百万人のMutt ——Mutt活用講座

滝澤隆史
taki@cyber.email.ne.jp

第1回　メールボックスの移行とメールアドレス管理

はじめに

先月号の特集に引き続き、今月からMuttの連載を開始します。今回の前半はメールボックスの移行とメールアドレスの管理について取り上げ、後半は操作系のカスタマイズに関して説明します。

本題に入る前に、Muttの最新情報をお知らせします。5月に入ってから開発系の最新版「1.3.18i」が公開されました。注意すべき変更点は、設定変数「$in_reply_to」がなくなったことです。これは、4月下旬にRFC 822*1の改訂版としてRFC 2822が公開され、そこでヘッダの「in-reply-to」フィールドの文法が変更されたためです*2。これに伴い、Muttでは、従来の「in_reply_to="%i"」を設定したのと同じヘッダを生成することになります。今まで設定していた方はバージョンアップの際にその設定を削除するようにしてください。

メールボックスの移行

●既存のメールボックスの利用

MUAを他のものに移行しようと思ったときに、まず頭を悩ませるのがメールボックスの移行です。Muttの場合は、mbox、MMDF、MH、Maildirといった、よく使われているメールボックス形式をサポートしているため、以前使っていたメールボックスがこれらの形式であれば、そのまま使うこともできます。

mbox、MMDF、Maildirの場合は、$folderで指定したディレクトリの下に、そのファイルやディレクトリを移せばいいだけです。

MHを使っている場合は注意が必要です。Muttは、そのディレクトリがMHフォルダかどうかを、.mh_sequencesや.xmhcacheといったファイルがあるかどうかで判断します。しかし、MHフォルダを扱うMUAやツールで作成されたフォルダには、必ずしもこれらのファイルがあるわけではありません。そのため、Muttでこれらのフォルダを開こうとすると「xxxはメールボックスではない」と言われることがあります。

この問題に対応するために用意されたのが、川口銀河氏が日本語パッチに提供した$mh_pathです。これを設定すると、この判断条件が変わり、.mh_sequencesや.xmhcacheといったファイルがなくてもMHフォルダであると認識してくれます。この判断条件は、

- $mh_pathで指定したディレクトリ以下である
- 他のフォルダ形式の条件に該当しない
- サブディレクトリを持たないディレクトリである

となっています。MHフォルダを扱うMUAから移行する場合は、ぜひこの$mh_pathを設定しておきましょう。

●メールボックスの形式の変換

以上は、既存のメールボックスをそのまま使う方法です

*1　メールなどのメッセージの形式を定めたRFC。
*2　筆者のWebサイトに「スレッドの生成(Message-ID、In-Reply-To、References に関して)」というタイトルで、変更内容について説明した記事を公開しています(http://www.emaillab.org/essay/thread.html)。

が、Muttに乗り換えるついでにメールボックスの形式も変更してみたいと思う人もいるでしょう。この場合は、まず$mbox_typeに新しいメールボックスの形式を設定します。これまでに説明したように、Muttは既存のメールボックスを読むことができます。ということは、既存のメールボックスを読み込み、その中のメッセージを新しいフォルダに保存すれば、保存先のフォルダは新しいメールボックスの形式になって、首尾よく移行できるというわけです。

具体的な手順は以下のようになります。

(1)新しいメールボックスの形式を$mbox_typeに設定
(2)Muttで移行元のメールボックスのフォルダに移動
(3)インデックス画面にてTキー(tag-pattern)を入力
(4)検索パターンとして「~A」を入力(この操作で、全メッセージにタグが付けられる)
(5);キー(tag-prefix)を入力後、sキー(save-message)を入力
(6)保存先として、新しいフォルダ名を入力

この(2)〜(6)までの操作を各メールボックスに対して行えばよいでしょう。なお、それぞれの設定、操作は前号の特集ですべて説明していますので、そちらもご覧ください。

●mbox2maildir

mbox形式からMaildir形式への変換に関してだけは、Muttを使わず、別のツールを使って移行する方法があります。これは、Ivan Kohler氏の「mbox2maildir」というPerlのスクリプトです(記事末RESOURCE[1]を参照)。

使い方を簡単に説明します。まず、スクリプトの先頭行のPerlのパスを使用環境に合わせて修正してください。次に、既存のmboxファイルのパスを環境変数「MAIL」に、移行先のMaildirディレクトリのパスを環境変数「MAILDIR」に設定してください。後はこのスクリプトを実行するだけです。例を示します。

```
$ MAIL=$HOME/Mailbox
$ MAILDIR=$HOME/Maildir/
$ export MAIL MAILDIR
$ mbox2maildir
```

なお、このスクリプトの実行後は、問答無用に既存のmboxファイルは削除されますので、念のため残しておきたいという場合は、スクリプトの最終行の「unlink("$ENV{MAIL}");」をコメントアウトしてください。

M メールアドレスの管理

メッセージを作成するときに宛先のメールアドレスを毎回入力するのは面倒なことです。また、メールアドレスを正確に覚えておくなんてことは、そうそうできるものではありません。そのためMuttには、メールアドレスの入力を補助するための機能としてエイリアスとクエリー(外部へのアドレスの問い合わせ)があります。

●エイリアスを使う準備

エイリアス(alias)は、文字通りメールアドレスに対応付けられた別名で、たいていは名前やニックネームなどを付けます。Muttでこのエイリアスを使うためには、次のような2行をMuttの設定ファイルに記述する必要があります。なおこの例では、エイリアスを設定するファイルを~/.mutt/aliasとします。

```
set alias_file=~/.mutt/alias
source ~/.mutt/alias
```

$alias_fileには、Muttのcreate-alias機能(入力キーはaキー)によって作成されたエイリアスを追加するファイルを指定します。このファイルの実体はエイリアスを定義するMuttのaliasコマンドの集まりです。例えば、次のようなコマンドが記述されていきます。

```
alias mutt-dev mutt-dev@mutt.org
alias mutt-j mutt-j@ribbon.or.jp
```

このようにエイリアスを定義するファイルは、単なるMuttのaliasコマンド群であり、エイリアスを使うに当たって特別扱いはされません。そのため、エイリアスを使うためには、このファイルをsourceコマンドで読み込まなければなりません。また逆に、$alias_fileで指定していないファイルでもsourceコマンドで読み込みさえすれば、そのファイルで定義したエイリアスが使えます。極端な話、Muttの設定ファイル~/.mutt/muttrcにaliasコマンドを記述しても構いません。

●エイリアスの登録

以上でエイリアスを使う設定ができたので、実際にメールアドレスを登録してみましょう。

まず、インデックス画面において、登録したいメールアド

レスがヘッダのFromやReply-Toフィールドに記述されているメッセージにカーソルを移動させます。Reply-ToはFromより優先されます。ここで、aキー(create-alias)を入力すると、そのメールアドレスのエイリアスを作成できます。このとき、エイリアスには、デフォルトとしてメールアドレスの「@」の左側の文字が設定されますが、これは、自分で好きなように(覚えやすいように)変更します。メールアドレスはデフォルトのものでいいでしょう。カーソルがメッセージの上になければ、デフォルトの文字列は設定されず、各項目を自分で入力することになります。

aliasコマンドは

```
alias <key> <address> [, address, ……]
```

のように1行1件の形式なので、他のMUAで使っていたエイリアスファイルやメールアドレスのデータがあるのであれば、PerlやAWKなどのスクリプト言語を用いて簡単にMutt用のエイリアスファイルを生成できると思います。

ここで「address」の部分は、ヘッダのTo、Cc、Bccフィールドに入力される形式ですので、RFC 2822に従った形式にする必要があります。例えば、次のようになります。

```
alias mutt-j mutt-j@ribbon.or.jp
alias mutt-dev Mutt Developer <mutt-dev@mutt.org>
alias mutt-user mutt-user@mutt.org (mutt-user)
```

また複数のアドレスを定義する場合は、

```
alias mutt mutt-dev@mutt.org, mutt-po@mutt.org
```

のように、カンマで区切ります。

●エイリアスの使い方

次にエイリアスを使ってみましょう。メッセージを新規に作成するときに、ラインエディタ上で宛先(To)を入力しますが、このときに、エイリアスを入力してEnterキーを押します。エディタが起動したら、ヘッダのToフィールドに入力したエイリアスがメールアドレスに置き換わっているのを確認できます。

実際にエイリアスを使う場合はもっと横着をして、補完機能を使うのが普通でしょう。エイリアスの先頭の数文字を入力したところで、Tabキーを押すと補完の機能が働きます。エイリアスを最後まで入力した場合、あるいは複数の候補がある場合は、エイリアスのリスト画面に切り替わり、候補の一覧が表示されます。カーソルを入力したいものの上に移動させ、改行キーを押すと、そのアドレスがラインエディタに入力されます。また、始めから何も記入せずにTabキーを押すと、登録されているエイリアスがすべて候補として表示されます。ここで、選ぶのもいいでしょう。

このエイリアス画面でも他の画面と同様に/キーで検索を行うことができます。また、Spaceキーやtキーでタグを付けることにより複数のアドレスを選択することができます。

ここで例を示します。先ほどの例のエイリアスが登録されているとします。まず、「To:」に「mutt」と入力します。

```
To: mutt
```

ここでTabキーを押すと、画面1のようにエイリアスの候補の一覧が表示されます。カーソルを「1」に合わせてEnterキーを入力すると、そのアドレスがラインエディタに入力されます。後は、Enterキーを押すだけです。

●クエリ

Muttでは、LDAP、ph/qi、bbdb、NISといった外部のディレクトリデータベースにラッパースクリプトを通してアクセスすることができます。外部データベースを使うためには、それにアクセスするためのラッパースクリプトを「$query_command」に設定します。このとき、スクリプトのオプションとして「%s」を指定してください。%sが、クエリの文字列に置き換えられます。設定例を示すと次のようになります。

```
set query_command = "mutt_ldap_query2-ja.pl '%s'"
```

ここで使うことのできるスクリプトは、「External Address Query Scripts」([2])からダウンロードできます。また、LDAPで日本語での検索に対応したスクリプトは、「Mutt Japanese Edition」のWebページ([3])からダウンロードできます。なお、スクリプトやデータベースによっては検索の動作(部分一致検索が可能かどうかなど)が異なることがあるのでご注意ください。

画面1　エイリアスの候補一覧

使い方としては2通りあります。1つは、インデックス画面にてクエリを行なう方法です。Qキー(query)を入力するとクエリの入力を要求するプロンプトが表示されます。ここで、検索したい名前を入力し、Enterキーを押します。

```
問い合わせ: mutt
```

検索に成功したら、部分一致した候補が次のようなリストで表示されます。

```
1  taro mutt     tmutt@example.org
2  hanako mutt   hmutt@example.org
```

目的のメールアドレスにカーソルを移動します。エイリアスを作成する場合は、ここでaキー(create-alias)を入力します。また、このアドレスを宛先としたメールを作成する場合は、mキー(mail)を入力します。なお、さらに検索を行なってそのクエリの結果を追加したい場合は、Aキー(query-append)を入力します。ここでは、タグを付けて複数のメールアドレスを選択することもできるので、複数のアドレスにタグを付けた後に、;キー(tag-prefix)を入力しmキーを入力すると、そのアドレスが宛先としてセットされた状態でメールを作成できます。

2つ目の方法は、ラインエディタでメールアドレスを入力する際に、クエリを行なう方法です。宛先の名前やメールアドレスの一部を入力してCtrl+Tキー(complete-query)を入力します。クエリの結果が1つだけであれば、ラインエディタ上にそのメールアドレスが入力されます。もし複数の候補があれば、最初に紹介した方法と同様にリストで表示されます。このとき、目的のメールアドレスにカーソルを移動して、Enterキーを押すとラインエディタ上にメールアドレスが入力されます。また、タグを複数のアドレスに付けてEnterキーを押せば、それぞれのアドレスがラインエディタ上に入力されます。

操作系のカスタマイズ

●キーの割り当て

Muttでは、各機能に対するキーの割り当てを自由に設定できます。キー割り当ての文法は次の通りです。

```
bind <map> <key> <function>
```

<map>は、そのキーが使うことのできるメニューで、次のものがあります。

表1　特殊キーの割り当て

文字列	キー
\t or <tab>	Tab
\n	newline
\e or <esc>	Esc(Escape)
<up>	↑(up arrow)
<down>	↓(down arrow)
<left>	←(left arrow)
<right>	→(right arrow)
<enter>	Enter
<space>	Space
<f1>	F1(function key 1)

```
generic  index   pager  alias    query   attach
compose  postpone        browser  pgp     editor
```

genericは、特定のメニューというわけではなく、ここで定義されたキーはpagerとeditor以外で共通に使えます。

<key>の部分には、割り当てられるキーを指定します。連続したキーでも構いません。「Ctrlキーを押しながらxを押す」というキー操作は「\Cx」で表わします。このCとXは、大文字、小文字の区別をしません(つまり、Ctrl+xというキー操作と、Ctrl+Shift+xというキー操作は区別されず、同じものとして扱われます)。特殊なキーの表記については、Muttのマニュアルの「3.3. Changing the default key bindings」に記述されていますが、表1にその一部を紹介します。

<function>はMuttの機能のことです。各メニューで使うことのできる機能およびキーの割り当ては、ヘルプ(?キー)で確認できます。また、マニュアルの「6.4. Functions」にも記述されています。

キー割り当ての設定例として、lessのようにNキーで逆順検索(search-opposite)を行なうようにキーの割り当てを変える例をリスト1に示します。search-oppositeをNキーに割り当てるためには、既存のNで割り当てられているキーを、「noop」に割り当てることにより無効にして、別のキーに割り当て直す必要があります。ここでは「\eN」(Esc+N)に割り当て直しています。

リスト1　Nキーで逆順検索するためのキー割り当て(設定例)

```
bind index   N    noop
bind index   \eN  toggle-new
bind pager   N    noop
bind pager   N    search-opposite
bind pager   \eN  mark-as-new
bind browser N    noop
bind browser \eN  select-new
bind query   N    search-opposite
bind generic N    search-opposite
```

COLUMN

Muttも歩けば棒に当たる

このコーナーでは、Muttユーザーに登場してもらって、どのようにMuttを使っているかを紹介してもらいます。初回は開発系日本語パッチのメンテナーの滝澤が担当します。

まず、私とMuttとの関わり合いについて紹介しましょう。私がMutt(犬)を使い始めたのはそれほど古くなく1年半前です。それまではMew(猫)を使っていました。その当時のMuttのバージョンは1.01で、吉田さんのjpパッチを当てて使っていました。しかし、何を思ったか、途中から開発系1.1に手を出し、jpパッチなしに、最低限の日本語が通るようにして使い始めました。そうしているうち、ちょうど1年前、Mutt 1.2が出ると同時に開発系1.3が出ました。国際化パッチが取り込まれたという話なのですぐに使ってみましたが、日本語を使うに当たって不具合がいくつかありました。そこで、不具合を修正するパッチを作ってmutt-devメーリングリストに送ってみたり、mutt-jメーリングリストに公開してみたりしているうちに、いつの間にか日本語パッチのメンテナンスをするようになり、今に至ります。

次に、私が実際に、Muttをどのように使っているかを紹介します。まずは、どのような設定ファイルがあるかを見てみます。私の$HOME/.mutt/ディレクトリの中を覗いてみると、表2のようなファイル構成になっています。

3つのアカウント用の設定ファイルは、それぞれsourceコマンドでcommonを読み込むように、さらに、commonは、colorとaliasを読み込むようにしています。

Muttの起動方法としては、直に起動するのではなく、各アカウントごとに用意したシェルスクリプトから起動しています。このスクリプトでは、ロケール関係の環境変数を設定し、80桁40行のサイズのKtermを、-eオプションでmuttを指定して起動し、さらにmuttの-Fオプションでアカウント用の設定ファイルを指定しています。例えば、mutt-mainというシェルスクリプトはリスト2のようになっています。

メールの受信はfetchmailで行って、maildropで振り分けています。メーリングリストやアナウンス系のメールをそれぞれのフォルダに振り分け、さらに、フリーメール系のドメインから来たものと同報送信メーラーを使ったと思われるものは「SPAM」としてゴミ箱フォルダ送りにしています。どの条件にも一致せず、最後まで残ったものが個人宛のメイルということになります。

メイルを読み進めていく操作は、インデックスとページャにおいてはひたすらスペースキー(display-message)とタブキー(next-new)です。未読のメッセージがなくなったら、cキー(change-folder)で未読のあるフォルダがデフォルトで入力されているので(mailboxesコマンドで設定してあれば)、そのままEnterキーを押します。

メールを書く環境としては、エディタとしてEmacsを使っています。折角ならJedにしたいところですが、利用しているElisp系アプリケーションがあるのでなかなか移行できません。

送信に関しては、複数のアカウントが使えるnullmailerを使っています。

と、大体こんな感じです。最後に、Muttの標語とも言うべきこの言葉で締めくくります。

All mail clients suck. This one just sucks less.

(滝澤隆史)

表2　$HOME/.muttディレクトリのファイル構成

main	メインのアカウント用
sub	海外メイリングリスト向けアカウント用
co	お仕事アカウント用
common	共通
alias	エイリアス
color	色の設定
signature-ja	日本語の署名
signature-en	英語の署名
muttrc	mainのスタティックリンク

リスト2　シェルスクリプトmutt-main

```
#!/bin/bash
LC_CTYPE=ja_JP
LC_MESSAGES=C
ARGS=$@
COLORFGBG="white;black"
export LC_CTYPE LC_MESSAGES ARGS COLORFGBG
kterm -fg white -bg black -xim -geometry 80x40 -e sh -c 'mutt -F ~/.mutt/main $ARGS' &
```

リスト3　2種類の署名ファイルを切り替えるマクロ

```
macro index \csj ":set signature=~/.signature-ja\n" "Set Japanese signature."
macro pager \csj ":set signature=~/.signature-ja\n" "Set Japanese signature."
macro index \cse ":set signature=~/.signature-en\n" "Set English signature."
macro pager \cse ":set signature=~/.signature-en\n" "Set English signature."
```

●キーボードマクロ

キーボードマクロを使うことにより、連続したキー操作を一括して行なうことができます。文法は次の通りです。

```
macro <menu> <key> <sequence> [description]
```

<menu>は、そのマクロが使うことのできるメニューのことで、<key>はマクロを実行するためのキーです。いずれもキー割り当ての項目で説明した方法と同じですので、そちらを参照してください。

<sequence>は、連続したキーの操作を記述したものです。基本的にはキーの割り当ての項目で説明したキーの記述方法を用います。ただし、コントロールキーはキャレット「^」で表します。例えばCtrl＋x は「^x」となります。キャレット自身は「^^」で表します。またここの部分には、Muttの機能を直接「<function>」のように記述することができます。

[description]はこのマクロの説明で、省略可能です。

それでは、マクロの例を見てみましょう。まず、サンプルとしてMuttrcに記述されているマクロを見てみます。

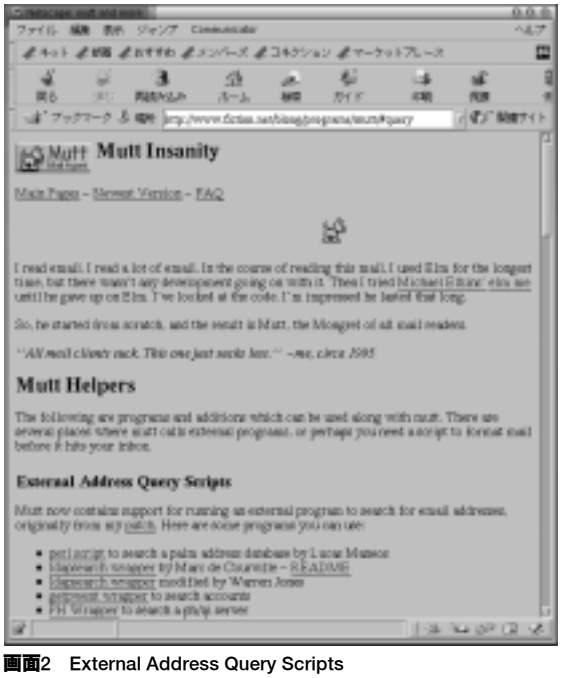
```
macro index \eb '/~b ' 'search in message bodies'
macro index \cb |urlview\n 'extract URLs'
macro pager \cb |urlview\n 'extract URLs'
```

1行目は「Esc b」で実行するマクロで、メッセージのボディを検索します。/キー(search)の後に、パターンとしてメッセージボディを示す「~b」が記述されています。2行目と3行目は、インデックスにおいてカーソルのあるメッセージおよび現在ページャに表示されているメッセージを外部コマンドのurlviewに渡します。これは|キー(pipe-message)でパイプの機能を使っています。

次に実際にマクロを書いてみましょう。ここでは、日本語用と英語用にそれぞれ用意した署名ファイルを相互に切り替えるマクロを作ってみます。通常の設定の切り替えは、インデックスとページャでできればいいので、<menu>としては、indexとpagerの2つを設定します。ただし、2つ同時に設定できないので、それぞれ別のマクロとして定義する必要があります。

また<key>としては、署名の切り替えを行なうため「s」に関係あるキーに設定したいところですが、すでに別の機能に割り当てられているため、空いているCtrl＋s(\cs)の後に「j」と「e」を使うことにします。

<sequence>としては、まず、設定コマンドを実行する:キーで始まります。次に変数$signatureに日本語の署名ファイル「~/.signature-ja」あるいは英語の署名ファイル「~/.signature-en」を設定するコマンドを記述します。このようにして、リスト3のようなマクロができました。これを読み込んでおけば、「Ctrl＋s j」というキー操作で日本語の署名を、「Ctrl＋s e」というキー操作で英語の署名を設定できます。



画面2　External Address Query Scripts

R E S O U R C E

[1] **The qmail home page**
http://www.qmail.org/

[2] **External Address Query Scripts**
http://www.fiction.net/blong/programs/mutt/#query

[3] **Mutt Japanese Edition**
http://www.emaillab.org/mutt/