special

Mutt にしよう！

世界的定番メーラで始めるLinuxメール環境構築入門

Linuxでメールを読んでますか？　最近はLinuxでもいろいろなメーラを選択できるようになりましたが、世界で広く使われている、最もLinuxらしいメーラと言えば、それは「Mutt」なのですよ。これまで日本では、あまり紹介されることがありませんでしたが、この記事を機に、あなたも、Muttの軽快で奥深い世界に足を踏み入れてみませんか？

滝澤隆史
taki@cyber.email.ne.jp

Muttの概要

皆さんは「Mutt」というMUA*1をご存じでしょうか？　非常に軽快で高機能な割には、日本国内では知名度がイマイチで利用者も少ないMUAです。知名度が低い原因には、Muttに関する日本語の情報が少ないことが挙げられます。Muttに関する日本語のWebサイトも片手の指で数えられるくらいしかありませんし、UNIX系の雑誌でMuttに関する記事もほとんどありません*2。

そこで本記事では、「Muttとは何ぞや」から始まり、インストール、設定、メールの送受信が出来るまでの説明を行います。この記事を読んでMuttについて興味を持っていただけたら幸いです。

Muttとは？

Muttは、Michael R. Elkins氏によって開発されたUNIX系OSの端末で動く、CUIなMUAです（記事末のRESOURCE[1]を参照）。ユーザーインターフェイスや実装している機能は、「ELM」、「pine」、「Mush」、「slrn」といった他のMUAやニューズリーダーを参考にして作られており、文字通り、「雑種犬（mutt）」のようなメーラです（画面1）。

●サポートしている機能

プロトコルに関しては、以下のものサポートしています。他のメーラでも対応しているプロトコルは、ほとんど実装していますが、送信に関しては、外部プログラムを使用するようになっています。ただし、一部のプロトコル*3は、バージョン1.3以降だけで実装されています。

- **MIME（RFC2231も含む）**
- **POP3、IMAP4 Rev.1**
- **POP/IMAP over SSL**

画面1　CUIながら、豊富なカラー対応など、見た目は派手（にもできる）

＊1　「Mail User Agent」あるいは「Message User Agent」の略。ユーザーによるメールの読み書きを補助するためのプログラム。メールリーダやメーラとも呼ばれる。

＊2　筆者の知る限り、これまで1つしか見たことはない。

画面2　インデックス一覧画面では、文字罫線でスレッドを表現

- POP/IMAP over ssh
- APOP、SASLによる認証
- DSN（Delivery Status Notification）、PGP/MIME
- mbox、MMDF、MH、Maildir形式のメールボックス

表示に関しては以下のような特徴を持っています（画面2、画面3、画面4）。

- カラフルな表示
- スレッド表示（slrnライク）
- 未読、既読、返信、暗号、電子署名等のフラグの表示
- 視覚的に分かりやすいマルチパート構造の表示

また、操作に関しては以下のような特徴を持っています。

- メーリングリストのための便利な機能
- メッセージ作成時にヘッダを自由に書くことが可能
- キーバインドやマクロなど豊富なカスタマイズが可能
- フォルダごとに設定を変更可能
- 非常に強力なパターンマッチング
- 複数のメッセージに対しての返信や転送が可能

さらに、Mutt自身が持っていない機能（エディタや送信の機能など）は外部プログラムを用いることになります。

●安定版と開発版およびその日本語パッチ

Muttは、安定版と開発版の2つのバージョンが公開されています。現在は、安定版が「バージョン1.2」、開発版が「バージョン1.3」となっています。バージョン1.2までは、マルチバイト文字をサポートしていなかったため、そのままでは日本語が扱えず、吉田行範氏を中心として開発された日本語パッチが必要でした。一方、バージョン1.3は、XPG5*4の国際化機能が実装され、基本的には日本語を扱えるようになりました。しかし、日本語特有の事情（いわゆる「ヘッダの生JIS問題」や「機種依存文字の文字化け」など）があるため、そのままでは実用上、問題があります。そこで、筆者が中心となって、実際に日本語を扱う上で問題となる点を修正し、ほぼ通常の利用には差し支えないようにした日本語パッチを開発しています。

バージョン1.2とバージョン1.3の日本語パッチは全く別の実装なので、バージョン1.2以前のものからバージョン1.3に移行する場合は注意が必要です（Column「旧バージョンからの移行」を参照）。マニュアルの邦訳と日本語のメッセージカタログ*5は、共にあります。なお、執筆時点での最新バージョンは、安定版が「1.2.5i」、開発版が「1.3.17i」です。なお、バージョン番号の最後に「i」が付いていますが、これは、以前のPGPの輸出制限のせいで、US版と国際版が別々となっていた頃の名残です。

本記事では原則としてバージョン1.3を中心に話を進めます。

画面3　MIMEによる添付ファイルも直感的に操作できます

画面4　メール送信画面では各種ヘッダを視覚的に確認できます

＊3　APOPは、バージョン1.3の途中から実装されているが、バージョン1.2では日本語パッチにより実装されている。
＊4　「X/Open Portability Guide, Issue 5」のこと。UNIXの国際化機能の標準仕様が定義されている。
＊5　これらは、バージョン1.3.16iから本家に取り込まれた。

インストール

M ライブラリ

インストールに必要なソフトウェアは、筆者が作成している「Mutt Japanese Edition」([2])のサイトから、すべて取得できます。今月号の付録CD-ROMにも、一通りのファイルを収録してあります。

なお以降では、シェルとしてbashを使っていることを前提として説明を行いますので、他のシェルをお使いの方は、適宜そのシェルの仕様に読み替えてください。

●コンパイルに必要なライブラリ

Muttで日本語を扱うためには、日本語の表示が可能なcursesライブラリと、国際化関数(iconvとワイド文字関数)を実装したCライブラリが必要です。cursesライブラリには「ncurses」と「S-Lang」がよく使われていますが、S-Langをお勧めします*6。S-Langとiconvに関しては、後で解説します。

「POP/IMAP over TLS/SSL」を使いたい場合や、IMAPでSASL認証を使いたい場合、あるいはPGPを使いたい場合は、それぞれ「OpenSSL」、「Cyrus SASL Library」、「PGP」または「GnuPG」を、あらかじめインストールしておいてください。これらはすべて、Ring Server([3])からダウンロードできます。

●S-Lang

まず、注意点を先に書いておきますが、オリジナルのS-Langのデフォルトの設定のままでは、日本語の表示の一部が文字化けすることがあります。そのため、ソースコードの「src/sl-feat.h」において「SLANG_HAS_KANJI_SUPPORT」を「1」に定義する必要があります。ただし、jpパッチが当たったS-Lang(以降「slang-j」と略、付録CD-ROMに収録)では、この定義がデフォルトで行なわれているため、何もする必要はありません。

では本題に入ります。まず、お使いのシステムのライブラリにS-Langがすでにインストールされているかどうか確認してください。例えば、

```
$ /sbin/ldconfig -p | grep slang
```

を実行しとき、S-Langがインストールされていれば次のような結果が返ってきます。

```
libslang.so.1 (libc6) => /usr/lib/libslang.so.1
libslang.so (libc6) => /usr/lib/libslang.so
```

COLUMN 1

旧バージョンからの移行

Muttの1.2系列以前の日本語パッチと1.3系列の日本語パッチの実装は全く別であり、また、国際化関数の導入もあるため、1.2系列以前からアップデイトするときはいくつか注意が必要です。以下に注意点をあげますので、参考にしてください。

古いMuttrc(サイト共通の設定ファイル)があると新しいMuttrcがインストールされないので、予め、名前を変えて待避した状態でインストールしてください。

$recordで指定したフォルダにある作成済みの文書は作成した文書の文字符号化方式のままでした。つまり、エディタの設定をEUC-JPにして文書を作成した場合は、EUC-JPのまま$recordフォルダに保存されていたということです。1.3系列では送信用の文字符号化方式(ISO-2022-JP)に変換されたもの(相手に届くものと同じもの)が保存されます。既存のメイルを適当なスクリプトを使ってISO-2022-JPに変換してください。

文字符号化方式の指定の方法が変わっているため、以前に設定した$charsetはそのままでは使ってはいけません(恐らく"ISO-2022-JP"になっているケースが多いはず)。トラブルの元になるので設定し直す必要があります。

POP3の認証でAPOPを使うための日本語パッチの独自実装である$pop_apopはなくなりました。その代わり、別実装により本家の方でSASL/APOPの認証が加わりました。自動認識で行われるため、設定項目はありません。

日本語パッチの独自実装である$kanjithreadは$tree_charsに置き換わりました。関連するいくつかの変数$tree_*により使用する文字を任意に設定できます。

$numbered_mlは$delete_prefixに名前が変わり、さらに、$delete_regexpにより正規表現で指定することができるようになりました。

これが最大の注意点かもしれませんが、1.3系列では国際化関数などのオーバーヘッドにより1.2系列より動作が遅くなっています。それでもGUIなMUAよりは軽いとは思いますが、パワー不足のマシンで動かす場合は覚悟してください。

(滝澤隆史)

*6 ncursesでも日本語の表示はできますが、時々、一部分の文字の表示が化けることがあります。

S-Langがすでにインストールされている場合は、次の項目へ移ってもかまいませんが、この情報だけでは先に述べた、日本語に対応するための定義が行なわれているどうかは判断できません。そのため、実際に使ってみて一部の文字が化けるようであれば、新規にインストールを行なってみてください。もちろん確実を期すために、無条件にインストールを行なっても構いません。

なお、includeディレクトリ内にあるslcurses.hに「KANJI」という文字が含まれていれば、jpパッチが当たったものなのでそのまま使えます。また、システムによってはslang-jの共有ライブラリ名が「libslang-ja.so」となっている場合があります。この場合は、「Muttのインストール」の項でも説明しますが、注意が必要です。

システムにS-Langがインストールされていない場合は、新規にインストールする必要があります。ここではslang-jをインストールすること前提に説明します。オリジナルのS-Langをインストールする場合は、先の注意点で指摘した定義を行なっておいてください。

もし、お使いのパッケージシステム(RPMやdebなど)においてslang-jのパッケージがあれば、それを追加するのが簡単です。パッケージがない場合、あるいはオリジナルのS-Langがすでにインストールされていて、パッケージの依存関係で削除や更新ができない場合は、次のようにしてコンパイルおよびインストールを行ってください。なお、slang-jをシステムにインストールせずに、静的リンクにすることもできます。

まず、slang-1.4.2jp0.tar.gz(付録CD-ROMに収録)を展開して、configure を実行します。

```
$ gzip -dc slang-1.4.2jp0.tar.gz | tar xvf -
$ cd slang-1.4.2jp0
$ ./configure --prefix=/usr/local
```

もし静的にリンクする場合は、以下のように、単にmakeするだけでOKです。

```
$ make
$ make runtests
```

ELF形式の共有ライブラリとしてインストールする場合は、次のようにしてインストールを行います。

```
$ make elf
# make install-elf
# make install-links
```

この場合は、/etc/ld.so.confに「/usr/local/lib」という記述が必要です。そして、インストール後に、rootで

```
# ldconfig
```

などと実行して、共有ライブラリのキャッシュの更新を行います。また、オリジナルのS-Langが/usr/libなどにインストールされている場合は、インストールしたslang-jが優先して使われるようにするために、次のようにライブラリの先読みを行う環境変数の設定を行ってください。

```
$ LD_LIBRARY_PATH=/usr/local/lib
$ export LD_LIBRARY_PATH
```

●iconv

iconvとは、文字符号化方式の変換を行うライブラリです。XPG5に準拠したシステムで実装されています。しかし、日本語の文字符号化方式とUTF-8への対応も含めた「まとも」な実装は多くなく、GNU/Linux(glibc-2.2)、Solaris 2.7、AIX 4.3以上であれば使えるという投稿をMuttの開発メーリングリストで読んだことがあります。iconvが実装されていない、あるいは実装が不完全な環境の場合は、Bruno Haible氏のlibiconvパッケージをインストールしてください。

iconvの実装に関して、Linuxディストリビューションで使われているCライブラリの状況は、次の3通りに分けることができます。

- libc5以前、glibc-2.0 …… iconvの実装なし。
- glibc-2.1 …… iconvの実装あり。しかし、不完全。
- glibc-2.2 …… iconvの実装あり。

そのため、glibc-2.2でない場合は、libiconvをインストールする必要があります。

以下に、libiconvのインストールの手順を説明します。libiconv-1.6.tar.gz(付録CD-ROMに収録)を展開し、以下の手順でコンパイルしてインストールします。

```
$ gzip -dc libiconv-1.6.tar.gz | tar xvf -
$ cd libiconv-1.6
$ ./configure
$ make
# make install
```

/etc/ld.so.confに/usr/local/libが記述されていることを確認してください。もしなければ追加します。次に、rootで

```
# ldconfig
```

と実行して共有ライブラリのキャッシュの更新などを行います。その後、環境変数を設定します。これは、ログインシェ

ルの初期化ファイル(.bash_profileなど)に追加しておくと良いでしょう。

```
$ LD_PRELOAD=/usr/local/lib/libiconv_plug.so
$ export LD_PRELOAD
```

なお、システムがiconvを実装していない場合は、次のような環境変数を設定してください。

```
$ LD_LIBRARY_PATH=/usr/local/lib
$ export LD_LIBRARY_PATH
```

Mutt本体

最初に、Muttのソース「mutt-1.3.17i.tar.gz」と日本語パッチ「mutt-1.3.17i-ja0.tar.gz」をそれぞれを展開します(いずれも付録CD-ROMに収録)。

```
$ gzip -dc mutt-1.3.17i.tar.gz | tar xvf -
$ gzip -dc mutt-1.3.17i-ja0.tar.gz | tar xvf -
```

続いて、パッチを当てます。

```
$ cd mutt-1.3.17
$ patch -p1 < ../mutt-1.3.17i-ja0/mutt-1.3.17i-
ja0.diff
```

準備ができたら、configureスクリプトの実行です。ただし、ここで注意する点がいくつかあります。

slang-jが共有ライブラリとしてシステムにインストールされている場合は、オプションとして「--with-slang」を指定してください。オリジナルのS-Langがすでにインストールされていて、slang-jを後から/usr/localにインストールした場合は「--with-slang=/usr/local」を指定します。単にslang-jをmakeしただけで静的リンクする場合は「--with-slang=../slang-1.4.2jp0」のように、slang-jのソースを展開してmakeしたディレクトリを指定してください。なお、共有ライブラリ名が「libslang-ja.so」となっている場合は、あらかじめ、configureとconfigure.inに記述されている「-lslang」を「-lslang-ja」に置換しておいてください。

iconvについては、Cライブラリに実装されている場合は特にオプションを設定する必要ありませんが、libiconvをインストールしている場合は「--with-iconv=/usr/local」のように、インストールしたディレクトリを指定してください。

glibc 2.1以前のCライブラリでは、ワイド文字関数を実装していなかったり、実装していても日本語の扱いで不具合があったりします。そのため、glibc 2.2の環境以外の場合は、必ず「--without-wc-funcs」を指定してください。このオプションを指定すると、Mutt付属のワイド文字関数を使うようになります。

以上がMuttで日本語を使うために最低限必要なオプションです。では、POP3を使う前提でコンパイルをしてみましょう。次の例では、slang-jとlibiconvを/usr/localにインストールし、Mutt付属のワイド文字関数を用いる場合です。regexライブラリもシステムのものが新しいのでそちらを使うことにします。

```
$ ./configure --with-slang=/usr/local \
--with-iconv=/usr/local --without-wc-funcs \
--enable-pop --with-regex
$ make
# make install
```

コンパイルが終わったら、lddコマンドでMuttにリンクされている共有ライブラリを確認してください(実行例1)。libslangやlibiconvを利用していれば、それらが実行例1のように表示されるはずです。

続いて、日本語パッチに含まれている文書と設定ファイルのサンプルを、適当なディレクトリにコピーします。

```
$ cd ../mutt-1.3.17i-ja0
# cp README.JA-PATCH /usr/local/doc/mutt/
# cp manual-ja-patch.txt /usr/local/doc/mutt/
# cp usage-japanese.txt /usr/local/doc/mutt/
# cp mutt-ja.rc /usr/local/doc/mutt/samples/
# cp sample.muttrc-tt /usr/local/doc/mutt/samples/
```

なお、ホームディレクトリにインストールする場合は、スプールをホームディレクトリ以下に置く必要があります。そのため、configureのオプションとして「--prefix=$HOME --with-homespool=FILE」を追加します。FILEは「$HOME/Mailbox」や「$HOME/Maildir/」などのスプールのパスを指定します。システムにMTA*7がインストールされていて、外部から配送される場合で、スプールが/var/spool/mail/にある

実行例1　lddコマンドでライブラリを確認

```
$ ldd `which mutt`
   libslang.so.1 => /usr/lib/libslang.so.1 (0x40021000)
   libm.so.6 => /lib/libm.so.6 (0x40080000)
   libc.so.6 => /lib/libc.so.6 (0x4009f000)
   libdl.so.2 => /lib/libdl.so.2 (0x401c2000)
   /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

ときは、スプールをホームディレクトリに設定するようにしてください。

sendmailの場合は、.forwardファイルに「$HOME/Mailbox」を記述するか、あるいはprocmailなどを使い、スプールをホームディレクトリに設定するようにします。procmailの場合は、.procmailrcに「DEFAULT=$HOME/Mailbox」などを記述します。詳細はそれぞれのマニュアルなどを参照してください。

インストールが終わったらすぐに使ってみたくなりますが、ここはちょっと我慢して、送信用プログラムのインストールを行います。

M 送信用プログラム

Muttは、先に述べたように、送信に関しては外部プログラムを使います。お使いのシステムにsendmailやqmail、exim、postfixなどのMTA[*7]がインストールされていて、外部のドメインあるい中継サーバに送信できる設定が行われている環境であれば、別途プログラムをインストールする必要はありません。

そうでない場合は、これらのMTAのどれかをインストールして設定を行うか、あるいは送信専用のプログラムをインストールする必要があります。ここでは後者の、送信専用のプログラムを2つ紹介します。

●Nomail

「Nomail」は、おおつかまさひと氏によって開発された「ダイアルアップ・モバイル環境用オフラインSMTP サーバ」です（付録CD-ROMに収録）。sendmailをフェイクしているnomailプログラムにより、送信するメッセージをキューに溜めておいて、nosendプログラムによって後でまとめて送信できます。そのため、常にネットワークに接続されているわけではないという、ダイヤルアップユーザーに適しているソフトウェアです。しかし、現在の安定版では、「envelope sender」[*8]としては1つのアドレスしか使えないという制限があります。

インストール方法は、付属のINSTALLドキュメントに記述されている通りに行えばいいだけです。ただし、Muttではnomailをモードで動かすため、inetd.confの設定は基本的には必要ありません。

●nullmailer

「nullmailer」は、送信専用のMUA兼SMTPクライアントソフトウェアです。インターフェイスはqmailの送信部分（qmail-inject、qmail-send）に似せて作られています。メールを送るとすぐに中継用のMTAに送信するため、常時接続あるいはダイアルアップルータを利用している人にお勧めです。オフライン環境でも工夫次第で使えますので、機会がありましたら紹介します。

RPMパッケージも提供されていますので、これを使えば、

```
# rpm -ivh nullmailer-1.00RC5-1.i386.rpm
```

とするだけでインストールできます。

設定は、/etc/nullmailer/remotesに次の1行を記述します。

```
HOSTNAME  smtp
```

なお「HOSTNAME」の部分を、中継に用いるMTAのFQDN（完全修飾ドメイン名）に置き換えてください。例えば、MTAのFQDNがsmtp.example.orgであれば、次のようになります。

```
smtp.example.org  smtp
```

後は、他のデーモンと同じように起動する設定を行います。

一方、ソースから自分でコンパイル、インストールする場合の手順は、以下のようになります。

まず「nullmail」というユーザーを追加します。このユーザーは権限のないグループか、あるいは、「nullmail」というグループを新たに作成して、そこに所属するようにします。

```
$ ./configure
$ make
$ make check
# make install
# make install-root
```

ここまでできたら、/usr/local/etc/nullmailer/remotesに次の1行を記述します。「HOSTNAME」は中継に用いるMTAのFQDNに置き換えます。

```
HOSTNAME  smtp
```

そして、ユーザーnullmailの権限でnullmailer-sendを起動させます。

```
$ nullmailer-send &
```

この起動、停止の制御を簡単に行いたい場合は「daemontools」

*7 「Message Transfer Agent」の略。メッセージをホストからホストへ転送するプログラム。
*8 SMTPのコマンド「MAIL FROM:」で指定する送信者のメールアドレス。途中経路でエラーが生じたら、このメールアドレスに送り返されることになっている。

(付録CD-ROMに収録)というソフトウェアを導入するといいでしょう。daemontoolsを使っている場合は、ソースのscriptディレクトリにサンプルのrunスクリプトが入っていますので、それを適当に編集すれば使えます。なお、daemontoolsに関しての詳細は、本題から外れるので割愛しますが、RESOURCEで紹介するページを参考にすれば、それほど難しくないでしょう。

環境設定

さて、ここまでで、最低限必要なプログラムをすべてインストールしましたので、これから使うための設定を行います。

Muttの設定方法

●設定ファイル

サイト共通の設定ファイルは/usr/local/share/mutt/Muttrcあるいは/etc/Muttrcです。ユーザー全員に共通だと思われる設定はここで行ってください。

一方、ユーザーごとの設定ファイルは~/.muttrcに記述します。このファイルが見つからなければ、Muttは~/.mutt/muttrcを探します。また、-Fオプションで、起動時に任意の設定ファイルを指定することもできます。メールアドレスの別名を保存するaliasファイルなど、別のファイルを呼び出したりすることもできますので、ホームディレクトリに.muttというディレクトリを作って運用することをお勧めします。

●設定ファイルの構文

Muttの設定ファイルの構文は、シェルと同じように

<コマンド> <1つ以上の引数>

の形式になっています(コマンドと引数の間は半角スペース)。例えばこんな感じです。

```
set realname="Mutt user"
auto_view text/enriched text/html
macro index \eb '/~b ' 'search in message bodies'
```

「;#'"\`$」などの文字は、bashやzshのシェルスクリプトと同様の働きをします。それぞれ、「;」がコマンドの区切り、「#」がコメント、「'」が引用符、「"」が二重引用符、「\」がエスケープ文字、「`」がコマンドの出力、「$」が環境変数です。また、「~」はホームディレクトリを示します。

なお、以降の説明において、Muttの変数は「$」を付けて表すことにします。例えば、変数hostnameは$hostnameです。

日本語環境の設定

●環境変数の設定

Muttが国際化関数を使用しているため、ロケールの各カテゴリを設定するために、いくつかの環境変数を設定する必要があります。

1つ目は、文字に関する「LC_CTYPE」カテゴリです。同名の環境変数「LC_CTYPE」を、日本語を示す「ja_JP」に設定する必要があります。ただし、Mutt付属のワイド文字関数はLC_CTYPEカテゴリをいっさい参照しないため、このLC_CTYPEは無視されます。

2つ目は、メッセージカタログに関する「LC_MESSAGES」カテゴリです。メッセージを日本語で表示させる場合は「ja_JP」を、英語のままで表示させる場合は「C」に設定します。

3つ目は、時間に関する「LC_TIME」カテゴリです。Muttではディレクトリ表示の際の日時の表示に関係があります。しかし、Muttではこの環境変数「LC_TIME」は反映されず、Muttの設定ファイルの変数「$locale」で設定した値が有効になります。時間を日本語で表示したくない人はデフォルトの「C」のままで使いましょう。

以上の設定は、LC_ALLが定義されていないことが前提です。また、何も設定していなければ、環境変数LANGの値が有効になります。設定例としては、以下のようになります。

```
$ LC_CTYPE=ja_JP
$ LC_MESSAGES=ja_JP
$ export LC_CTYPE LC_MESSAGES
```

以上の環境変数の設定を行ったら、これから説明する項目に関して、Muttの設定ファイルに記述を行っていきます。なお、日本語環境の設定に関しては、日本語パッチと一緒に配布している「mutt-ja.rc」というファイルに一通り記述されていますので、Muttの設定ファイル内で

```
source /usr/local/doc/mutt/samples/mutt-ja.rc
```

のようにして読み込むようにすれば完了するはずです。

●文字符号化方式の設定

表示用、作成用の文字符号化方式を「$charset」に設定します。使用するエディタの設定で、新規に作成するファイルの文字符号化方式を同じものに指定してください。また、署名ファイル「.signature」の文字符号化方式も同じものにしてください。なお、Mutt付属のワイド文字関数は、ステートフル

な(stateful、つまりエスケープシーケンスによりシフト状態が変わる)ISO-2022-JPを扱えないため、付属の関数を使っている場合は、$charsetにはISO-2022-JPを設定しないでください。設定例を示すと次のようになります。

```
set charset="euc-jp"
```

送信用の文字符号化方式は「$send_charset」に設定します。$send_charsetには、複数の文字符号化方式を「:」をセパレータとしたリストで記述できます。作成したメッセージがどの文字符号化方式で送信できるかをリストの先頭から順番に試します。日本語を使う場合は、次のようにします。

```
set send_charset="us-ascii:iso-2022-jp"
```

テキストの添付ファイル用の文字符号化方式は「$file_charset」で設定します。これは日本語パッチで加えられる機能です。これは次のように指定してください。

```
set file_charset="iso-2022-jp:euc-jp:shift_jis:utf-8"
```

さらに、日本語(ISO-2022-JP)で記述されているにも関わらず、文字符号化方式が指定されていないメールを受け取ることがあります。また、ヘッダにおいてISO-2022-JPの文字列をMIME符号化せずに、そのまま記述したものもあります。Muttは、文字符号化方式の指定がないものを「US-ASCII」と見なすので、標準ではこのようなメールを正しく表示するすることができません。しかし、日本語パッチでは、次の設定を行うことで、そのようなメールを期待通りに表示することができます。

```
set strict_mime=no
set assumed_charset="iso-2022-jp:euc-jp:shift_jis:utf-8"
```

ヘッダに関しては、指定したリストの先頭の文字符号化方式から順番に試されます。本文(body)に関しては、先頭のものしか試されません。本文がうまく表示されないようでしたら、Ctrl+E(edit-type)でContent-Typeフィールドの編集ができますので、適当なcharsetを指定してください。ただし、Content-TypeフィールドのcharsetパラメータがMuttはそのパラメータを信用しますので、実際の文字符号化方式と一致していなければ表示できません。この場合も、Ctrl+E(edit-type)でContent-Typeフィールドの編集モードに入り、適当なcharsetの指定を行ってください。

●日本語の設定

標準の設定では、日本語でボディの検索を行うことができません。必ず次の設定を行ってください。

```
set thorough_search=yes
```

機種依存文字があると、その文字以降の文字列は文字化けします。これを防ぐためには、次の設定を行ってください。ただし、これは日本語パッチの機能です。

```
set sanitize_ja_chars=yes
```

添付ファイルのファイル名が日本語である場合、MIME B encodingの形式に符号化されたものがほとんどです。これはRFC違反(本来はRFC 2231の形式で符号化するべき)であるため、Muttは標準では復号化しようとしません。そのため、これに対応するためには、次の設定を行ってください。

```
set rfc2047_parameters=yes
```

メッセージを転送する際に不具合が生じる可能性があるため、次の設定値はデフォルト値のままにして変更しないでください。

```
set forward_decode=yes
set mime_forward_decode=no
```

Content-Typeが「text/enriched」である日本語のメールは、Muttでは正常に表示できません。そのため、「text/html」であるメールも含めて、外部プログラムを利用して表示することになります。詳しくはコラム「mailcap」をご覧ください。

M その他の基本設定

●送信プログラムの設定

Muttは、送信機能を外部プログラムに任せるために、変数「$sendmail」で送信プログラムを指定する必要があります。sendmailそのものか、あるいはsendmailをフェイク(fake)するMTAがインストールされていれば、インストール時のconfigureで設定されるデフォルト値のままで大丈夫です。

この$sendmailの設定は、「mutt -v」で表示されるメッセージのSENDMAILの項目で確認できます。そのMTAの設定が適切に行われていて、かつ、envelope senderを指定する-fオプションが使えるのであれば、メッセージヘッダのFromフィールドからenvelope senderを取り出す「$envelope_from」を

```
set envelope_from=yes
```

のように設定するだけです。この変数を設定すると-fオプションを自動的に付けるようになります。

もし、-fオプションが使えない、あるいは、MTA付属のMUAとは別のプログラムを使っているのであれば、変数$sendmailで送信プログラムを明示し、envelope senderを指定するオプションを付けてください。以下に、いくつかの例を示します。

Nomailを使う場合は、envelope senderは固定であるため、特にオプションは必要ありません。

```
set sendmail="/usr/local/sbin/nomail"
```

nullmailerを使う場合は、-fオプションが使えるので、$envelope_fromを設定した場合にはenvelope senderを指定する必要はありません。

```
set sendmail="/usr/local/bin/nullmailer-inject"
```

●ユーザーの情報の設定

ユーザー自身のメールアドレス、名前なども、mattrcファイルに設定します。$hostnameは、メールアドレスのドメイン部(@の右側)だけを記述します。ヘッダのFromフィールドやMessage-IDフィールドがこの情報に基づいて生成されます。

```
set realname="Mutt user"
set from="mutt@example.org"
set hostname="example.org"
```

●メールボックスの形式とフォルダの設定

Muttがサポートしているメールボックスの形式は、mbox、MMDF、MH、Maildirです。それぞれの違いや特徴についてはコラム「メールボックスの形式」を参照してください。ここではMaildir形式のメールボックスを使うことにします。

メールボックスを読むときは自動認識されるため、特にその種類を指定する必要はありませんが、作成するときに使用するメールボックスの形式を「$mbox_type」という変数で指定する必要があります。

```
set mbox_type=Maildir
```

COLUMN 2

mailcap

Muttは、text/plain、text/enriched、message/rfc822などのいくつかのMIMEタイプをサポートしていますが、それ以外のメッセージの表示については、外部のプログラムにその処理を任せています。その動作を補助するための仕組みとして、「mailcap」を利用します。

mailcapを使うには、まず、Muttの設定でmailcapファイルのパスを指定します。ユーザのmailcapファイルを「$HOME/.mailcap」とすると、サイトの設定ファイルも含めて次のような設定を行います。

```
set mailcap_path="$HOME/.mailcap:/etc/mailcap"
```

次に、自動閲覧(autoview)を行うMIMEタイプをauto_viewというコマンドで設定します。次の例はtext/enrichedとtext/htmlのメッセージを自動閲覧する場合です。

```
auto_view text/enriched text/html
```

さらに、mailcapファイルにそれぞれのタイプのファイルを表示できるプログラムを指定してください。以下の例はtext/enrichedなパートをrichtextで、text/htmlなパートをw3mで、それぞれ表示するようにする例です。

```
text/enriched; nkf -e %s | richtext -t; copiousoutput
text/html; w3m -dump -T %t %s; copiousoutput
```

それぞれの外部プログラムに渡されるときに、「%s」はファイル名に、「%t」はMIMEタイプに置き換えられます。なお、行末の「copiousoutput」は、外部プログラムの標準出力をMuttのページャに渡すことを示します。自動閲覧の場合は必ずこれを付けてください。

ちなみに、Muttは日本語のtext/enrichedなメッセージは表示できないので、上のように設定した方がいいでしょう。

text/htmlなメッセージを自動閲覧しない場合はtext/htmlをauto_viewの設定から外します。このときは、mailcapの設定ファイルのtext/htmlの項目行の最後の「; copiousoutput」を取り除いてください。

自動閲覧しない場合についても、例を示しておきましょう。画像ファイルをImageMagickパッケージのdisplayコマンドで表示する場合と、text/htmlなメッセージをnetscapeを起動して表示する場合は、以下のように設定します。

```
image/*; display %s
text/html; netscape %s
```

なおMetamailパッケージには、先ほど例で示したrichtextを始め、MIMEを扱うために便利なツールやmailcapのサンプルファイルなどが入っていますので、まだの人は、ぜひMetamailを導入することをお勧めします。大抵は標準でインストールされていると思われますが、インストールされていなければパッケージシステムで追加してください。(滝澤隆史)

さらに、各種フォルダの設定を行います。

「$spoolfile」は、メールがスプールされているフォルダを指定します。

```
set spoolfile="~/Maildir"
```

これを指定しない場合は、環境変数MAILで指定されたものが設定されます。ショートカットは「!」です。

ショートカットは、特定のフォルダを示すことができ、設定ファイルだけでなく、ファイルやディレクトリの入力に使うことができます。なお、パターンの中で使う場合は論理否定演算子の「!」と区別するために、引用符で囲まれた文字列の中で使用してください。

$folderは、メールボックスのデフォルトの場所を指定します。デフォルト値は「~/Mail」です。ショートカットは「+」または「=」です。ただし、このショートカットを使うためには、他の変数の設定を行う前に、この変数を設定しなければなりません。

```
set folder="~/Mail"
```

また$mboxは、$spoolfileフォルダから読み込んだメールを格納するフォルダを指定します。ショートカットは「>」です。次の例は~/Mail/mboxを指定した場合です。

```
set mbox="+mbox"
```

先に記述したように、+は「~/Mail/」に展開されます。

$recordは、送信したメッセージを保存するフォルダを指定します。いわゆるFccです。ショートカットは「<」です。次の例は、~/Mail/outboxを指定した場合です。

```
set record="+outbox"
```

$postponedは、作成したメッセージの送信を延期したときにメールを保存するフォルダを指定します。次の例は~/Mail/postponedを指定した場合です。

```
set postponed="+postponed"
```

$tmpdirは、一時ファイルを置く場所を指定します。次の例は、「~/tmp」を指定した場合です。

```
set tmpdir="~/tmp"
```

●エディタの設定

使用するエディタは、$editorで設定します。以下の例は、エディタにjedまたはEmacsを使う例です。なお、それぞれ、バックアップファイルを作成しないオプションを指定しています。バックアップファイルを作成するようにしていると、先に指定した$tmpdirフォルダが、書いたメールのバックアップファイルでいっぱいになってしまうからです。

```
set editor="jed %s -f set_buffer_no_backup"
set editor="emacs %s --eval '(setq make-backup-
files nil)'"
```

●POP3の設定

MuttのPOP3機能を使う場合は、$pop_hostに次のようなPOP URLの形式で指定します。

```
[pop[s]://][username[:password]@]popserver[:port]
```

ただし、認証方式を明示的に指定することは出来ず、Muttは、SASL、APOP、USER/PASSの順に認証を試すようになっています。パスワードを記入するのは危険であるため極力避けるべきですが、記入する場合は、この設定ファイルが他人に絶対に見られないようにパーミッションを設定してください。以下は、ユーザー「mutt」が「pop.example.org」からメールを取得する場合の例です。

```
set pop_host="pop://mutt@pop.example.org"
```

なお、このPOP周りの仕様は、1.3系列の途中から大きく変更されたので、それ以前のバージョンを使っている方は注意してください。

また、複数のPOPサーバからメールを受信したり、UIDLコマンドを利用して既読のものだけを受信したり、受信時に振り分けをしたい場合は、別のプログラムのfetchmailを使いましょう。fetchmailは最後の章で説明します。

M カスタマイズ

以上の設定で、最低限、Muttを使えるようになります。ここからは、より快適に使うために、いくつかのカスタマイズをしてみましょう。

●表示に関するカスタマイズ

X上のターミナルエミュレータを使っている場合は、その表示行数を大きくしてからMuttを起動するようにしてみましょう。シェルスクリプトを作り、それで起動してみてもいいでしょう。次の例は、ktermを40行にしてmuttを起動するシェルスクリプトです。

```
#!/bin/sh
```