Emacsの使い方

牧之内研 修士2年 峯 肇史

2002/4/15

1　リージョン指定

指定したい領域の一方の隅に目印をつけた後、もう一方の隅にカーソルを移動させます。目印をつけた場所とカーソルとの間が領域 (リージョン) です。Emacs では目印を「マーク」と呼び、目印を設定することを「マークする」といいます。マークするには、カーソルをマークする場所まで移動して、そこで”C-Space”とタイプします。マークが設定されるとミニバッファに、

```
Mark set
```

と表示されます。

カーソル位置の文字全てが領域として指定されるわけではありません。カーソルの4角形の左辺を基準として領域が指定されます。この基準となる部分をポイントと呼びます。リージョンはマークしたときのポイントから今のカーソルのポイントまでになります。

2　削除

Emacs には、文字をバッファから取り除くコマンドとして delete と kill があります。2つともバッファから文字を取り除く機能は同じですが、delete は消去、kill は切り取りの意味合いがあります。delete は文字の消去のみですが、kill とつくコマンドは、文字をバッファから取り除く際に取り除いたテキストを kill-ring と呼ばれる領域に保存しています。kill-ring に保存されたテキストは、別のコマンドによって取り出すことができますが、delete で消去されたテキストは undo などで、操作自体の取消しをしないかぎり復活しません。

表 1: 削除キー

キー	コマンド	機能
C-d	delete-char	1文字削除
M-d	kill-word	1単語カット
Del	delete-backward-char	1文字後ろ向きに削除
M-Del	backward-kill-word	1単語後ろ向きにカット
C-x Del	backward-kill-sentence	1センテンス後ろ向きにカット
C-k	kill-line	カーソル行から行末までカット
M-k	kill-sentence	カーソル行から1センテンスカット

2.1 領域の削除

領域指定した状態で”C-w”をタイプしてみます。すると領域指定された箇所が削除されます。
”C-w”は指定領域を削除するコマンド (kill-region) です。

3 複写

領域をバッファから削除せずにキルリングへ格納するには”M-w”を使います (kill-ring-save)。このコマンドもマークとカーソルで領域を指定します。
また、マウスの左ボタンドラッグで複写もできます。ドラッグした範囲の色が変わります。

4 張り付け

キルリングに取り込まれたテキストを取り出すにはヤンク (yank) を使います。ヤンクコマンドは”C-y”に割り当てられています。
”C-y”をタイプするとキルリングに格納されているテキストをカーソル位置に挿入します。
さらに、その前にキルリングに記憶されたテキストも順次取り出すことができます。古い削除テキストを取り出すには、”C-y”コマンドの直後に”M-y”をタイプします (yank-pop)。”M-y”を押すたびに、取り出されるテキストが順々に古い削除テキストに入れ換わります。
また、マウスの中ボタンクリックでマウスカーソルがある位置にテキストを挿入することも可能です。

5 取り消し

間違えて削除をしてしまった場合などに、前の状態に戻りたい場合があります。”C-/”を入力すると間違えて操作した前の状態に戻ることができます。これをアンドゥ(操作の取り消し) といいます。アンドゥは何度も繰り返せます (繰り返すことができる回数に制限はあります)。

6 編集モード

一言でテキストと言っても、テキストはその用途に応じてさまざまなスタイルがあります。例えば、プログラムコードを書くときには、プログラムの構造がわかりやすいように文法に応じてインデントを行います。テキストはその種類や用途によってスタイルがまったく異なります。
Emacs は文章の特徴をふまえて編集できる編集環境を提供します。Emacs は「メジャーモード」と「マイナーモード」という 2 つのモードを使っており、現在のメジャーモードとマイナーモードはモードラインに表示され、一目で確認できます。

6.1 メジャーモード

メジャーモードは主モードとも呼ばれます。プログラミング言語の編集や、特定の種類の文章操作や編集に関する知識でもあります。専用のコマンドとキー設定や入力の補助を提供します。

例えば、C 言語編集用のメジャーモード「c-mode」は C 言語スタイルのインデントを自動で行います。メジャーモードは各バッファに一つだけ設定でき、複数のメジャーモードを設定することはできません。

メジャーモードには以下のようなものがあります (他にもある)。

- fundamental-mode
  特に機能を持っていないモード
- text-mode
  通常の文章を編集するためのモード
- c-mode c++-mode
  C 言語および、C++言語のプログラムソースコードを編集するときに便利なモード

6.1.1 モードのロード

新しいバッファは特に何も指定しなければ、何も機能を持っていない「fundamental-mode」になります。メジャーモードを設定するにはそのモードをロードします。モードのロードは”M-x”で、コマンドを直接実行します。例えば、C モードを設定するには、

```
% M-x c-mode
```

とタイプします。

新しいバッファを作成したりファイルを読みこんだときは、バッファ名の拡張子から自動的に適切なメジャーモードをロードします。この設定は.emacs に記述できます。/usr/local/share/skel/.emacs-skel には以下のように記述されています。

```
(setq auto-mode-alist
      (append '(("\\.C$"  . c++-mode)
                ("\\.cc$" . c++-mode)
                ("\\.c$"  . c-mode)
                ("\\.h$"  . c++-mode)
                ) auto-mode-alist))
```

6.2 罫線

Emacs 上で罫線を使いたいときは”@けいせん”で変換すると色々な向きの罫線が表示されるので、使いたいものを選んでください。あと、線が太くなった”@ふとけい”というのもあります。

罫線を書くためのモードを追加することもできます。/u/mine/pc/M2/rinkou/emacs/my-keisen-mule.el を適当なところにとってきてミニバッファで

```
M-x load-file
```
あなたの保存したディレクトリ/my-keisen-mule.el
```
M-x keisen-mode
```

これでカーソルキーで罫線が引けるようになります。詳しい使い方については my-keisen-mule.el の中身を眺めて見てください。

6.3 マイナーモード

マイナーモードはメジャーモードに対して、各種の機能を付加します。これらのモードは常に設定されているというわけではありません。マイナーモードを設定するには”M-x”を使って直接マイナーモードの関数を呼び出します。上書きモードを設定するときは、

```
% M-x overwrite-mode
```

とタイプします。モードが設定されているなら解除し、設定されていない場合は設定します。マイナーモードはメジャーモードと異なり、一つのバッファに複数設定できます。

マイナーモードには以下のようなものがあります (他にもある)。

- overwrite-mode
  通常 Emacs では、カーソルの位置に文字を挿入します。上書きモードでは入力された文字は挿入されず上書きされます。
- auto-fill-mode
  自動的に文章を一定の位置で改行します。自動整形モードを使うと行末を自動的に改行します。改行する桁は fill-column 変数で指定した文字数です。
- line-number-mode
  モードラインに現在の行を表示します。

6.3.1 フィルカラムの設定

fill-column 変数はデフォルトで 70(72?) に設定されています。1 行の長さを変更するには改行したいところまでカーソルを移動し”C-u C-x f”と入力します。またはミニバッファから

```
M-x set-variable
fill-column
数字
```

数字を設定しただけだとすでに書かれている部分は整形されないので、以下のコマンドをつかいます。

M-q(fill-paragraph) パラグラフを整形し直す

M-x fill-region 領域指定した部分を整形し直す

7 ispell

英語で文章を記述する際に気をつけなければならないのが、スペルミスです。UNIX のコマンドにスペルミスを調べるスペルチェッカと呼ばれるツールがあります。スペルチェッカの代表的なコマンドの一つに”ispell”というコマンドがあります。ispell はテキストを読み込み、辞書を調べて辞書にない単語を検出します。また、検出された各単語に対し、どの単語を間違えたのかを推測して修正候補を出力し、対話的に単語を修正することもできます。ここでは UNIX のコマンドとして使用する際については触れません。

表 2: ispell のコマンド

コマンド	機能
M-x ispell-buffer	バッファ内のテキストのスペルチェックを対話的に行う
M-x ispell-region	リージョン内のスペルチェックを対話的に行う
M-x ispell-word	カーソルの位置のスペルチェックを行う

表 3: ispell の操作

Space	今回のみこの単語を受け付ける
<number>	番号で指定された修正候補に置き換える
R)eplace	入力された修正候補に置き換える
A)ccept	今回のセッションを通してこの単語を受け付ける
I)nsert	この単語を受け付け、個人的な辞書に登録する
L)ookup	システム辞書の中を探す
U)ncap	この単語を受け付け、個人的な辞書に小文字に変換した単語を登録する
Q)uit	即座に終了する。修正を行わない
e(X)it	このファイルの残りの部分の修正は行わずに終了する
?	ヘルプを表示する

Emacs では表 2 のようなスペルチェックのコマンドがあります (他にもあります)。

ispell-buffer と ispell-region は、スペルチェックの対象がバッファ全体かリージョンかの違いがあるだけで基本的な操作方法は同じです。

コマンド ispell-buffer を実行するとスペルチェックが開始され、バッファ内のスペルミスの検出を開始します。スペルミスを検出すると、ユーザに正しい単語の候補を提示し、修正を求めます。スペルミスを検出すると、表 3 のような操作によって単語を修正したり、そのまま認めたりすることができます。

ispell が使えない人はまずインストールします。

```
cd /usr/ports/textproc/ispell/
make
make install
```

8 本日の課題

適当な英語の文章を書いて (写して)ispell を使ってスペルチェックしてみる。

9 Reference

- Emacs&Mule Manual&Reference
- Mule ハンドブック