Vimの使い方 (これだけは、知らないと)

2004/06/05
2004/06/20

Vimの使い方です。基本動作はわかっている人向けです。

単語の書き換え

エディタを使うときは、文字を1つずつ書き換えるなんて考えてはいけません。単語単位に書き換えることを考えましょう。

sample.txt
first line first line third line last line

↑を
↓こうしたい場合

sample.txt
first line **second** line third line last line

2行目のfirstのfにカーソルを移動して、
cw
と押します。それから、
second<esc>
ですね。(<esc>はエスケープキーを押してください)

cwは、単語を削除して入力モードに入る

って意味になります。Deleteを5回押すって考えちゃだめですよ。この場合は、単語を書き換えるんだから「cw」です。1文字ずつ変えるんじゃなくて、単語を変えるんだから単語を削除して、新しい単語を入力する。当たり前のことですよね。

ここで紹介した操作は、Windowsのエディタで言うと、以下の操作に当たります
Ctrl + Delete

w 単語単位に移動

カーソルを移動するときにも、単語単位で考えましょう。

sample.txt
/* * Check for we support the binfmt * if we do, return the node, else NULL * locking is done in load_misc_binary */

↑こんなテキストのforをifに書き換えるためにforまで移動するには、
forの書かれている行まで移動して、
www
と押します。それから、
cwif<esc>
ですね。(<esc>はエスケープキーを押してください)
wで、単語単位に移動

って意味になります。「→」を押し続けて、fの位置まで移動するって考えちゃだめですよ。この場合は、カーソルの移動先が単語の先頭なので、単語単位で移 動すると考えてみます。もっと良い移動方法もありますが、とりあえず、右の方へ移動するときには、wって覚えていても良いと思います。「→」って覚えてい るなら治しましょう。
cwif の部分は、単語を削除(cw)してifって入力しています。

単語の途中に「-,.*;」などの記号が入っている場合には、

W

の方がよいかも知れません。Wはwと違って、空白文字(スペース、TAB文字)毎にジャンプしてくれます。
逆の操作としては、

b
B

があります、これは単語単位で左方向に進みます。

ここで紹介した操作は、Windowsのエディタで言うと以下の操作に当たります
wは、 Ctrl + → Wは、 ？ bは、 Ctrl + ← Bは、 ？

f 特定の文字へジャンプ

wよりも格好の良い移動方法を紹介します。

sample.txt
/* * Check if we support **Teh** binfmt * if we do, return the node, else NULL * locking is done in load_misc_binary */

↑Tehをtheに変える場合、Tの位置まで移動したいですね。

Tehの書かれている行まで移動して、
fT
と押します。それから
cwthe<esc>
です。

f?で、カーソル位置から右の?の文字にジャンプ

という意味になります。wを何回か押して移動するよりずっと早くて気持ちがよいですね。
一発で移動できない場合もあります。例えば、

sample.txt
/* * Check if we support teh binfmt * if we do, return the node, else NULL * locking is done in load_misc_binary */

↑こうなっている時に、
ft

と押しても、期待通りの位置に行きません。tehのtよりも前にsupportのtにヒットしてしまうためです。そんなときには、
;
と打ってください。

「;」は、次を検索

という意味になります。もう一度「ft」と押すのと同じ意味ですね。
「;」と逆で、「,」は前を検索という意味になります。行き過ぎた場合なんかに使えますね。
ちなみに、
Ft
と押すと、現在のカーソル位置から左方向にtを検索するコマンドになります。

ここで紹介した操作は、Windowsのエディタで言うと以下の操作に当たります
?

ct? 特定の文字まで削除

特定の文字までジャンプするときには、f?で出来ました。このジャンプは移動コマンドになるので、他のコマンドと組み合わせることが出来ます。たとえば、

sample.c

```
static char *scanarg(char *s, char del)
{
    char c;

    while (c = *s++ != del) {
        if (c == '\\' && *s == 'x') {
            s++;
            if (!isxdigit(*s++))
                return NULL;
            if (!isxdigit(*s++))
                return NULL;
        }
    }
    return s;
}
```

↑whileのカッコの中を
↓変えるには、

sample.c

```
static char *scanarg(char *s, char del)
{
    char c;

    while ( 1 ) {
        if (c == '\\' && *s == 'x') {
            s++;
            if (!isxdigit(*s++))
                return NULL;
            if (!isxdigit(*s++))
                return NULL;
        }
    }
    return s;
}
```

編集したい「while (」の後の文字に移動してから、

ct)

と押します。cは削除して挿入モードに移行することを意味していて、t?でf?の様に、ジャンプします。ただし、t?はf?と違って、?の1文字前を意味 します。この場合は、最後の「)」を削除したくはありませんから、cf)ではなく、ct)になります。これで、きれいに( )の中が消えました。後は、「1」と入力して終わりです。「ct)1」と押すだけでこれだけの操作が可能になります。

他にも、dt;でカーソル位置から行末の「;」以前を削除などが考えられますね。これを使えるようになると、かなり高速に編集できます。

ここで紹介した操作は、Windowsのエディタで言うと以下の操作に当たります
？

* 単語の検索

カーソル下の単語の検索方法です。

カーソル位置の単語を検索するには、
*****

と押します。これで、カーソル位置の単語を下方向に向かって検索します。次のヒット箇所を表示させるには、「n」を押します。単語の検索は、*nnn・・・という感じで使います。その後、逆方向に検索するには、「N」を押します。
最初から、カーソル位置の単語を前方に検索するには、

#

と押します。この時は、「n」で、さらに前方のヒット箇所を表示することになります。
ただし、*や#は単語単位で完全に一致しなければヒットしません。一部分でも一致した場合にもヒットさせるためには、

g*
g#

を使ってください。
ヒットした箇所の色が変わってうっとうしい場合には、

:nohl<TAB><Enter>

と打ってください。<TAB>キーを押した段階で「:nohlsearch」と補完されたと思います。この状態で<Enter>を押せば、ヒット箇所のハイライトが消えてくれます。

ここで紹介した操作は、Windowsのエディタで言うと、たぶん以下の操作

に当たります

*は、
？

#は、
？

nは
F3

Nは
Shift + F3

g*は
Ctrl + F3

g#は
Ctrl + Shift + F3

v 文字の選択

文字を選択する方法です。

文字を選択するためには、ビジュアルモードに移行します。

v

と押してください。この状態で、移動すれば、移動しただけ選択範囲が変わります。

1行丸ごと選択するには、

V

と押します。さらに、上下に移動すれば、選択範囲もその文変化します。

ファイルすべてを選択するなら、

ggVG

と押します。
ggは、ファイルの先頭に移動
Vは、行選択
Gで、ファイルの最後までジャンプです。

矩形選択をするなら、

Ctrl + v

と押します。移動すると矩形で選択されますね。この状態で、$を押せば、行の最後まで選択されるようになります。ファイルの最初に書かれている単語を無視して選択したい場合なんかに便利ですね。

ここで紹介した操作は、Windowsのエディタで言うと以下の操作に当たります

vは、
Shift + 移動
マウスのドラッグ

Vは、
行番号を表示している部分をマウスでドラッグ

ggVGは、
Ctrl + A

Ctrl + vは、
Alt + マウスのドラッグ (?)

～大文字←→小文字の変換

大文字から小文字、小文字から大文字への変換方法です。

sample.c

```
static char *scanarg(char *s, char del)
{
    char c;

    while ((c = *s++) != del) {
        if (c == '\\' && *s == 'x') {
            s++;
            if (!isxdigit(*s++))
                return null;
            if (!isxdigit(*s++))
                return NULL;
        }
    }
    return s;
}
```

↑ありゃ、nullって小文字で書いてしまいました。大文字になおしましょう。

では、nullのnまで移動して、
vw~
と押します。
vw は、「null;」を選択するという意味です。

~は、大文字←→小文字の変換

という意味になります。文字を選択していれば、選択した文字をすべて大文字←→小文字で変換します。文字を選択していない場合は、カーソル位置の1文字だ け変換するという意味になります。文字を選択せずに、「~」を押すと変換後に1文字右にカーソルが動きますから、連続して「~」を押せば、押した文字数だ け大文字小文字が切り替わります。

ここで紹介した操作は、Windowsのエディタで言うと以下の操作に当たります
？

. 同じことの繰り返し

さっきの編集をもう一度やりたいって時の操作です。

「.」

と押します。

.は、直前のコマンドを繰り返す

という意味になります。この機能は本当に便利です。例えば、

sample.txt
これは、間違った日本語の文章です。 これは、間違った日本語の文章です。 これは、間違った日本語の文章です。 これは、間違った日本語の文章です。 これは、間違った日本語の文章です。

↑これを
↓こうするには

sample.txt
これは、日本語の文章です。 これは、日本語の文章です。 これは、日本語の文章です。 これは、日本語の文章です。 これは、日本語の文章です。

1行目の削除したい「間違った」の「間」まで移動して、
2dw
と押します。dは削除、wは単語単位の移動を意味するので、単語を削除という意味になります。さらに、その前に2を付けているので、2つ単語を削除するという意味になります。
※日本語の場合は、「漢字」「ひらがな」「カタカナ」の連続が、それぞれ単語として扱われるようです。
※「間違った」は「間違」で1つ「った」で2つ目です。
これで、1行目が直せました↓。

sample.txt
これは、日本語の文章です。 これは、間違った日本語の文章です。 これは、間違った日本語の文章です。 これは、間違った日本語の文章です。 これは、間違った日本語の文章です。

修正個所は、残り4行ですね。キーボードマクロを使うまでもありません。次の行に移動して同じことをすれば良いわけです。次のように入力してください。

j.

.は、直前のコマンドを繰り返す

でしたから、jで下の行に移動、「.」で(2dwを)繰り返すという意味になります。
j.j.j.j.と押せば完了ですね。

ここで紹介した操作は、Windowsのエディタで言うと以下の操作に当たります
？

. 同じ文字を一気に複数行に挿入する

同じことを繰り返すのは何となくバカバカしいですよね。Vimには良いコマンドがあります。

sample.cpp

```
#include <iostream.h>

main()
{
    for(;;)
    {
        cout << "Hello World! ";
        cout << "Hello World! 2";
        cout << "Hello World! 3";
        cout << "Hello World! 4";
        cout << "Hello World! 5";
        cout << "Hello World! 6";
    }
}
```

↑たとえば、これを

↓こうするには、

sample.cpp

```
#include <iostream.h>

main()
{
    for(;;)
    {
        cout << "Hello World! ";
//        cout << "Hello World! 2";
//        cout << "Hello World! 3";
//        cout << "Hello World! 4";
//        cout << "Hello World! 5";
//        cout << "Hello World! 6";
    }
}
```

まず、「//」を付けたい行の先頭で、矩形ビジュアルモードに入ります。

Ctrl + v

下に移動して、「//」を付けたい行を選択します(行末まで選択する必要はありません)。つぎに、

I//

と押して「//」を入力します。I(大文字)で入力モードに入って、//で「//」を書き込み

ます。この操作では、選択した1行目にしか「//」が入りませんが、それで構いません。//を押したら、
<Esc>
で、入力モードを抜けます。そうすると、あ〜ら不思議、選択していた行の全てに「//」が入りましたね。これで簡単に複数行を修正できます。

ちなみに、選択位置の後ろに書き込む場合には、
A//
というようしてください。

ここで紹介した操作は、Windowsのエディタで言うと以下の操作に当たります
?

iw ビジュアルオペレータ

便利なビジュアルモードですが、さらに便利になる操作です。

カーソル位置の単語を選択するなら、

viw

vは選択の開始、iwがカーソル位置の単語を意味します。
カーソルが置かれている下の単語の削除なら

diw

です。これなら、カーソルが単語の途中にいるせいで、cwじゃうまく削除できないときにも使えます。
カーソル位置の単語のコピーなら

yiw

で出来ます。私は、yiwをよく使います。打ちやすいし、カーソル位置の単語をコピーしたいことはたくさんありますからね。

実は、iwだけじゃなく、オブジェクトを選択する操作は他にもあります。(:help visual-operators)

操作	意味

aw	1単語(空白文字を含む)
iw	1単語(空白文字を含まない)
aW	空白で区切られた1単語(空白文字を含む)
iW	空白で区切られた1単語(空白文字を含まない)
as	1文(空白文字を含む)
is	1文(空白文字を含まない)
ap	1段落(空白文字を含む)
ip	1段落(空白文字を含まない)
ab	(丸括弧文字を含む) ()の中身
ib	()の中身
aB	(波括弧文字を含む) {}のブロック
iB	{}のブロックの中身

これらのビジュアルオペレータが効くのは、以下のコマンドです。

コマンド	意味
~	大/小文字の切替
d	削除
c	変更
y	ヤンク
>	右シフト
<	左シフト
!	外部コマンドによるフィルタ
=	'equalprg'オプションで指定されたフィルタ
gq	'textwidth'の長さによる行の整形

ここで紹介した操作は、Windowsのエディタで言うと以下の操作に当たります
?

範囲を指定した置換

置換コマンドを使うときには、置換しちゃいけないところを置換しないように気を

付ける必要がありますよね。Vimには良いコマンドがあります。
選択範囲内だけの置換を行うためには、置換したい範囲を選択して、

:s/pattern/string/g

で出来ます。範囲を選択した状態で、「:」を押すと、コマンドライン(画面の一番下、ステータスライン？)に選択行を示す数字が表示されます。選択位置の先頭行から、最終行を示しています。これで、安心して思い切った置換を行っちゃいましょう。

置換コマンドで選択が解除されてしまいますので、再度同じ範囲を選択するには

gv

と押してください。これで、複雑な置換も複数に分けて実行することが出来ます。

ここで紹介した操作は、Windowsのエディタで言うと以下の操作に当たります
？