百万人のMutt
～Mutt活用講座～

滝澤隆史
taki@cyber.email.ne.jp

第6回 POP3

Muttの正式版1.4が近々公開される予定です。この記事を皆さんが読むころには公開されていると思います。開発系1.3の開発が始まってから1年5カ月という長い期間をかけただけあって、機能もたくさん増えて便利になっていますし、かなり安定しています。そこで、前半では1.2から1.4への主な変更点や追加された機能のいくつかを説明します。過去の記事と重複するような内容が多いため、所々「○月号の記事を参照してください」との記述がありますがご了承ください*1。後半では、変更点の大きな1つであるPOP3関連の機能について説明します。

主な変更点

●変更点のリスト

1.2.5から1.4への設定変数、設定コマンド、機能の変更点を表1から表5までにまとめました。

●国際化

1.4の最大の特徴は、国際化されたことで「標準で」日本語が扱えることになったことでしょう。

1.2でも、シングルバイト文字圏内に関しては部分的に国際化が行われていたのですが、1.4では、文字符号化方式の変換関数iconvとワイド文字関数に対応したことで、マルチバイトである日本語の文字も扱うことができるようになりました。ただし、システムのCライブラリが日本語に対応したiconvとワイド文字関数を実装していることが前提です。実装していない場合でも、ワイド文字関数に関してはMutt内蔵のものが日本語を扱えるため、それを使うことで対応できますし、iconvに関してはGNU libiconv（[2]）を別途インストールすることで対応できます。

日本語のメッセージカタログも含まれているため、各種メッセージが日本語で表示できます。

また、国際化に伴い、設定変数$charsetと$send_charsetの仕様が変わりました。表示／編集用文字符号化方式を指定する$charsetは、国際化されたシステムではlocaleから自動的に取得できるようになりました（表1）。もちろん従来通り、明示的に指定しても構いません。送信用文字符号化方式を指定する$send_charsetは次のようなコロン区切りのリストになり、リストの中から送信に適したものに変換して送るようになりました。

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

国際化に関しては、特に1.2.x以前の日本語版Muttを使っている人が戸惑うと思われる点があります。1.2.x以前に生成した$recordフォルダ（作成したメッセージのコピーが保存されるフォルダ）のメッセージが文字化けして表示されることです。これは、1.2.x以前ではメッセージを作成した状態（EUC-JP）のまま保存されるので、Content-Typeフィールドのcharsetパラメータ（ISO-2022-JP）と一致していないためです。1.2.x以前ではこのようなメッセージでも無理やり表示するようになっていたのですが、1.4の日本語版では特に何もせずオリジナルの動作に従っています。このオリジナルの動作とは「送信した状態の文字符号化方式（大抵はISO-2022-JPのはず）で保存される」かつ「文字符号化方式の変換はcharsetパラメータで判断する」ということです。対処方法としては$recordフォルダのメッセージをnkf*2などを使ってEUC-JPからISO-2022-JPに変換してください。変換シェルスクリプトの例をリスト1（mutt-e2j）に示します。

なお、日本語特有の問題に対処するため、日本語版ではいくつかの拡張を行っています。詳しくは本誌2001年11月号の記事をお読みください。

【リスト1】EUC-JPからISO-2022-JPへの変換シェルスクリプトの例（mutt-e2j）

```
#!/bin/sh
#   requirement: GNU grep, nkf
backup=$HOME/tmp/backup
record=$HOME/Mail/record

if [ ! -d $backup ]; then
  mkdir -p $backup || exit 1
fi
cd $record || exit 1
for file in *; do
  egrep -q "(^X-Mailer: Mutt|^User-Agent: Mutt/1\.(1|2))" $file && \
  grep -i -l "^Content-Type:.*charset=iso-2022-jp" $file && \
  mv $file $backup/ && \
  nkf -Ejm0 $backup/$file > $file
done
exit 0
```

●POP/IMAP機能の改善

次の大きな特徴はPOP/IMAP機能の強化です。項目を次に挙げます。

- POPにおけるAPOP/SASL認証のサポート
- SASL認証の強化
- STARTTLSのサポート
- $preconnectのサポート
- $tunnelのサポート
- POPフォルダのサポート
- POP/IMAP URLのサポート
- IPv6対応（$use_ipv6）
- IMAPサーバへの新着メッセージのチェック機能（imap-fetch-mail）

また、new-mailbox機能はcreate-mailboxに名前が変わりました。

日本語版の独自実装であった$pop_apopはなくなりました。オリジナルの方でAPOP認証がサポートされたのでそちらの設定を行ってく

【表1】廃止された機能

機能	機能名	キー
国際化関連	change-charset	null
	recode-attachment	null
POP/IMAP関連	new-mailbox	n

＊1　バックナンバーをお持ちでない方は、バックナンバーを購入するか、編集部のご厚意により筆者のWebサイト（記事末のResource【1】を参照）に過去の記事を転載させていただいているので、そちらをご覧ください。　＊2　-m0オプションを付けてMIME解析機能を無効にしてください。

【表2】新しい設定変数

機能：国際化関連
assumed_charset (*ja)
create_rfc2047_parameters (*ja)
file_charset (*ja)
sanitize_ja_chars (*ja)
strict_mime (*ja)
機能：POP/IMAP関連
imap_authenticators
imap_delim_chars
imap_peek
connect_timeout
pop_auth_try_all
pop_authenticators
pop_checkinterval
pop_reconnect
preconnect
ssl_starttls
tunnel
use_ipv6
機能：フォルダ関連
maildir_trash
mh_seq_flagged
mh_seq_replied
mh_seq_unseen

機能：PGP関連
pgp_good_sign
pgp_ignore_subkeys
機能：その他
compose_format
digest_collapse
display_filter
keep_flagged
print_split
sig_on_top
sleep_time
text_flowed
delete_prefix (*ja)
delete_regexp (*ja)
msgid_use_from (*ja)
pager_spoil (*ja)
pager_spoiler (*ja)
tree_chars (*ja)
tree_llcorner (*ja)
tree_ulcorner (*ja)
tree_ltee (*ja)
tree_hline (*ja)
tree_vline (*ja)

＊ja　日本語版のみの変数

【表3】廃止された設定変数

機能：POP/IMAP関連
imap_checkinterval
imap_cramkey
imap_preconnect
pop_port
pop_apop (*ja)
機能：PGP関連
pgp_sign_micalg

機能：その他
in_reply_to
numbered_ml (*ja)
pager_spoiler (*ja)
pager_spoiler_char (*ja)
kanjithread (*ja)

＊ja　日本語版のみの変数

【表4】新しい機能

機能	機能名	キー
POP/IMAP関連	create-mailbox	C
	imap-fetch-mail	null
PGP関連	check-traditional-pgp	<Esc>P
ラインエディタ	forward-word	<Esc>f
	backward-word	<Esc>b
	upcase-word	<Esc>u
	downcase-word	<Esc>l
	capitalize-word	<Esc>c
	kill-eow	<Esc>d
	transpose-chars	null
その他	collapse-parts	v

【表5】新しい設定コマンド

コマンド名
account-hook
iconv-hook
message-hook

ください。

POPに関しては、記事の後半で解説します。IMAPに関しては次回で取り上げる予定ですのでお待ちください。

●フォルダの改善

Maildirにおいて「trashed」フラグがサポートされました。$maildir_trashを設定すると、deleteフラグが付けられたメッセージは削除される代わりにtrashedフラグが付きます。

MHフォルダが「.mh_sequences」ファイルをサポートするようになりました。フラグを保存するために「unseen」、「flagged」、「replied」が使われます。これは設定変数$mh_seq_unseen、$mh_seq_flagged、$mh_seq_repliedを使って設定できます。この変更に伴い、各メッセージファイルのStatusフィールドを書き換えることはなくなりました。なお、1.3.21から変更されたため、それ以前に設定されたフラグはすべてクリアされるので注意してください。

●PGP機能の改善

PGP機能に関しては、表2、表3、表4にそれぞれ示したような設定変数や機能の増減がありました。細かい点でいろいろ改良されています。設定変数や機能の詳細いついては、本誌2001年10月号の記事をご覧ください。

●ラインエディタの強化

ラインエディタ上でEmacsライクな操作ができる機能が加わりました（表4）。動作はその機能の名前から判断できると思います。割り当てられているキーは標準のEmacsの操作と同じですが、transpose-charsだけは割り当てられていません。

●mailto URLのサポート

Muttのコマンドラインでmailto URLを使うことができるようになりました。例えばリスト2のように、特別なオプションを必要とせずに直接mailto URLを引数としてMuttを起動することができます。mailto URLではシェルで特別な意味を持つ文字「&」や「?」が使われるので、2つ目と3つ目の例のように、mailto URLを引用符で括る必要があります。mailto URLの文法に関して詳しくは、「RFC 2368 The mailto URL scheme」を参照してください。

Muttで読んでいるメッセージ中のmailto URLに対してMuttを起動したい場合はurlviewを使います。まず、urlviewパッケージに含まれているurl_handler.shの「Configurable section」の「mailto_progs」で指定している文字列をMuttへのパス（ここでは「/usr/local/bin/mutt」）と「VT」という文字列に変更します。「VT」とは同じターミナル上で起動する意味を表す文字列です。

```
mailto_prgs="/usr/local/bin/mutt:VT"
```

次に、標準のままだとURLから「mailto:」が削除されてしまうので、97行目の

```
url="${url#mailto:}"
```

を削除します。これでMuttからmailto URLを使用することができます。

テキストブラウザw3mで表示しているWebページ上のmailto URLに対してMuttを起動するには、ひと工夫が必要です。メーラにURLが渡される前に「mailto:」が削除されてしまうので、「mailto:」を付け加えるラッパスクリプトmailtoprefixを用意します（リスト3）。ここで、変数progには実際にインストールしたMuttのパスを記述してください。そして、w3mのオプション設定の「利用するメーラ」にmailtoprefixを登録してください。これでw3mからMuttを起動させることができるようになります。

●新しいフックコマンド

新しいフックコマンド「message-hook」、「iconv-hook」、「account-hook」が追加されました。「message-hook」と「iconv-hook」に関しては本誌2001年9月号の記事「フック」をご覧ください。「account-hook」に関しては9月号の記事を書いた後にできたフックコマンドです。これらはPOP3やIMAP関連のコマンドなので、記事の後半で紹介します（表5）。

【リスト2】mailto URLのサポート

```
mutt mailto:chris@example.com
mutt "mailto:majordomo@example.com?body=subscribe%20bamboo-l"
mutt "mailto:joe@example.com?cc=bob@example.com&body=hello"
```

【リスト3】mailtoprefix

```
#!/bin/sh
prog="/usr/local/bin/mutt"
url="mailto:$1"
${prog} "${url}"
```

●インストール方法の違い

国際化の関係で注意すべき点がいくつかあります。それはiconvライブラリとワイド文字関数、および日本語まわりの設定です。インストール方法に関しては本誌2001年6月号の特集記事に詳しく書いてありますので、そちらをお読みください。ただし6月号当時のバージョン1.3.17以降で、さらに変更が加えられました。

今までの連載記事の冒頭でも変更があるたびに紹介してきましたが、ここでまとめて紹介しておきましょう。

- `./configure`のオプション「`--with-iconv=DIR`」が「`--with-libiconv-prefix=DIR`」という名前に変わりました。役割自体は変わっていないのでオプションの名前だけ注意してください。
- 日本語パッチによって、デフォルトで日本語の設定が行える`./configure`のオプション「`--enable-default-japanese`」が追加されました。これによって、設定ミスによる妙なメールを送ったり、表示が文字化けしたりする可能性が減ります。
- 日本語パッチの付属文書のインストール用`Makefile`ができました。これにより付属文書のインストールが楽になりました。

POP3

続いて、1.4の大きな変更点の1つであるPOP3の機能について説明を行います。

●./configureのオプション

POP3の機能を使うには、`./configure`のオプションとして「`--enable-pop`」を付ける必要があります。SASL認証を行う場合はCyrus SASLライブラリがインストールされていることを前提として「`--with-sasl`」を付けます。さらに、POP over SSL/TLSを行おうと思ったら、OpenSSLがインストールされているのを前提として「`--with-ssl`」を付けます。後は必要なオプションを付けて`./configure`を実行し、コンパイル後、インストールしてください。

●POP3サーバの指定

POP3サーバとユーザー名を`$pop_host`にリスト4に示すようなPOP URLの形式で記述します。

ユーザー「foo」がPOP3サーバ「pophost」に接続する場合は次のように記述します。

```
set pop_host=pop://foo@pophost
```

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。先ほどの例にパスワード「secret」を記述する場合は次のようになります。

```
set pop_host=pop://foo:secret@pophost
```

POP over SSL/TLSで接続する場合は「`pop:`」の代わりに「`pops:`」と記述します。また、POP URLの形式を使わず、従来通り次のように記述することもできます。

```
set pop_host=pophost
set pop_user=foo
set pop_pass=secret
```

なお、mutt-1.2のころにあった`$pop_port`は、1.4からなくなったため、ポート番号を指定する場合は`$pop_host`のホスト名の後にコロンに続けてポート番号を記述します。

```
set pop_host=pophost:110
```

●認証方式の指定

認証方式を`$pop_authenticators`にコロン区切りのリストで記述します（リスト5）。認証方式は先頭から順に試されます。何も設定しない場合は、サポートしている認証方法の中で、安全なものから順に試されます。認証に失敗したとき下位の方式に移らないようにするには、`$pop_auth_try_all`（デフォルト`yes`）を「`no`」に設定します。

```
set pop_auth_try_all=no
```

認証方式としては、「apop」か「user」しかサポートしていないサーバがほとんどなので、通常は`$pop_authenticators`にはサーバの管理者あるいはISPから指定された方式のみを設定すればよいでしょう。なお、「`user`」は平文認証（パスワードを生のままで流す）を行うことを示します。通信路が安全でない（暗号化されていない）限り、必要もないのに「`user`」を指定するのは危険なので注意しましょう。

●メッセージの取得時の処理

メッセージの取得後、サーバ内のメッセージを削除するかどうかを`$pop_delete`で指定します。デフォルトは「`ask-no`」です。削除する場合は次のようにします。

```
set pop_delete=yes
```

ここでメッセージを削除しないでおくと、次回メッセージを取得したとき、既に取得済みであるかどうかにかかわらず、サーバ内にあるメッセージを全て再び取得することになり、同じメッセージが手元に何個もできるようになるので注意してください。

対応策として、未取得メッセージのみを取得するPOP3のLASTコマンドを使う方法があります。この場合、`$pop_last`を指定します。デフォルトは「`no`」です。使う場合は次のようにします。

```
set pop_last=yes
```

ただし、LASTコマンドはPOP3の標準化の過程の途中で削除されたので、使用できないサーバもあります。その場合には、後で説明する「POPフォルダを利用してローカルに取得しない方法」でも対応できます。また、MuttのPOP機能を使わずに、UIDLを利用した取得済みメッセージの管理を行えるfetchmailなどの外部プログラムを使ってもよいでしょう。

●IPv6

IPv6に対応するには`$use_ipv6`を設定します。デフォルトは「`no`」です。対応させる場合は次のようにします。

```
set use_ipv6=yes
```

なお、この設定変数はIMAPと共通です。

●メッセージの取得

以上の設定で、POPサーバからメッセージを取得できます。取得方法はインデックス画面において「G」（fetch-mail）を入力するだけです。取得したメッセージは`$spoolfile`で設定したスプールフォルダに格納されます。

POPフォルダ

●POPフォルダとは

「POPフォルダ」とは、IMAPのメールボックスのように、ローカルのメールボックスをブラウズするような感覚でPOPサーバ上のメールボックスを閲覧する機能のことです。Muttでは1.4からサポートされることになりました。

メッセージの内容を表示する際にメッセージ

【リスト4】POP URLの形式

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

【リスト5】認証方式の指定

```
set pop_authenticators="cram-md5:apop:user"
```

を取得しにいくため、表示がワンテンポ遅れますが、それを除けばローカルフォルダを扱うような感覚で操作できます。検索や絞り込みなどの操作も行えます。特に、ヘッダ情報のみの検索／絞り込みであれば、瞬時といってもいいくらいの素早さで行えるので、特定のメッセージのみを読みたい場合には役に立つでしょう。また、不必要なメッセージを読まずに削除することもできます。

なお、POPフォルダ内のメッセージは、ローカルには保存されない点に注意してください。もちろん保存やコピー操作を行えば、そのメッセージはローカルに保存されます。

●アクセス方法

POPフォルダへアクセスするには、Muttのコマンドライン(-fオプション)、あるいはメールボックスのパスを指定する場所で、メールボッ

Column

Muttも歩けば棒に当たる

■ Mutt とのなれ初め

ども、はじめして。町田と申します。

Muttの開発には、なーんにも協力できていませんが、Muttのrpmソースパッケージを公開させて頂いているのでお声が掛かりました（分不相応な気もする……）。

Muttを使う前はWindowsやMac OSのメーラをあれこれ試していたのですがどれもしっくりこなく、LinuxでMewも使ってみたものの、当時はそもそも「Emacsの使い方がよく分からん」という状態で挫折していました。

Muttを使い始めたのは1.0の直前ぐらいからで、今やjfbtermとw3mとjedとMuttがあれば、GUIがなくても生きていける体になりました。ちょっと変な人かも（笑）。

ほとんど、Muttを使いたいがためにLinuxを使っているといっても過言ではない（過言かもしれない）です。

■どこでも Mutt

そんなわけで、ほとんどXを使わずに暮らしているのですが、勤務先ではクライアントマシンがWindowsなため、少々ナンギしています。cygwinあたりで使えるのかもしれませんが、コンジョなしなので試していません。

幸いというかなんというか、勤務先のLAN内メールサーバはtelnetでログインできるので（って僕がrootなんですが）メールサーバにログインしてMuttでメールを読み書きしていたりします。

この方法のメリットとしては、まともに日本語が通るtelnetクライアントが入っているマシンであればどこからでもメールを扱えることです。「IMAPを使えばよい」というハナシもなくはないですが、自分の席以外のマシン上のOutlookなどで、うっかり自分のパスワードを保存してしまうとちょっとヤなので、重宝しています。

自分の席以外で作業することが多い人にはお勧めです（そんな人がどれほどいるか知りませんが……）。

また、自宅から勤務先のアカウントに届いたメールを読みたいときのために、ファイヤウォールに穴を開けて、IMAP over SSLプロトコルを通すようにしてあります。当然クライアントはMuttなんですが、残念ながら現在のMuttでは、IMAPで特にキャッシュなどはしていないので、動作は大変遅くなります。sshでサーバにログインして、サーバ上のMuttを動かしたほうが断然速かったりするのがちょっと残念です。

■ Mutt と S-Lang について

Muttで問題なく日本語を扱うには、日本語対応パッチの当たったS-Langライブラリが必要です。しかしこの日本語対応S-Langは、各ディストリビューションによって名前が違ったりなんかしたりします。例えばRed Hat Linux（日本語版）は「slang-j」、Kondara MNU/Linuxは「slang-ja」、Vine LinuxとLASER5 Linux、Turbolinuxは「slang」です。

そんなわけで、せっかく元から日本語対応S-Langが入っているのにもかかわらず、Muttのコンパイルがうまくいかなかったり、わざわざソースtar ballからS-Langを入れ直しておかしくしてしまったりするのは腑に落ちないわい、と感じていました。そこで、ビルド時にちょこっとオプションを付ける程度で各種ディストリビューションに対応できるrpmソースパッケージは作れないものか……ということで作ってしまいました。

■私家版 nosrc.rpm の使い方

以下は、僕が公開しているMuttの私家版rpmパッケージについて説明させていただきます。2001年9月現在、僕の私家版パッケージは「私家版nosrc.rpm集」（[3]）で公開しています。

まず、僕はRed Hat Linux 7Jを使っているため、rpm-4.0でパッケージングしています。rpm-4.0で作成されたパッケージは、rpm-3.0.5以降でないとほどけないようです。

Red Hat Linux 6.2以前、LASER5 Linux 6.2以前、Kondara MNU/Linux 2000以前の各ディストリビューションでは、rpmのアップデートパッケージが存在するようなので、まずはそれを入れてください。

また、Vine Linux 2.0については、Vine Linux 2.1収録のrpm-3.0.6をソースパッケージからリビルドしてインストールすればよさそうです（が、確認していないので責任はとれません）。

次に、お使いのディストリビューションのglibcのバージョンを確認してください。「rpm -q glibc」とやれば、バージョンが表示されます。もしglibcが2.1.3より古い環境の場合で開発版mutt-1.3.xを使用するには、libiconvというライブラリが必要です。libiconvも私家版パッケージにしてあるので、それをお使いいただくか、あきらめて安定版mutt-1.2.5i-jp2をお使いください。もちろん安定版Muttもrpmパッケージにしてあります。

なお、僕が公開しているのはnosrc.rpmです。バイナリパッケージではないので、自分の環境でリビルドする必要があります。また、nosrc.rpmですので、僕の作ったrpmファイルだけでなく、Muttそのもののソースtar ballやmutt-j MLの皆さんによる日本語パッチも必要です。

最近のRPM系ディストリビューションには、RPMパッケージのビルド方法などが詳しく書かれたドキュメントが付属しているようなので、そちらをご参照ください。

基本的には

```
rpm --rebuild mutt-1.x.x.nosrc.rpm
```

のようにやるわけですが、前述のように、ディストリビューションによって日本語対応S-Langの名前が異なります。またglibcが2.2系（2.1.9xも含む）の場合や、前述のlibiconvを使用する場合など、微妙な差異を吸収するため、妙なオプションを付加してもらうことで逃げています。

付加すべきオプションを表Aにまとめました。えーと、なんかやたら複雑ですが、例えば僕が使っているRed Hat Linux 7Jでmutt-1.3.xをビルドするには、リストAのようにします。

【表A】rpmのビルド時に付加するオプション

ディストリビューション名	S-Langの名称	オプション
Vine Linux	slang	--define 'vine 1'
Kondara MNU/Linux	slang-ja	--define 'kondara 1'
LASER5 Linux	slang	--define 'laser5 1'
Red Hat Linux 7J	slang-j	なし
SSLが不要な場合	--define 'disable_ssl 1'	
glibc-2.1.9x以降の場合	--define 'with_wc_funcs 1'	
libiconvを使う場合	--define 'with_iconv 1'	

【リストA】Red Hat Linux 7Jでmutt-1.3.xをビルドする

```
rpm --rebuild --deinfe 'with_wc_funcs 1' mutt-1.3.x.nosrc.rpm
```

■おわりに

と、エラソーなことを言ってますが、最近は自分の常用環境であるRed Hat Linux 7Jをベースにした、ごった煮環境でしか試せていません。そもそもMuttのバージョンアップに付いていけなくなりつつある有様でして（……とほほ）。

そんなわけで、「俺の環境でうまくいかんぞ！」という方は、いきなり開発者の皆さんに言う前に、まず僕のほうにご連絡ください。もちろん動作報告も大歓迎です。

（町田秀企　matchy@happy.email.ne.jp）

クスの代わりにPOP URLを指定するだけです。

例えば、コマンドラインからアクセスするには、読み込むメールボックスを指定する-fオプションを使って次のようにしてMuttを起動させます。

```
mutt -f pop://foo@pop.example.org
```

Muttが起動すると、最下行に次のようなメッセージを表示しながら、POPサーバからメッセージヘッダを取得します。

```
メッセージヘッダ取得中 ... [7/22]
```

取得し終わると、インデックス画面にメッセージの一覧が表示されます。また、インデックス画面でcキー（change-folder）を入力して他のメールボックスに移るときに、リスト6のようにPOP URLを入力してもPOPフォルダを開くことができます。

後はコマンドラインでの指定と同じように、メッセージヘッダを取得後、インデックス画面にメッセージの一覧が表示されます。

●接続の処理

POPフォルダを開いているときの接続の処理に関する設定変数は次の2つがあります。

POPサーバに新着メッセージがないかどうかを確認する間隔（秒数）を$pop_check intervalに指定します。デフォルトは60秒です。5分（300秒）間隔にする場合は、次のように設定します。

```
set pop_checkinterval=300
```

サーバとの接続が切れたときに再接続するかどうかは$pop_reconnectで指定します。デフォルトは「ask-yes」です。再接続を必ずする場合は次のように設定します。

```
set pop_reconnect=yes
```

account-hook

●account-hookの概要

account-hookはfolder-hookと似たような働きをするフックコマンドです。folder-hookがローカルのフォルダを扱うのに対して、account-hookはPOPフォルダやIMAPフォルダを扱います。学校や会社やISPなど複数のアカウントを持っている場合に、それぞれのアカウントに応じた設定を一括して行えるため重宝するでしょう。設定は次の書式で行います。

```
account-hook regexp command
```

読み込むフォルダがPOP URLやIMAP URLの形式で記述してあり、それが正規表現regexpに一致するときにcommandを実行し、POP/IMAPフォルダを開きます。コマンドが実行されるタイミングはサーバに接続する前です。

●account-hookの設定例

では、表6に示す2つのアカウントがある場合の設定例を見ていきましょう。

2つのアカウントともPOPフォルダとして接続するには、リスト7の2行目、3行目のように、それぞれPOPフォルダごとに$pop_userと$pop_passを設定します。関連する他の設定変数もここで設定します。

ユーザー名を$pop_userで設定せずに、POP URLの中に記述しても構いませんが、POP URLを入力するときには、短い方が楽なのでこの例のようにした方がいいでしょう。

最後にfolder-hookで行うようにデフォルト値を設定します。account-hookで指定する設定変数を解除するものを記述するといいでしょう。このデフォルト値はどのaccount-hookコマンドよりも先に記述する必要があります。

なお、pophost1からはメッセージの取得を行い、pophost2はPOPフォルダとして接続する場合には、pophost1用に$pop_hostの設定をPOP URLの形式で記述します（リスト8）。

【表6】account-hookのサンプルユーザー

POPサーバ	ユーザー名	パスワード
pophost1	foo	secret1
pophost2	bar	secret2

【リスト6】インデックス画面からPOPフォルダを開く

```
メールボックスをオープン: pop://foo@pop.example.org/
```

【リスト7】account-hookの設定（POPフォルダを使う）

```
account-hook .  'unset pop_user pop_pass'
account-hook pop://pophost1/ 'set pop_user=foo pop_pass=secret1'
account-hook pop://pophost2/ 'set pop_user=bar pop_pass=secret2'
```

【リスト8】account-hookの設定（メッセージを取得する場合）

```
set pop_host=pop://foo:secret1@pophost1/
account-hook .  'unset pop_user pop_pass'
account-hook pop://pophost2/ 'set pop_user=bar pop_pass=secret2'
```

ユーザー名とパスワードもここで記述します。pophost2に関しては先のaccount-hookの例と同じように記述します。

最後に

この連載が始まってから半年が経ちました。1.4も出ることだし、切りがいいので、ちょっと振り返ってみます。この連載は、TIPSを織り混ぜながらMuttの膨大な設定*3をテーマごとに分かりやすくまとめていこうという主旨で記事を書いてきました。マニュアルを読めば済むようなこともたくさん書いているので、全てを熟知している人*4にはつまらないかもしれません。しかし、マニュアルに書いてあることでも単に読んでいるだけではよく分からず、具体的な事例を読んだり、実際に使ってみないと分からないことが多いようです。その辺をフォローできていたらいいなと思っています。

次回以降の大きなテーマとしてはIMAPと通信路の暗号化（ssh/ssl）について取り上げるつもりです。どちらも重いテーマですし、試験的に使ったことがないわけではないですが、正直言って普段はあまり使っていない機能だったりします。そのため墓穴を掘らないように注意して書いていこうと思っています。

Resource

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

[2] GNU libiconv
http://www.gnu.org/software/libiconv/

[3] 町田氏による「私家版nosrc.rpm集」
http://www.asahi-net.or.jp/%7EYJ7H-MCD/rpms/

＊3　日本語版での拡張機能も含めれば250個くらいの設定変数があります。

＊4　果たしてどれだけいるのだろうか？　開発版を追っ掛けている筆者自身ですら全貌は把握しきれていません……。