1 カスタマイズ変数

elmo-spam-spamoracle-program
: 初期設定は spamoracle。spamoracle の実行ファイルの名前を設定します。実行ファイルが、環境変数 PATH 上にない場合は、フルパスを設定する必要があります。

elmo-spam-spamoracle-config-filename
: 初期設定は nil。spamoracle の設定ファイルを指定します. nil を指定すると、デフォルトの設定ファイル (~/.spamoracle.conf) が使用されます。

elmo-spam-spamoracle-database-filename
: 初期設定は ~/.elmo/.spamoracle.db。spamoracle で使用するデータベースファイルのパスを指定します。

elmo-spam-spamoracle-spam-header-regexp
: 初期設定は "^X-Spam: yes;"。spam メールであることを示すヘッダの正規表現を指定します。spamoracle の設定ファイル内で、 spam_header パラメータの設定を変更した場合は、この変数の値を設定してください。

13.2.6 Regular Expressions Header Matching

メッセージヘッダの各フィールドが正規表現と合致するかどうかを検査し、spam かどうかを判定します。このバックエンドを使用するには、~/.wl などに以下の設定を記述します。

```
(setq elmo-spam-scheme 'header)
```

overview 情報に含まれないフィールドを検査の対象とする場合、elmo-msgdb-extra-fields に追加しておくと、出来るだけメッセージ本体を読み込まずに overview 情報を元に検査します。

13.2.6.1 カスタマイズ変数

elmo-spam-header-good-alist
: 初期設定は以下の通り。

```
'(("X-Spam-Flag" . "No"))
```

ヘッダフィールド名と合致した時に non-spam と判定する為の正規表現の組をリストで指定します。elmo-spam-header-spam-alist より優先されます。

elmo-spam-header-spam-alist
: 初期設定は以下の通り。

```
'(("X-Spam-Flag" . "Yes"))
```

ヘッダフィールド名と合致した時に spam と判定する為の正規表現の組をリストで指定します。

14 より進んだ使い方

14.1 パッケージのある生活

他のパッケージを使うための設定例です。

14.1.1 imput

util/im-wl.el を load-path において以下のように設定すれば OK です。

```
(autoload 'wl-draft-send-with-imput-async "im-wl")
(setq wl-draft-send-function 'wl-draft-send-with-imput-async)
```

14.1.2 bbdb.el

The Insidious Big Brother Database (http://bbdb.sourceforge.net/) を Wanderlust と共に使うには、 util/bbdb-wl.el を load-path に置いて以下のように設定すれば OK です。

ただし、util/bbdb-wl.el は http://savannah.nongnu.org/projects/bbdb/ で開発されている、BBDB 3.x とは互換性がありません。この場合は、BBDBV3-Wl (https://gna.org/projects/bbdbv3-wl/) が、有用かもしれません。

インストール時に load-path 上に BBDB があれば、bbdb-wl.el はバイトコンパイル/インストールされます。2.3 節 「Install」(4 ページ) を参照してください.

```
(require 'bbdb-wl)

(bbdb-wl-setup)
;; ポップアップ表示
(setq bbdb-use-pop-up t)
;; 自動収集
(setq bbdb/mail-auto-create-p t)
;; 自動収集しないフォルダの指定
(setq bbdb-wl-ignore-folder-regexp "^@")
(setq signature-use-bbdb t)
(setq bbdb-north-american-phone-numbers-p nil)
;; サマリに bbdb の名前を表示 :-)。
(setq wl-summary-from-function 'bbdb-wl-from-func)
;; 自動的に ML フィールドを加える
(add-hook 'bbdb-notice-hook 'bbdb-auto-notes-hook)
(setq bbdb-auto-notes-alist '(("X-ML-Name" (".*$" ML 0))))
```

ドラフトバッファで *M-TAB* により BBDB を用いたアドレスの補完ができます。

14.1.3 lsdb.el

The Lovely Sister Database (http://sourceforge.jp/projects/lsdb/) を Wanderlust と共に使うための設定例を以下に示します。

```
(require 'lsdb)
(lsdb-wl-insinuate)
(add-hook 'wl-draft-mode-hook
          (lambda ()
            (define-key wl-draft-mode-map "\M-\t" 'lsdb-complete-name)))
```

この例では LSDB を用いたアドレスの補完を *M-TAB* に割り当てています。

14.1.4 sc.el(supercite), sc-register.el

普通のメーラと同じ設定で OK です。以下は、設定の一例です。

```
(autoload 'sc-cite-original "supercite" nil t)
(add-hook 'mail-citation-hook 'sc-cite-original)
```

14.1.5 mu-cite.el

普通のメーラと同じ設定で OK です。以下は設定の一例です。

mu-cite 8.0 以前のバージョンをお使いなら、以下のように設定してください。

```
(autoload 'mu-cite/cite-original "mu-cite" nil t)
(setq mail-citation-hook 'mu-cite/cite-original)
```

mu-cite 8.1 以降のバージョンをお使いなら、以下のように設定してください。

```
(autoload 'mu-cite-original "mu-cite" nil t)
(add-hook 'mail-citation-hook (function mu-cite-original))
```

14.1.6 x-face

以下のいずれかをインストールすることで、 メッセージバッファの 'X-Face:' フィールドをデコードして、画像を表示することができます。

また、エンコード済みの X-Face 文字列を `~/.xface` (変数 `wl-x-face-file` の値です) の内容に用意しておくと、ドラフトが準備されるときに自動的に 'X-Face:' フィールドとして挿入されます。(変数 `wl-auto-insert-x-face` が non-nil の場合)

14.1.6.1 x-face-xmas (XEmacs の場合)

x-face (`ftp://jpl.org/pub/elisp/`) 1.3.6.13 以降に付属の `x-face-xmas.el` を使う場合は以下のように設定してください。

```
(autoload 'x-face-xmas-wl-display-x-face "x-face")
(setq wl-highlight-x-face-function 'x-face-xmas-wl-display-x-face)
```

14.1.6.2 x-face-mule (Emacs の場合)

bitmap-mule (`ftp://ftp.jpl.org/pub/elisp/bitmap/`) 8.0 以降に付属の `x-face-mule.el` を使う場合は以下のように設定してください。

```
(autoload 'x-face-decode-message-header "x-face-mule")
(setq wl-highlight-x-face-function 'x-face-decode-message-header)
```

14.1.6.3 x-face-e21 (Emacs 21.x の場合)

Emacs 21.x の場合には、`x-face-mule.el` の代わりに `x-face-e21.el` (`ftp://jpl.org/pub/elisp/`) を使って X-Face を表示することもできます。その場合、bitmap-mule は不要です。以下のように設定してください。

```
(autoload 'x-face-decode-message-header "x-face-e21")
(setq wl-highlight-x-face-function 'x-face-decode-message-header)
```

14.1.7 dired-dd(Dired-DragDrop)

dired-dd パッケージに含まれる dired-dd-mime.el を組み込めば、GNU Emacs で編集中の草稿バッファへ dired からドラッグ&ドロップするだけで簡単にマルチパートを作成できます (もっとも、Wanderlust 専用ではなく SEMI 汎用ですが)。

```
;; dired-dd: http://www.asahi-net.or.jp/~pi9s-nnb/dired-dd-home.html
(add-hook 'dired-load-hook
          (function
           (lambda ()
             (load "dired-x")
             ;; Set dired-x variables here.
             ;; To and flo...
             (if window-system
                 (progn (require 'dired-dd)
                        (require 'dired-dd-mime))))))
```

14.1.8 mhc.el

Message Harmonized Calendaring system (http://www.quickhack.net/mhc/)

MHC を用いると、メッセージを元に予定表を作れます。

mhc-0.25 の場合：

```
(setq mhc-mailer-package 'wl)
(autoload 'mhc-mode "mhc" nil t)
(add-hook 'wl-summary-mode-hook 'mhc-mode)
(add-hook 'wl-folder-mode-hook 'mhc-mode)
```

mhc-current の場合：

```
(autoload 'mhc-wl-setup "mhc-wl")
(add-hook 'wl-init-hook 'mhc-wl-setup)
```

14.1.9 wl-addrbook.el

Addrbook of Mew (http://www.mew.org/)

Mew の Addrbook を Wanderlust で使用できるようにするには、util/wl-addrbook.el と util/wl-complete.el を load-path において、以下のように設定します。

```
(require 'wl-addrbook)
(wl-addrbook-setup)
```

14.1.10 mime-w3m.el

emacs-w3m (http://emacs-w3m.namazu.org/) に付属の mime-w3m.el を使って、メッセージの html パートを表示させることが出来ます。設定方法は mime-w3m.el の先頭のコメントを参照してください。SEMI が SEMI-EPG である場合は、特別な設定は不要です。

14.2 ハイライトの設定

14.2.1 カスタマイズ変数

`wl-summary-highlight`
: 初期設定は `t`。サマリのハイライトを行うかどうか。Non-nil ならサマリのハイライトを行います。

`wl-highlight-max-summary-lines`
: 初期設定は 10000。サマリの行数がこの値より大きい場合、サマリのハイライトを行いません。

`wl-summary-highlight-partial-threshold`
: 初期設定は 1000。サマリ全体をハイライトするかどうかの閾値。この値を越える行数のメッセージがサマリに存在する場合、部分的なハイライトを行います。

`wl-summary-partial-highlight-above-lines`
: 初期設定は 30。`wl-summary-highlight-partial-threshold` を越える行数のメッセージがサマリに存在する場合、バッファ末から、カーソル行よりこの値の行数分だけ上のメッセージまで部分的なハイライトが行なわれます。(この値を `nil` にするとバッファ末から `wl-summary-highlight-partial-threshold` 行分だけハイライトされるようになります。)

`wl-highlight-body-too`
: 初期設定は `t`。Non-nil ならドラフトおよびメッセージの本文もハイライトの対象とします。

`wl-highlight-message-header-alist`
: ドラフトおよびメッセージのヘッダのハイライトを行う際に、重要 (`wl-highlight-message-important-header-contents`) な、二番目に重要 (`wl-highlight-message-important-header-contents2`) な、そして重要ではない (`wl-highlight-message-unimportant-header-contents`) ことを表す face をそれぞれどのメッセージヘッダに割り当てるかを設定します。同様に、任意の正規表現に対して任意の face を割り当てることもできます。

`wl-highlight-citation-prefix-regexp`
: ドラフトおよびメッセージの本文内の引用行を示す正規表現を指定します。この正規表現にマッチした本文は、(`wl-highlight-message-cited-text-*`) で指定される face でハイライトされます。

`wl-highlight-highlight-citation-too`
: 初期設定は `nil`。Non-nil なら `wl-highlight-citation-prefix-regexp`で与えられる引用行の引用を示す正規表現自体もハイライトの対象とします。

`wl-highlight-citation-header-regexp`
: 引用を始めることを示すヘッダの正規表現を指定します。この正規表現にマッチした本文は、`wl-highlight-message-headers`で指定される face でハイライトされます。

`wl-highlight-max-header-size`
: 初期設定は `nil`。メッセージのヘッダサイズがこの値より大きい場合、ヘッダをハイライトしません。`nil` の場合、サイズに関係なくハイライトします。

wl-highlight-max-message-size
: 初期設定は 10000。メッセージがこの値より大きい場合、メッセージのハイライトを行いません。これにより uuencode や非常に大きなダイジェストなどのハイライトの抑止を行います。

wl-highlight-signature-separator
: シグニチャの境界を表す正規表現を指定します。正規表現でも、正規表現のリストでも構いません。この正規表現にマッチした場所以降のメッセージは、wl-highlight-message-signatureで指定される face でハイライトされます。

wl-max-signature-size
: 初期設定は 400。シグニチャをハイライトする場合、ハイライトする最大のシグニチャの大きさを指定します。

wl-use-highlight-mouse-line
: 初期設定は t。Non-nil ならフォルダモード、サマリモードなどでマウスポインタの行をハイライトの対象とします。

14.2.2 文字の色、フォントの設定

文字の色や、フォントを変えるには、Wanderlust で定義されている face を変更する必要があります。フォントを変えたいときは set-face-font、色を変えたいときは set-face-foreground などを使えばよいでしょう。face の設定は .emacs に書くことはできないので ~/.wl に書いてください。

たとえば、シグニチャの色を黄色に変えたいときは、

```
(set-face-foreground 'wl-highlight-message-signature "yellow")
```

を ~/.wl に書きます。

以下に Wanderlust で定義されている face について説明します。

wl-highlight-message-headers
: メッセージヘッダの名前部分の face です。

wl-highlight-message-header-contents
: メッセージヘッダの内容部分の face です。

wl-highlight-message-important-header-contents
: メッセージヘッダの内容のうち重要な部分の face です。デフォルトでは、'Subject:' の内容部分が設定されています。この値は wl-highlight-message-header-alist を変更すれば変えることができます。

wl-highlight-message-important-header-contents2
: メッセージヘッダの内容のうち重要な部分の face その 2 です。デフォルトでは、'From:' と 'To:' の内容部分が設定されています。この値は wl-highlight-message-header-alist を変更すれば変えることができます。

wl-highlight-message-unimportant-header-contents
: メッセージヘッダの内容のうち重要ではない部分の face です。デフォルトでは、'X-' で始まるヘッダと 'User-Agent:' の内容部分が設定されています。この値は wl-highlight-message-header-alist を変更すれば変えることができます。

`wl-highlight-message-citation-header`
: メッセージの引用ヘッダ部分の face です。

`wl-highlight-message-cited-text-*`
: メッセージの引用テキスト部分の face です。最後には**数字**がつき、10 段階まで引用ごとに色分けできるようにしています。

`wl-highlight-message-signature`
: メッセージのシグニチャ部分の face です。初期設定は、明色バックでは ‘`khaki`’、暗色バックでは ‘`DarkSlateBule`’ です。

`wl-highlight-header-separator-face`
: ドラフトのメッセージのヘッダセパレータの face です。

`wl-highlight-summary-important-face`
: サマリで重要マークのついたメッセージ行の face です。

`wl-highlight-summary-new-face`
: サマリで新規マークのついたメッセージ行の face です。

`wl-highlight-summary-displaying-face`
: サマリで現在表示中のメッセージ行の face です。この face は overlay されます。

`wl-highlight-thread-indent-face`
: サマリで現在表示中のメッセージ行の face です。

`wl-highlight-summary-unread-face`
: サマリで未読マークのついたメッセージ行の face です。

`wl-highlight-summary-deleted-face`
: サマリで削除マークのついたメッセージ行の face です。

`wl-highlight-summary-refiled-face`
: サマリでリファイルマークのついたメッセージ行の face です。

`wl-highlight-refile-destination-face`
: サマリでリファイルマークの付いたメッセージ行の、リファイル先情報の部分につく face です。

`wl-highlight-summary-copied-face`
: サマリでコピーマークのついたメッセージ行の face です。

`wl-highlight-summary-target-face`
: サマリでまとめ処理用マーク ‘`*`’ のついたメッセージ行の face です。

`wl-highlight-summary-thread-top-face`
: サマリでスレッドトップのメッセージ行の face です。

`wl-highlight-summary-normal-face`
: サマリでスレッドトップではないメッセージ行の face です。

`wl-highlight-folder-unknown-face`
: フォルダモードで、いくつ未同期メッセージがあるか分らないフォルダの face です。

`wl-highlight-folder-zero-face`
: フォルダモードで、未同期メッセージがないフォルダの face です。

wl-highlight-folder-few-face
: フォルダモードで、未同期メッセージが少しあるフォルダの face です。

wl-highlight-folder-many-face
: フォルダモードで、未同期メッセージがたくさんあるフォルダの face です。「少し」と「たくさん」の切れ目は、変数 wl-folder-many-unsync-threshold で設定されます。

wl-highlight-folder-unread-face
: フォルダモードで、未同期メッセージがなくて未読メッセージがあるフォルダの face です。

wl-highlight-folder-killed-face
: フォルダモードで、アクセスグループ中の削除されたフォルダの face です。

wl-highlight-folder-opened-face
: フォルダモードで、開いたグループにつく face です。変数 wl-highlight-folder-by-numbers が nil か 数 のとき有効です。

wl-highlight-folder-closed-face
: フォルダモードで、閉じたグループにつく face です。変数 wl-highlight-folder-by-numbers が nil か 数 のとき有効です。

wl-highlight-folder-path-face
: フォルダモードで、現在選択中のフォルダまでのパスにつく face です。

wl-highlight-logo-face
: デモでロゴにつく face です。

wl-highlight-demo-face
: デモの文字列 (バージョン番号など) につく face です。

14.3 メールの着信を知らせる

以下のように設定しておくと '%inbox' にメールが届いたときに、モードラインの表示によりメールの着信を知らせてくれます。

```
(setq wl-biff-check-folder-list '("%inbox"))
```

14.3.1 カスタマイズ変数

wl-biff-check-folder-list
: 初期設定は nil。メールの着信をチェックするフォルダのリスト。nil の場合は着信のチェックを行ないません。

wl-biff-check-interval
: 初期設定は 40 (単位:秒)。この値ごとにメール着信のチェックを行ないます。

wl-biff-use-idle-timer
: 初期設定は nil。nil なら wl-biff-check-interval で指定した時間が経過するとメール着信のチェックを行います。Non-nil なら wl-biff-check-interval で指定した時間だけアイドル状態が続くとメール着信のチェックを行います。

wl-biff-notify-hook
: 新しいメールが届いた際に実行されるフック。着信時にビープ音を鳴らす（初期設定）なら以下のように設定します。

```
(setq wl-biff-notify-hook '(ding))
```

音を鳴らしたりしない場合は nil に設定してください。

14.4 パスワードの管理

サーバに接続する為に入力したパスワードは変数 elmo-passwd-alist に接続毎に保持されます。この変数にはエンコードされた生のパスワードが保持されており、Emacs を他人に触られるとパスワードを見られる危険性があるので十分に注意して下さい。

パスワードが保持されている状態で *M-x elmo-passwd-alist-save* を実行するとパスワードがファイルに保存され、パスワードの入力が不要になります。この場合、キー入力を盗み見られる危険性は減るかもしれませんが、生のパスワードがファイルに保存されるので、その扱いには十分注意して下さい。ファイルに保存されたパスワードを消去するには *M-x elmo-passwd-alist-clear* の後、*M-x elmo-passwd-alist-save* を実行して下さい。

elmo-passwd-alist-file-name
: 初期設定は passwd。パスワードをセーブしておくファイルの名前です。コマンド elmo-passwd-alist-save を実行するとこのファイルに現在設定されているパスワードがセーブされます。

elmo-passwd-life-time
: 初期設定は nil。Non-nil な値が設定された場合は、新規にパスワードが入力されてから elmo-passwd-life-time 秒後にパスワードを消去するタイマが、セットされます。nil の場合は入力されたパスワードを消去しません。

14.5 メッセージの振り分け

elmo-split を使うと、変数 elmo-split-folder で指定したフォルダ内のメッセージを特定の規則に従って procmail 風に振り分けることができます。この機能を使うには、まず ~/.emacs に以下のように設定して下さい。

```
(autoload 'elmo-split "elmo-split" "Split messages on the folder." t)
```

振り分け元のフォルダを以下のように設定します。

```
(setq elmo-split-folder "%inbox")
```

振り分けのルールは変数 elmo-split-rule に記述します (書き方は後で説明します)。以上の設定をした上で *M-x elmo-split* すると elmo-split-rule に従って振り分けを実行します。*C-u M-x elmo-split* とすると実際には振り分けはせずにリハーサルを行ない、その結果を表示します。

以下ではルールの記述の仕方を説明します。まずは次の例を見て下さい。

```
(setq elmo-split-rule
      ;; SPAM は '+junk' へ
      '(((or (address-equal from "i.am@spammer")
     (address-equal from "dull-work@dull-boy")
     (address-equal from "death-march@software")
     (address-equal from "ares@aon.at")
     (address-equal from "get-money@richman"))
 "+junk")
;; mule メーリングリストからのメールを '%mule' へ
((equal x-ml-name "mule") "%mule")
;; wanderlust メーリングリストからのメールを '%wanderlust' へ
;; そして続けてそれ以下の規則も評価する。
((equal x-ml-name "wanderlust") "%wanderlust" continue)
;; Yahoo 利用者からのメッセージを '+yahoo-{username}' へ
((match from "\\(.*\\)@yahoo\\.com")
 "+yahoo-\\1")
;; マッチしなかった残りを '+inbox' へ
(t "+inbox")))
```

規則の基本単位は

```
('CONDITION' 'ACTION' [continue])
```

の組で、'CONDITION' が真の場合に 'ACTION' を実行します。第一の要素 'CONDITION' には条件を S 式で記述します。書式についてはすぐ後で説明します。第二の要素 'ACTION' にはメッセージの振り分け先のフォルダ名、もしくはシンボルを指定します。第三の要素 continue をシンボルとして与えると、'CONDITION' が満たされた場合にも振り分け規則の評価を継続します。

'CONDITION' の記法は以下のようになります。実際の書き方は上で挙げた例を参考にして下さい。

1. '**フィールド名**' および '**値**' を引数として取る関数。('**フィールド名**' はフィールド名を表すシンボルです)

   `equal` フィールドの値が '**値**' に等しければ真。大文字小文字の差は無視されます。

   `match` フィールドの値が '**値**' にマッチすれば真。'**値**' は `\&` や `\N` を含むことができます。それらはその前の '**値**' で `\(\)` にマッチしたパターンに置き換えられます。

   `address-equal` そのフィールドにあるアドレスのいずれかが '**値**' に等しければ真。大文字小文字の差は無視されます。

   `address-match` そのフィールドにあるアドレスのいずれかが '**値**' にマッチすれば真。'**値**' は `\&` や `\N` を含むことができます。それらはその前の '**値**' で `\(\)` にマッチしたパターンに置き換えられます。

2. 1 つの整数 ('SIZE') を引数としてとる関数。

   `<` メッセージのサイズが 'SIZE' より小さければ真。

   `>` メッセージのサイズが 'SIZE' より大きければ真。

3. 任意数の引数を取る関数。

   `or` 引数のいずれかが真を返すならば真。

   `and` 引数のすべてが真を返すならば真。

4. シンボル。

   シンボルが指定されると、それを評価します。

‘ACTION’ の値として指定できるのは以下のいずれかです。

1. フォルダ名

   文字列が指定されるとそれを振り分け先のフォルダ名とみなして、そのフォルダへメッセージを追加します。

2. ‘delete’

   シンボル ‘delete’ が指定されると `elmo-split-folder` 内にあるメッセージの実体を削除します。

3. ‘noop’

   シンボル ‘noop’ が指定された場合、そのメッセージに対しては何もせず、そのままの状態に保ちます。

4. 関数

   関数が指定された場合、それを実行します。

全ての振り分け規則を通過したメッセージは、変数 `elmo-split-default-action` で指定した ‘ACTION’ に沿って処理されます。

14.6 バッチ処理

コマンドラインから wanderlust に仕事をさせることができます。現在できる処理は指定したフォルダの新着メッセージのプリフェッチです。

`wl-batch-prefetch-folder-list` にプリフェッチを行うフォルダを指定してコマンドラインから以下のようにするとプリフェッチを行います。

```
% emacs -batch -l wl-batch -f wl-batch-prefetch
```

14.6.1 カスタマイズ変数

`wl-batch-prefetch-folder-list`
: `wl-batch-prefetch` でプリフェッチを行うフォルダを、フォルダ名のリストで指定します。

14.7 カスタマイズ〜応用編〜

14.7.1 返事用ドラフト

サマリモードで `a` を押すと返事用のドラフトが用意されます。用意される草稿の宛先は、以下のようにして設定することができます。

例えば、

```
(setq wl-draft-reply-without-argument-list
      '(("Mail-Followup-To" . (("Mail-Followup-To") nil ("Newsgroups")))
        ("Followup-To" . (nil nil ("Followup-To")))
        (("X-ML-Name" "Reply-To") . (("Reply-To") nil nil))
        ("From" . (("From") ("To" "Cc") ("Newsgroups")))))
```

のように設定します。ここでリスト wl-draft-reply-without-argument-list の各要素は

```
(key . (to-list cc-list newsgroup-list))
```

となっており、'key' で指定したフィールドが存在するときに、親メッセージの 'to-list' で指定されるヘッダを、草稿の 'To:' にコピーします。また同様に、親の 'cc-list', 'newsgroup-list' は草稿の 'Cc:', 'Newsgroups:' にコピーされます。

例を挙げて説明します。

```
("Mail-Followup-To" . (("Mail-Followup-To") nil ("Newsgroups")))
```

親メッセージに 'Mail-Followup-To' フィールドが存在したときにマッチします。親メッセージの 'Mail-Followup-To' および 'Newsgroups' フィールドの内容を、草稿の 'To' および 'Newsgroups' にコピーします。

```
(("X-ML-Name" "Reply-To") . (("Reply-To") nil nil))
```

親メッセージに 'X-ML-Name' および 'Reply-To' の両方が存在したときにマッチします。親メッセージの 'Reply-To' を草稿の 'To' にコピーします。

```
("From" . (("From") ("To" "Cc") ("Newsgroups")))
```

親メッセージの 'From' を草稿の 'To' に、親の 'To' および 'Cc' を草稿の 'Cc' に、'Newsgroups' を 'Newsgroups' にそれぞれコピーします。

これらは順番に評価され、最初にマッチしたものが使われます。

同様にして、prefix argument 付きで a を押したときの動作が、wl-draft-reply-with-argument-list で設定できます。

また、関数 (親メッセージのバッファで評価される) を 'key' や 'to-list' 等の変わりに用いることもできます。

自分の書いたメッセージへの返信の場合にマッチさせたい場合には 'key' として関数 wl-draft-self-reply-p を指定します。

C-u a したときに、wl-subscribed-mailing-list のアドレスが含まれるメールに対してはメーリングリストにのみ返信したい場合、次のような設定ができます。

```
(defun wl-mailing-list-addresses ()
  (let (list-addrs)
    (dolist (to (mapcar
 (lambda (addr)
   (nth 1 (std11-extract-address-components addr)))
 (wl-parse-addresses
  (wl-concat-list
   (elmo-multiple-fields-body-list (list "To" "Cc"))
   ","))))
      (when (elmo-string-matched-member to wl-subscribed-mailing-list t)
(setq list-addrs (cons to list-addrs))))
    (nreverse list-addrs)))

(setq wl-draft-reply-with-argument-list
      '((wl-mailing-list-addresses . (wl-mailing-list-addresses nil nil))
("Reply-To" . (("Reply-To") nil nil))
("Mail-Reply-To" . (("Mail-Reply-To") nil nil))
("From" . (("From") nil nil))))
```

14.7.2 スレッドの見ため

```
  389  09/18(金)01:07 [ てらにし          ] wl-0.6.3
  390  09/18(金)07:25 +-[ 津邑さん          ]
  391  09/18(金)19:24 +-[ 村田さん          ]
  392  09/20(日)21:49 +-[ 奥西さん          ]
  396  09/20(日)22:11 | +-[ 津邑さん          ]
  398  09/21(月)00:17 |   +-[ 津邑さん          ]
  408  09/21(月)22:37 |     +-[ 奥西さん          ]
  411  09/22(火)01:34 |       +-[ 津邑さん          ]
  412  09/22(火)09:28 |       +-[ てらにし          ]
  415  09/22(火)11:52 |         +-[ 津邑さん          ]
  416  09/22(火)12:38 |           +-[ てらにし          ]
  395  09/20(日)21:49 +-[ 奥西さん          ]
  397  09/21(月)00:15 +-[ 奥西さん          ]
```

スレッドの見ためを上記のようにしたい場合の設定は以下の通りです。

```
(setq wl-thread-indent-level 2)
(setq wl-thread-have-younger-brother-str "+")
(setq wl-thread-youngest-child-str       "+")
(setq wl-thread-vertical-str             "|")
(setq wl-thread-horizontal-str           "-")
(setq wl-thread-space-str                " ")
```

枝を表示しないようにしたい場合の設定は以下のようにします。

```
(setq wl-thread-indent-level 2)
(setq wl-thread-have-younger-brother-str " ")
(setq wl-thread-youngest-child-str       " ")
(setq wl-thread-vertical-str             " ")
(setq wl-thread-horizontal-str           " ")
(setq wl-thread-space-str                " ")
```

14.7.3 User-Agent フィールド

'X-Mailer:' フィールドや 'User-Agent:' フィールドに凝りたいという変わった人は、文字列を生成する関数を自分の好きなように定義して、変数 wl-generate-mailer-string-function に設定してください。

'User-Agent:' フィールドを短くしたいのであれば、以下の設定をしてください。

```
(setq wl-generate-mailer-string-function
      'wl-generate-user-agent-string-1)
```

以下は設定の例です。

```
(setq wl-generate-mailer-string-function nil)
(setq wl-draft-additional-header-alist
      (list
       (cons 'X-Mailer (lambda () (product-string-1 'wl-version)))))
```

14.8 その他のカスタマイズ変数一覧

その他のカスタマイズ変数一覧。

wl-default-folder
: 初期設定は '%inbox'。フォルダ移動時などのデフォルト値となります。

wl-draft-folder
: 初期設定は '+draft'。ドラフトをセーブするフォルダです。書き込みのできるフォルダを指定する必要があります。IMAP のリモートフォルダや、Maildir なども指定可能です。なお、wl-draft-config-exec で設定された変数は elmo-msgdb-directory 以下に保存されます。したがって、wl-draft-folder でリモートのフォルダを指定した場合、ドラフトのセーブ前に wl-draft-config-exec で設定された変数は別のマシンで wl-summary-reedit で再編集した時には効果を持ちません。

wl-trash-folder
: 初期設定は '+trash'。ゴミ箱フォルダです。この値を変更したときは Wanderlust を再起動することをお勧めします。

wl-interactive-exit
: 初期設定は t。Non-nil ならば、Wanderlust 終了時に確認を行います。

wl-interactive-send
: 初期設定は t。Non-nil ならば、メール送信時に本当に送信して良いかを確認します。

wl-default-sync-range
: 初期設定は 'update'。デフォルトのサマリ更新レンジで、'all', 'update', 'rescan', 'no-sync' のいずれかを指定します。レンジの意味については wl-summary-sync の説明を参照して下さい。

wl-folder-sync-range-alist
: 初期設定は、以下の連想リスト。

```
(("^&.*$" . "all")
 ("^\\+draft$\\|^\\+queue$" . "all"))
```

フォルダ名の正規表現とサマリ更新レンジとの連想リストです。更新レンジには 'all', 'update', 'rescan', 'no-sync' のいずれかを指定します。マッチしなかった場合は、wl-default-sync-range の値 (初期値は 'update') を使います。レンジの意味については wl-summary-sync の説明を参照して下さい。

wl-ask-range
: 初期設定は t。nil なら、フォルダ移動時のサマリ更新で wl-folder-sync-range-alist の値を使用します。

wl-mime-charset
: 初期設定は x-ctext。MIME ではないメッセージの場合 ('Content-Type:' がないメールなど) や、サマリの表示で用いられる MIME charset です。(Nemacs とその他の Emacsen でサマリを共有したい場合は、この値を iso-2022-jp としてください。)

wl-highlight-folder-with-icon
: XEmacs または Emacs 21 で有効です。初期設定はその Emacs に依存します (アイコンを使用できる Emacsen では t になります)。

wl-strict-diff-folders
: フォルダ名の正規表現のリストです。フォルダモードで *s* を押すなどして未読のメッセージ数をチェックした場合、通常は簡易的な方法でチェックしています (処理は速いが、正確ではない)。この変数にマッチするフォルダは厳密にチェックします。IMAP4 フォルダに対する条件フィルタフォルダのような場合には、この変数にマッチするよう設定すると良いでしょう。初期設定は nil。

wl-folder-use-server-diff
: フォルダモードで *s* を押すなどして未読のメッセージ数をチェックした場合、通常は (サーバ上のメッセージ数) − (ローカルにあるメッセージ数) が未読とみなされます。しかし、この変数が non-nil ならば、サーバ上の未読のメッセージ数をチェックします。IMAP4 フォルダにのみ影響があります。ただし、変数 elmo-imap4-disuse-server-flag-mailbox-regexp にマッチするメールボックスの IMAP4 フォルダは、この変数にマッチしてもサーバ上の未読のメッセージ数をチェックしません。初期設定は t。

wl-auto-check-folder-name
: 初期設定は nil。起動時に未読数をチェックするフォルダやグループを指定します。チェックを行ないたいフォルダ (グループ) のリストを指定することもできます。nil ならば起動時に Desktop 全体をチェックします。none ならば、起動時に何もチェックしません。

wl-auto-uncheck-folder-list
: 初期設定は以下のリスト。

```
("\\$.*")
```

wl-auto-check-folder-name で指定されたグループに含まれていても起動時に未読チェックしないフォルダ名の正規表現のリストです。また、Desktop に対して未読チェックをする際にも、これらのフォルダはスキップされます。

wl-auto-check-folder-list
: 初期設定は nil。wl-auto-uncheck-folder-list に指定されていたとしても例外的に未読チェックしたいフォルダ名の正規表現のリストを指定します。

wl-no-save-folder-list
: 初期設定は以下のリスト。

```
("^/.*$")
```

セーブしないフォルダ名の正規表現のリストです。

wl-save-folder-list
: 初期設定は nil。セーブするフォルダ名の正規表現のリストです。wl-no-save-folder-list よりも優先します。

wl-folder-mime-charset-alist
: 初期設定は以下の連想リスト。

```
(("^-alt\\.chinese" . big5)
 ("^-relcom\\." . koi8-r)
 ("^-tw\\." . big5)
 ("^-han\\." . euc-kr))
```

フォルダ名の正規表現と MIME charset の連想リストです。マッチしなかった場合は wl-mime-charset が使われます。

wl-folder-init-load-access-folders
: 初期設定は nil。初期化時に特定のアクセスグループのみロードする場合に、そのグループのリストを指定します。nil の場合は wl-folder-init-no-load-access-folders が参照されます。

wl-folder-init-no-load-access-folders
: 初期設定は nil。初期化時に特定のアクセスグループを除いてロードする場合に、ロードしないアクセスグループのリストを指定します。wl-folder-init-load-access-folders が non-nil の場合は無視されます。

wl-dispose-folder-alist
: 初期設定は以下の連想リスト。

```
(("^-" . remove)
 ("^@" . remove))
```

処分マーク 'd' をつけたメッセージを処分する方針を設定します。リストの各要素はフォルダと処分先になっており、処分先には次のものが指定できます。

```
remove or null : メッセージを即削除する。
string         : 指定したフォルダに移動する。
trash or その他  : wl-trash-folder に移動する。
```

wl-x-face-file
: 初期設定は ~/.xface。エンコード済みの X-Face 文字列を内容とするファイル名です。14.1.6.2 節 「x-face-mule」(97 ページ) を参照してください.

wl-demo-display-logo
: Non-nil ならばオープニングデモでビットマップのイメージを表示します。xpm, xbm を指定すると (可能ならば)、その画像タイプのイメージを表示します。

`elmo-use-database`
: XEmacs のみ有効です。初期設定は、XEmacs に依存します。(dbm の機能をもつ XEmacs ならば `t` となります。) Non-nil ならば dbm を使って Message-ID の管理を行ないます。

`elmo-nntp-list-folders-use-cache`
: 初期設定は 600 (秒)。NNTP において '`list`' や '`list active`' の結果をキャッシュしておく時間を秒単位で指定します。`nil` ならキャッシュしません。

`elmo-nntp-max-number-precedes-list-active`
: 初期設定は `nil`。NNTP において '`list active`' の結果得られる記事番号をフォルダの最大記事番号として利用します。NNTP サーバとして INN 2.3 などを使用していて、フォルダモードでの既読数が一致しない場合は `t` にしてください。

`elmo-nntp-default-use-listgroup`
: 初期設定は `t`。Non-nil なら、NNTP において 総記事数を調べるために '`listgroup`' を使います。`nil` ならば '`group`' の結果を用います。'`group`' を使うと、正確さには欠けますが、若干高速化されます。

`elmo-pop3-send-command-synchronously`
: 初期設定は `nil`。Non-nil なら POP3 のコマンドを同期的に発行します。サーバによってはこの値を設定しないとサマリ情報を取り出せない場合があるようです。POP3 参照時に処理がハングするような場合は、`t` にすると良いかもしれません。

`elmo-dop-flush-confirm`
: 初期設定は `t`。Non-nil ならばオフライン処理で溜った処理を実行するかどうかを確認します。

`elmo-network-session-idle-timeout`
: 初期設定は `nil`。ネットワークセッションをキャッシュするアイドル時間を秒で指定します。ここで指定された値よりも長い時間、なにも通信が行なわれなかった場合、そのセッションは再利用されません。nil なら、無条件に再利用します。

14.9 フック

(Not yet written)

15 旧バージョンからの移行

この章では、以前のバージョンから移行する場合に必要な設定の変更や、制限等について説明します。

15.1 2.12.0 より前のバージョンからの移行

15.1.1 msgdb の変換について

2.12.0 より前のバージョンから、msgdb の構造が変更されています。以後、新しく作成したフォルダの msgdb は、この新しい形式で作成/保存されます。以前の msgdb の形式をそのまま使い続ける場合には、`~/.wl` 等に以下の設定を記述します。

```
(setq elmo-msgdb-default-type 'legacy)
```

以前のバージョンで作成された msgdb は、デフォルトの設定ではロードされた時に自動的に新しい形式の msgdb に変換されます。自動的に msgdb の形式を変換しないようにするには、以下のいずれかの設定を `~/.wl` 等に記述します。

```
;; msgdb をロードした時に elmo-msgdb-default-type と
;; 異なった形式だった場合に自動的に変換する場合 (デフォルト)
(setq elmo-msgdb-convert-type 'auto)

;; サマリで s all した時に変換する場合
(setq elmo-msgdb-convert-type 'sync)

;; 変換しない場合
(setq elmo-msgdb-convert-type nil)
```

上述の様に以前の msgdb のまま使い続ける事も出来ますが、その場合には以下の制限があります。

1. 転送済みマーク (‘F’, ‘f’) が使用出来ません。
2. ‘important’ 以外のグローバルフラグが使用出来ません。

15.1.2 ‘’mark’ フォルダから ‘’flag’ フォルダへの変更

新しいバージョンの Wanderlust を起動した時に ‘’mark’ フォルダにあったメッセージは、‘’flag’ フォルダに自動的に引き継がれます。但し、こうやって引き継いだメッセージには、以下の制限があります。

1. ‘’flag’ フォルダでメッセージを消しても元のメッセージの ‘important’ フラグは消えません。
2. 元のメッセージの ‘important’ フラグを消しても ‘’flag’ フォルダからは削除されません。
3. help-echo で、元のメッセージが表示されません。

もしも、自動での引き継ぎが上手くいかなかった場合は、*M-x elmo-global-mark-upgrade* を実行する事で、再度 ‘’mark’ フォルダから ‘’flag’ フォルダにメッセージを取り込む事が出来ます。(重複したメッセージは取り込まないので、何回実行しても問題ありません)

16 用語の解説

このマニュアルで使われている用語についての説明をします。

‘フォルダ (folder)’
: メッセージが格納されているいれものです。

‘グループ (group)’
: 複数のフォルダをまとめた集合です。

‘アクセスグループ (access group)’
: 指定したパス以下のフォルダを自動的に集めてできる、特別なグループを指します。2.5 節 「Folder Definition」(7 ページ) を参照してください.

‘サマリ (summary buffer)’
: メッセージの一覧を表示するバッファです。

‘スティッキーサマリ (sticky summary)’
: サマリを抜けると、通常のサマリは破棄されますが、*q* や *g* でサマリを抜けても破棄されずに残る、特別なサマリを指します。5.5 節 「Sticky Summary」(37 ページ) を参照してください.

‘エクスパイア (expire)’
: 期限切れメッセージを自動的に削除したりアーカイブしたりすること。9.1 節 「Expire」(71 ページ) を参照してください.

‘スコア (score)’
: 第 10 章 「Scoring」(79 ページ) を参照してください.

‘プリフェッチ (prefetch)’
: オフラインでメッセージを読むために、あらかじめメッセージをキャッシュしておくことをいいます。

17 メーリングリスト

Wanderlust に関する議論は以下のメーリングリストで行われます。最新バージョンのアナウンスもこちらに流れます。

Wanderlust Mailing List <wl@ml.gentei.org>

ここでは主に日本語での議論が行われています。また、英語専用のリストとして

Wanderlust List in English <wl-en@ml.gentei.org>

もあります (こちらに投稿されたメッセージは前者にも配送されます)。

これらのメーリングリストのガイドを得るには、wl-ctl@ml.gentei.org 宛 (英語の方は wl-en-ctl@ml.gentei.org 宛) で、本文に

```
# guide
```

と書いたメールを送って下さい。

バグ報告やパッチの送付もこれらのメーリングリストへ送ってください。メーリングリストのメンバでなくとも、送信できる設定になっています。

また、バグ報告の場合はバックトレースを取って添付すると原因究明しやすくなります。[1]

メーリングリストの皆様には貴重な御助言、コードをたくさん御提供いただいております。この場を借りてお礼申し上げます。

17.1 アーカイブ

メーリングリストに投稿されたメッセージは、ウェブや NetNews でも読むことができます。

<wl@ml.gentei.org> に投稿されたメッセージ

```
http://dir.gmane.org/gmane.mail.wanderlust.general.japanese
news://news.gmane.org/gmane.mail.wanderlust.general.japanese
```

<wl-en@ml.gentei.org> に投稿されたメッセージ

```
http://dir.gmane.org/gmane.mail.wanderlust.general
news://news.gmane.org/gmane.mail.wanderlust.general
```

[1] バックトレースの取り方は http://www.jpl.org/elips/BUGS-ja.htmlが参考になります。

18 おまけ

18.1 略歴

1998 3/05 MH メッセージをスレッド表示するプロトタイプを作ってみる。
3/10 elisp による msgdb のしくみをつくる。
3/26 IMAP と NNTP もスレッド表示できるようになる。
4/13 スレッド表示用モジュールを elmo としてまとめはじめる。
5/01 0.1.0 ボロボロの initial version が完成。
6/12 tm-ja ML で、IMAP 対応の elisp メーラを作っている、とつい口を滑べらせてしまう。
6/16 tm-ja, elips ML で 0.1.3 をアナウンス。
6/22 北目さんのおかげで northeye.org でメーリングリストがスタート。
7/01 mm-backend 対応 (0.3.0)。
8/25 multi フォルダ追加 (0.5.0)。
8/28 filter フォルダ追加 (0.5.1)。
9/10 スレッドが開閉できるようになる (0.6.0)。
9/11 fldmgr by 村田さん によりフォルダの編集が簡単に。
9/18 lha フォルダ追加 by 奥西さん (0.6.3)。
9/24 スレッドの枝を表示 (0.6.5)。
9/28 圧縮フォルダがマルチアーカイバに対応 by 奥西さん。
10/28 オフライン処理 (0.7.4)。
12/09 ベータバージョンに。
12/21 wl-expire by 村田さん。
1999 2/03 auto-refile by 津邑さん。
4/28 wl-template by 村田さん。
5/18 1.0.0 stable リリース。
7/05 スコア機能 by 村田さん (2.1.0)。
9/26 プラグ管理システム by 村田さん (2.2.2)。
12/20 Modified UTF7 対応。
2000 3/24 1.1.0 stable リリース。
4/03 CVS サーバでの開発を開始。
5/07 スレッドつなぎ直し機能＆高速化 with 村田さん。
6/12 LDAP 対応 with 千葉さん＆後藤さん。
7/11 killed message 機能追加。
7/18 POP3 を UIDL 対応。
9/12 biff 機能 with 嵯峨田さん＆山岡さん。
10/17 expire-hide by 岡田さん。
11/08 2.4.0 stable リリース。
2001 7/04 2.6.0 stable リリース。
8/21 wl-addrmgr by 北本さん。
12/27 2.8.1 stable リリース。
2002 12/11 2.10.0 stable リリース。
2003 7/05 2.10.1 stable リリース。
9/18 flag フォルダ。

	9/20	新形式 msgdb（modb-standard）by 村田浩也さん。
	10/20	スパムフィルタ by 村田浩也さん。
2004	1/06	デモの背景色指定。
	2/09	'file' フォルダ。
	9/12	forwarded マーク。
		マーク文字列のデフォルト値変更。
	12/24	2.12.0 stable リリース。

詳しい変遷は ChangeLog を御覧ください。

18.2 名前

Wanderlust には、研究社英和辞典によれば、

> wanderlust　wan・der・lust
> ‡ドイツ語 'desire to wander' の意から‡
> – 名 放浪癖， 旅行熱， 旅心： have 〜 放浪癖がある.

という意味があります。が、名前にたいして深い意図はありません。(強いて言えば、IMAP ⇒ どこでもメールが読める ⇒ 放浪癖 ?)

elmo は、'Elisp Library for Message Orchestration' の略です。最初はあの赤いぬいぐるみのつもりでしたが、放浪 ⇒ 漂流 ⇒ 道標 ⇒ St. Elmo's fire ⇒ elmo という、それっぽい連想も可能です。

18.3 コードネーム

各バージョンにはコードネームがついています (ほとんど冗談です)。いまのところ 1980 年代の米ビルボード誌トップ 40 ヒット

(http://ntl.matrix.com.br/pfilho/html/top40/)

からアルファベット順に適当に好きなものをピックアップして使っています。

索引

概念索引

D

E

F

G

I

K

L

M

N

O

P

Q

R

S

T

U

W

X

Z

キーバインド索引

!

#

$

*

+

-

.

/

<

>

?

@

[

]

^

|

~

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

変数索引

E

M

W

関数索引

C

W

簡単な目次

目次