



## 第2回 パターン

今回は、Muttの大きな特徴の1つである「パターン」について説明します。

本題に入る前に、Muttの最新状況についてお知らせします。執筆時点での最新バージョンは「1.3.18i」ですが、本誌が出るころには「1.3.19i」が出ているはずです。大きな変更点は、「Content-Type: text/plain」の「format」パラメータに対応したこと。しかし、experimental(実験的)な状態なので、この機能を使う場合は注意が必要です。

## M パターンの基礎

### パターンの種類

Muttで使うことのできるパターンを表1に示します。この表の中で、EXPR、USER、ID、SUBJECTは正規表現であり、MIN、MAXは数値、あるいは日付です。これは「MIN-」`「-MAX」`、「>MIN」`「<MAX」`のような記述もできます。

日付の記述方法は「日/月/年」です。例えば2001年5月24日であれば「24/5/2001」となります。年は2桁の形式の「01」でも構いません。月と年は省略可能で、省略した場合は、現在の月と年になります。先の例で年を省略すると「24/5」となります。

期間の記述方法ですが、通常は「24/5-23/6」のように記述しますが、指定した日に対する「誤差」として指定することもできます。この場合は、正、負、正負を示す「+」`「-」`「\*」と、年、月、週、日を示す「y」`「m」`「w」`「d」`を使います。例えば、5月24日から1ヵ月後までの範囲であれば、「24/5+1m」と表すことができます。また、現在からのオフセットを指定して「その日以前」`「その日以降」`「ちょうどその日」を示す「>」`「<」`「=」を使うこともできます。例えば「2週間前の日以

降」を示す場合は「<2w」と表せます。

### パターンの記述方法

パターンの記述方法は、表のパターンの式を実際の正規表現の文字列に置き換えるだけです。例えば、Subjectに「mutt」を含むという指定は「~s mutt」になります。文字列の間に空白文字が含まれる場合は、文字列を二重引用符「」で囲み、「~s "foo bar"」のようにします。

複数のパターンを使う場合は、表2の論理演算子を用います。論理積(AND)の場合は、単に「~f foo ~s mutt」のように並べるだけです。論理和(OR)の場合は「|」を使い「~f foo | ~s mutt」のようになります。論理否定(NOT)の場合は、否定するパターンの前に「!」を置き「!~s mutt」ようになります。パターンをグループ化する場合は「()」を使い「~f foo | (~f bar ~s mutt)」ようになります。

メールアドレス関係のパターンでは、例えば、Toにfooがあるメッセージを探したい場合に、パターンとして「~t foo」を指定しますが、これは、「To: foo, bar」のように複数のアドレスが書いてある場合でも一致します。これをfooだけの場合にのみ一致させたい場合は、パターンの前に「^」を置き、「^~t foo」のようにします。そうすると、複数のアドレスには一致せずに「To: foo」の場合だけに一致します。

## M パターンの利用

### 検索

検索を行う場合は「/」(search)あるいは「Esc /」(search-

reverse)を入力します。

インデックスの場合は、検索パターンを入力して一致したものが見付かると、カーソルがそのメッセージの位置まで移動します。「n (search-next)を入力すると、次の検索結果に移動します。前の検索結果に戻るsearch-oppositeに対して、デフォルトではキーは割り当てられていません。

検索パターンとして、パターンではなく正規表現を入力した場合は、\$simple\_searchで設定したパターンに展開されます。このパターンのデフォルトは「~f %s | ~s %s」です。

インデックス以外で検索を行う場合は、パターンではなく正規表現を入力します。一致した文字列が見付かると色付けが行なわれ、ページの先頭行にその文字列のある行が移動します。後の操作はインデックスと同じです。

なお、ラインエディタに過去に入力した文字列が履歴として残っていて、上下のカーソルキーで参照できます。しかし、他のメニューの履歴は使えないのでご注意ください。

## タグ付け、削除、表示の絞り込み

パターンマッチングの結果に対して動く機能を表3に示します。この中で特に強力なのがtag-patternです。他の機能と組み合わせることにより非常に役立つことでしょう。

次の例は、nameという名前のメーリングリストをname-listというフォルダに移動する例です。

- (1) インデックスにおいて「T (tag-pattern)を入力
- (2) パターンとして「~h^X-ML-Name: name」を入力後、Enterキーを押す
- (3) ; (tag-prefix)の入力に続けて「s (save)を入力
- (4) 保存先として「=name-list」を入力後、Enterキーを押す
- (5) 確認のメッセージが出るので「y」を入力

このような操作を毎回入力するのは大変なので、次のようなマクロを使うと便利です。

```
macro index \eT "T~h\"^X-ML-Name: name\"n"
macro index \eS ";s=name-list\ny"
```

1つ目のマクロでタグ付けを行ない、2つ目のマクロで保存を行ないます。マクロを2つに分けた理由は、(1) (2)の操作で

表2 演算子

論理	演算子
論理積(AND)	なし(条件を続けて記述すればよい)
論理和(OR)	
論理否定(NOT)	!
論理グループ化	()

表1 パターン

メールアドレス関係のパターン	
~t USER	ToにUSERを含むメッセージ
~c USER	CcにUSERを含むメッセージ
~C EXPR	ToがCcにEXPRを含むメッセージ
~f USER	FromにUSERを含むメッセージ
~e EXPR	SenderにEXPRを含むメッセージ
~L EXPR	From、Sender、To、CcにEXPRを含むメッセージ
~p	あなた宛のメッセージ
~P	あなたから送られたメッセージ
~l	指定したメーリングリスト宛のメッセージ
特定のヘッダ・フィールド関係のパターン	
~s SUBJECT	SubjectにSUBJECTを含むメッセージ
~i ID	Message-IDがIDに一致するメッセージ
~x EXPR	ReferencesにEXPRを含むメッセージ
~d MIN-MAX	Dateがその日付の範囲にあるメッセージ
~r MIN-MAX	Receivedがその日付の範囲にあるメッセージ
~E	Expires <sup>*1</sup> で期限切れになったメッセージ
~S	Supersedes <sup>*2</sup> で置き換えられたメッセージ
~y EXPR	X-Label <sup>*3</sup> にEXPRを含むメッセージ
ステータス関係のパターン	
~D	削除フラグの付いたメッセージ
~F	重要フラグの付いたメッセージ
~N	新着メッセージ
~O	古い未読メッセージ
~U	未読メッセージ
~R	既読メッセージ
~Q	返信済みメッセージ
~T	タグ付きメッセージ
~g	PGP署名されたメッセージ
~G	PGP暗号化されたメッセージ
~k	PGP鍵を含むメッセージ
その他のパターン	
~A	すべてのメッセージ
~h EXPR	ヘッダにEXPRを含むメッセージ
~b EXPR	ボディにEXPRを含むメッセージ
~B EXPR	メッセージにEXPRを含むメッセージ
~v	折り畳んだスレッドに含まれるメッセージ
~m MIN-MAX	番号がMINからMAXの範囲にあるメッセージ
~n MIN-MAX	スコアがMINからMAXの範囲にあるメッセージ
~z MIN-MAX	サイズがMINからMAXの範囲にあるメッセージ
<p>*1 そのメッセージの有効期限の日時を記述するヘッダフィールド。RFC 2156およびRFC 1036を参照。</p> <p>*2 そのメッセージが置き換えるメッセージのMessage-IDを記述するヘッダフィールド。RFC 2156およびRFC 1036の改定案<sup>1</sup> News Article Format (draft-ietf-usefor-article-04.txt)を参照。</p> <p>*3 ユーザーが独自に設定できるヘッダフィールド。どのRFCにも規定されていない。</p>	

表3 パターンマッチングの結果に対して動く機能

キー	機能名	動作
D	delete-pattern	一致したメッセージを削除する
U	undelete-pattern	一致したメッセージの削除を取り消す
T	tag-pattern	一致したメッセージにタグを付ける
^T	untag-pattern	一致したメッセージのタグを外す
l	limit	一致したメッセージのみ表示する

タグ付けされたメッセージがなかった場合に、(3)のtag-prefixで失敗するからです。

次に、limitの使用例を示します。「foo@example.orgさんから送られてきたメッセージに、確かそういう話があったので読み返してみたいけど、どの辺にあったのかな?」という場合は、fooさんのメッセージだけを表示させてから作業した方が便利です。この場合は、パターンとして送信者を表す「~f」を使って次のような操作を行います。

- (1) `~f` (limit) を入力
- (2) パターンとして「~f foo@example.org」を入力後、Enterキーを押す

さらに「2月中旬のメッセージだったはず」という場合は、先ほどのパターンに「~d 15/2\*5d (2月15日±5日)」を付け加えます。追加するときは履歴の機能を有効に使ってください。

逆にlimitを解除するには、パターンとして、全メッセージを表す「~A」を使います。手順としては次の通りになります。

- (1) `~f` (limit) を入力
- (2) パターンとして「~A」を入力後、Enterキーを押す

なお、現在設定しているパターンは「Esc l (show-limit)」で確認できます。

## M パターンの応用

### マーク

メールを読み進めているときに、後で読み返したり返信したりするために印を付けたいことがあります。この場合は「F」(flag-message)を入力して、重要フラグ「!」を付けます。こうすると、後で読むときにsearchやlimitでパターンとして「~F」を指定すればよいだけです。フラグを外すときは、もう一度「F」を入力します。

さらに、さまざまな種類の印を付けたいと思うかもしれませんが、特別な意味のないフラグが1つしかないためできません。しかし、ちょっと裏技的ですが、「X-Label」フィールドを使えば同じようなことができます。ただし、メッセージを編集してヘッダにX-Labelフィールドを挿入するというオリジナルのメッセージに手を加える作業が生じるため、取り扱いには注意が必要です。

作業例を挙げると、「このメッセージはMuttに関する重要な



画面1 X-Labelフィールドの内容をインデックス表示する

記事なので後で簡単に見付けられるようにしたい」と思う場合、「e (edit)」を入力してエディタを起動させ、このメッセージを編集して「X-Label: mutt」をヘッダに追加します。エディタでX-Labelフィールドを挿入するようなマクロを組めば作業は楽になると思います。このメッセージを後から探すときはパターン「~y mutt」を使います。

また、\$index\_formatで「%y」や「%Y」を指定すると、X-Labelフィールドの内容をインデックスに表示することも可能です。例えば、

```
set index_format="%4C %Z %-5.5y %[%b %d] %-15.15L %s"
```

のような設定を行うと、フラグの後にX-Labelの内容を表示できます(画面1)。

## M 最後に

パターンを使う機能で今回説明しなかったものには「スコア」と「フック系コマンド」があります。この2つに関しては次回以降にテーマとして取り上げますのでお待ちください。

### おまけのTIPS

mailboxesコマンドの設定の省力化です。

```
mailboxes "!" 'echo $HOME/Mail/*'
```

R E S O U R C E

[1] Mutt Japanese Edition  
<http://www.emallab.org/mutt/>

## Muttも歩けば棒に当たる

mutt-jメーリングリスト管理人・太田氏の場合

## 使うようになったきっかけ

mutt-jメーリングリスト ML 管理人の太田です。ML管理をしている割には、Muttは比較的最近使い出した方です。なぜかという、elmをずっと使っていて馴染んでしまっていたからです。ちなみにelmは、MuttのモデルにもなったMUAです。その日本語版を開発していて愛着があったというのも事実です。しかし、elm(正確にはelm2.4ME+)の機能の拡張に限界を感じたのと、本家のバージョンアップについていけなくなり、代替のMUAとして利用し始めたのが、Muttを使い始めたきっかけです。

もちろん、elm日本語版の開発時にもMuttの存在は知っていました。elm2.4MEは、もともとMuttの最初の開発者Michael R. Elkinsさんが開発し始めたものですし、ドイツでelmのページを公開しているSven Guckesさんも、「Mutt vs. elm」という比較をしていて、Muttに軍配を上げていました。ただ、私がelmを開発していたころのMut(0.5とか0.6の時代)は、機能的には今一つだったということと、Muttが将来的にメジャーになるかが分からなかったことがあって、切り替えるには至らなかったのです。

Muttを使い始めたのはバージョン1.0台からですが、そのころ、菊谷さんと吉田さんを始めとする日本語版開発スタッフが開発していたものは、最低限の部分のみ、メッセージが日本語化されていました。しかし1.3版台になって、メッセージが大幅に追加、変更になったため、日本語化率が下がってしまいました。そこで、使うだけでは悪いので、elmの日本語化(当然メッセージの日本語化も含む)の経験を生かそうと、メッセージカタログの翻訳に手を出した次第です。

幸いにもMLの方々の協力が得られまして、現在はほぼ安定した状態になっています。語学力がないのでお気に召さないところもあるかと思いますが、何か問題がありましたら、ぜひMLの方にご連絡ください。

## MLの話

Muttの日本語版の開発には、mutt-j MLが大きな役割を果たしています。簡単にMLとMLサーバの話をししましょう。

mutt-jのMLサーバはfmlで運用しています。回線はOCNエコノミーを使っています。今となっては遅いのですが、運用し始めた当時は、128KBPSはとても速く感じました。ちなみに、OCNエコノミーの開設は、同一電話局管内で2番目でした。

mutt-j MLへの参加については「mutt-j-ctl@ribbon.or.jp」宛に、本文に「#subscribe」という行を含むメールを送信してください。折り返し登録方法のメールが送られてきます。

MLは基本的にクローズドな運用をしています。過去メールの参照と全文検索を提供していて、参加/離脱は自由ですが、参加者以外からの投稿/閲覧はできません。なぜクローズドなのかというと、SPAM防止のためです。最近、Webページからメールアドレスをかき集めてくることもあるので、残念ながら防御せざるを得ないのです。

MLの内容は、設定の話(使い方)が最近が多いようです。これは開発が一段落したからでしょう。ただ、新しいバージョンなどが出ると、開発関係の投稿が増えます。だ、流量としてはそれほど多くはなく、1日当たり0通~4、5通といったところです。

## MLサーバ

実は、現行のMLサーバマシンは4代目です。

1台目はEPSONのデスクトップタイプでした。秋葉原で展示品を買ってきて使っていました。Pentium-75MHzを120MHzに差し替え、その後、WinChip2で運用していました。しかし、マザーボードのメモリ回りがコケ、メモリを取り替えても駄目だったので(OSが突然パニックする)廃棄しました。

2台目はSPARCStation 5(実際にはNetra-i)でした。一応新品です。しかし、ファンがうるさいので1年ほど前に運用停止し、現在は他のところで第二の人生を歩んでいます。

3台目はショップブランドのミニタワーでした。某社のワークステーションと同じ筐体を使っています。12センチファン+IBMの5400回転のHDD、QuantumのFireball LCTシリーズを使って静寂性を重視しました。この辺までSamba-JPと共存していました。

4台目が現行マシンです。去年の冬から運用しています。これは工業用のマザーボードに手作り筐体で作成したものです。そのためファンレスで、ディスクの音しかしません。消費電力は、何とたったの10Wです。Pentium-200MHz相当のCPUなので、それほどパワーはないのですが、メンバが多くないIMLのサポート+ ならば問題はありません。そして、MUAとしてMuttを使えばパワーがなくても、とても快適です。

(太田俊哉)

## 番外編 メッセージカタログの翻訳

太田俊哉

ここでは、どのようにしてメッセージカタログの翻訳をしていったかについて紹介します。

### 翻訳の方針

「方針」と書いても、それほど明確な決まりを作ったわけではありません。用語などの対訳表を準備するというような大掛かりなことをしたわけでもありません。ただ、以下のことは考えて翻訳を進めました。

- ・菊谷さんの訳はできるだけ生かす

もともとMuttのメッセージカタログは菊谷誠さんが訳されています。そこで、使えるものは使おう、というわけで、生かせるものは生かすことにしました。

- ・できるだけ簡潔に

詳しくはこの後で説明しますが、表示幅の制限がありますので、あまり長い訳は不可能なのです。

- ・カタカナ語を使わないように

これは、個人的な好みもありますし、カタカナ語を使うと長くなる、というもあります(例: Sort xソート/整列)。

そういえば、算譜なんていう言い方は、最近はしないですね。

### 実作業

Muttはgettextを使ってメッセージカタログを処理しています。そのため、まずgettextのマニュアルをよく読むところから作業は始まります。

翻訳を1から始める場合には、xgettextでソースからメッセージを抜き出します。しかし、実際にはすでに抜き出したmutt.potというファイルがありますので、このファイルをベースに翻訳を進めました。

分からない、いい訳が思い付かないものはそのままにしました(苦笑)。ただ、一部未訳のものをMLで紹介すると、たちどころに良い訳を頂くことができました。この辺はMLの醍醐味ですね。

また、日本語と英語で語の順番が異なるものがあります。もともとgettextは、その対応がなされていますが、メッセージによっては、そのことがよく分からないものもあります。そのようなものは取りあえず訳して、動作を見ながら編集し直すことになります。

昔の訳は、生かす方針を取りましたので、統一感がないところもあります。例えば、「q」で終了するときば「Mutt抜けます?」になっています。これは昔のままの例です。でも、プログラムの美徳の「不精」という観点以外にも、作業工数の点からも、統一感を出す所までできないのが正直なところです。

翻訳の工数ですが、全部合わせて10数時間程度ではないかと思います。それから先は、コメントをもらって適宜直す、というパターンでした。

### ポイント

Muttのメッセージカタログの翻訳は、画面の幅の制限があるので、あまり冗長な翻訳ができないという若干の制限があります。制限に引っかかるのはあまりないのですが、例えば、リスト1のようなメッセージは工夫しないとみ出てしまいます。

まず、「(d)ate」は「(日)時」にはできないので、「d)日時」にしました。「(」を省くことで1byte節約します。次に、縮められるものは可能な限り縮めます。例えば「frm」は「送信者」ですが、これは「送者」とします。英文も「From」「frm」ですが、この程度は認められると考えました。行末の「?」や「:」も、なくても分かるのでカットします。最終的にはリスト2のようになりました。少し長くなりましたが、この辺が限界です。

#### リスト1 長いメッセージの例(原文)

```
Sort (d)ate/(f)rm/(r)ecv/(s)ubj/t(o)/(t)hread/(u)nsort/si(z)e/s(c)ore? :
```

#### リスト2 長いメッセージの例(翻訳文)

```
整列 d)日時/f)送者/r)着順/s)題/o)送信先/t)スレッド/u)非整列/z)サイズ/c)得点
```

そのほかに、訳を短くするための工夫として、

- ・分かち書きをしない(日本語では当たり前だけど)
- ・体言止め
- ・不要な分離文字(「:」など)は削除
- ・文末の「。」も付けない
- ・である調

などを心がけました。

また、外部に対訳表があるものはそれを参照しました。PGPの訳は、PGPの翻訳例を載せているIIJ技術研究所の「(仮)日本の公式 PGP ホームページ」[1]を参照した修正案をいただきました。もちろんそのまま吸収しました(笑)。

ところでMuttには、Bounce Messageという機能があります。これは、メッセージ(=メール本文+ヘッダ)に何も加工せず、別の人にそっくりそのまま転送する機能です。受け取った人は、一見、おおもとの人から直接メールが来たように見えます。この機能は、以前に訳したelmでは「中継」としてありますが、Muttでは「再送」としました。このように、同じような機能でも訳を変えたものもあります。

一方、複数形の取り扱いですが、これはMLの人の意見もあり、単数形と複数形を同じ訳にしています。

## やりにくい訳

オリジナルのメッセージも、短く表示されるようにしてあるので、訳の意味が分かりづらいものもままあります。その辺はMLの参加者の方が訳を提供してくれましたので、ほとんどそのまま採用しました。ただ、短くするという観点からは、やりにくいものもいくつかありました。

- ・chdir

英文だと5文字ですが、日本語だと「ディレクトリ移動」と長くなってしまいます。これは縮めるのが難しいので長いままの訳を使っています。

- ・no files

「noなんとか」という英語は、どうしても日本語にすると長くなってしまいます。「0個のファイル」とは言いませんから。「ファイルが1つもない」とせざるを得ません。

- ・Reverse Sort

順当に訳すと「逆順の整理」になりますが、これは「逆整理」としてしまいました。たぶんこのような言い方はないのでしょうが、「Rev-Sort」もありましたので、許容範囲と考えました。

- ・Capitalize

もともと日本語にはCapitalizeという概念がないので、「先

頭文字の大文字化」という訳になってしまいました。

- ・subscribe

これは「購読」としました。日本語の方がかなり短くなる例です。

## デバッグ

メッセージカタログのデバッグというのは、基本的にはありません。作成されたja.poメッセージカタログファイルをgettextでコンパイルし、LC\_MESSAGESで定義されているディレクトリに放り込めば終わりだからです。まあ、使ってみて、気に入らないとか、メッセージの順が変だとかいうものは、随時、直せば済みます。

ただ、システムによっては/usr/local/share/localeにあるメッセージカタログ格納用のディレクトリが「ja」ではないものもあります。NECのUX/4800シリーズ(EWS4800、UP4800)がその1つです。UX/4800シリーズでは、標準のlocaleが「japan」になっているのです。この場合は、jaとjapanをシンボリックリンクすれば問題は解決します。

また、xgettextですべてのメッセージを抜き出してみて分かったのですが、メッセージカタログとして抜き出されていないメッセージもまだいくつか残っています。簡単なエラーメッセージのようなものがgettext化から落ちているのです。これは、メッセージカタログ化のバグと言えますね。まあ、暇を見て、パッチを送る予定ではいます。もしもそのようなメッセージに気が付かれましたら、ぜひmutt-j MLや本家のMLに投稿してください。

## 今後の予定

日本語版メッセージカタログは、バージョン1.3.16から本家ソースコードの中に登録されました。よって、日本語化パッチの中からはなくなりました。その分、本体の大きさが大きくなってしまいましたが、多国語化を図るプログラムでは仕方ないですね。

今後は、新しいバージョンがリリースされるごとに、できるだけ早めに差分を出していこうと思います。

現在バージョン1.3.19に対するメッセージカタログの作成を行なっているところです。この記事を書いた雑誌が出る頃には作成する予定です。

R E S O U R C E

[1] IIJ技術研究所 (仮)日本の公式 PGP ホームページ  
http://pgp.iiijlab.net/