

Minnu's Filer2とはなんですか？

UNIX(Linux, FreeBSD, MacOSX)で動くファイラーです。

UNIXというのは普通のパソコンに入っているWindowsとは違う種類のパソコン(OS)です。Windowsは事務処理やネットサーフィンをするのに向いていますがUNIXは大学のワークステーションやウェブサーバー、組み込み機器などに利用されプロフェッショナルな用途に向いています。

Minnu's Filer2はそのUNIX上でコンピューティング全般の操作を簡単にするために開発されました。ファイラーというのはファイルのコピーや作成、プログラムの起動など基本的なコンピュータ操作全般を行います。同じことを行うのにUNIXではシェルというプログラムを使いますがファイラーはそれをよりビジュアル的に行います。同じくそれをビジュアル的に行うソフトにファイルマネージャーというものがありますが、こちらはキーボードよりマウスでの操作が主体です。それにたいしてファイラーはキーボードでの操作が主体となっている点が異なります。

Minnu's Filer2はキーボード主体のファイラーの中でもコンソールという文字だけのインターフェースを使ったものです。シェルも同じようにコンソール上で動くのでMinnu's Filer2はシェル的一种だと見ることもできます。事実内部的にはシェルを内蔵しています。このようなソフトをビジュアルシェルと呼ぶこともあります。Minnu's Filer2はファイラーの良さとシェルの良さを兼ね備えたソフトなのです。

ファイラーの歴史

歴史的にはファイラーは少し前のパーソナルコンピュータ上のOS, MS-DOS上で発展してきました。日本ではFD, 海外ではNorton Commanderというソフトがよく使われてきました。Windowsの時代になりMS-DOS時代のファイラーの流れを組むWindows用のファイラーもたくさん生まれましたが、Windowsに付属するエクスプローラーの出来が良かったこともあり、現在ではファイラーを使う人は少なくなっています。しかし、ファイラーには長い歴史があり、いいファイラーソフトが生まれれば、新規ユーザーも増えていくのではないかと考えています。最近はファイルマネージャー、ファイラーともにスクリプトを内蔵する傾向にあり、Minnu's Filer2もRubyを内蔵しています。スクリプトを内蔵すると単調な処理をスクリプトで自動化することができます。

インストール

インストールはMinnu's Filer2ホームページをご覧ください。まだパッケージが用意されていないので、少々敷居が高いと思います。各OSのメンテナにパッケージ化を頼もうと思っています。

さあ、mfiler2を使ってみよう。まずはコピーだ！

インストールが無事すめば、シェル上でmfiler2というコマンドを実行すればMinnu's Filer2が起動します。左右二つにディレクトリが表示されていると思います。アクティブなディレクトリは上部のパス名表示部分が強調されています。逆側のディレクトリをスリープディレクトリと呼びます。コピーなどのファイル操作の処理は大体アクティブなディレクトリからスリープディレクトリにされます。

まずはコピーの下準備。ディレクトリを移動しましょう。カーソルキー上下でカーソルが移動します。ディレクトリの上にカーソルを持って行ってENTERキーを押せば、そのディレクトリの中に入ります。逆にBSキーを押せば親ディレクトリに移動します。カーソルキー左右を押せばアクティブなディレクトリを左右移動できます。とりあえず、コピーのテストができるようにアクティブなディレクトリをコピーしてもいいような小さいファイルがあるディレクトリ、スリープディレクトリに/tmp, \$HOME/tmpなど、一時ファイルを置くディレクトリに移動してみましょう。

さあ、ファイルのコピーです。まずはファイルのマーク。スペースキーを押せばカーソル下のファイルがマークされます。マークされたファイルは*がファイル名の横に表示されたはず。ここでF5キーかcキーを押せば、スリープディレクトリにマークされたファイルがコピーされるはず。できましたか？mfiler2によるファイルコピー操作はこれだけです。わざわざいいディレクトリ指定など要りません。非常に直感的に簡単にコピーできる気がしませんか？

次はコピーしたファイルを削除してみましょう。スリープディレクトリに移動するため、左右どちらかのキーを押して、スリープディレクトリをアクティブにして、コピーされたファイルをマークしてください。その後、F8かdキーを押してください。確認の質問文が表示されます。Yesを選択してEnterキーを押してください。これでファイルがごみ箱(\$HOME/mtrashbox)に移動されます。誤操作で消したファイルは(\$HOME/mtrashbox)に置いてあるので、そこから復活させてください。本当にファイルを削除した

い場合は、Dキーを押してメニューからdeleteを選択してください。ファイルが本当に削除されるはずですが、

ファイルの移動は同様にディレクトリを移動して、ファイルをマークしたあとに、F6かmキーでできます。

いちいちファイル選択するのがめんどくさいよ！シェルなら*.cppとかで一括に選択できるのに！

ごもっともです。そのためにMinnu's Filer2には豊富な一括マーク操作を用意しています。

まずは全体選択です。HOMEキーを押すとディレクトリ以外のファイルがすべて選択されます。これはマークするのではなくてマークを反転します。つまりすでに選択されていたファイルは逆にマークが外されます。これはいろいろ応用範囲が広いので、覚えて置いてください。

ENDキーを押すとディレクトリを含むすべてのファイルが選択されます。

「これだけだとシェルで言う*という機能しかないじゃん！*.cppとかの方が便利だ」

大丈夫です。mfiler2には*.cppなどのグロブ（ワイルドカード）によるマークもサポートしています。*を押してください。メニューが表示されたはずですが、

glob mark (f)ileを選んでください。

```
glob_mark '*', '*', '*', '*'
```

と表示されたはずですが、*.cppのマーク場合

```
glob_mark '*.cpp', '*', '*', '*'
```

で全てのcppファイルがマークされます。

toggle mark?と聞かれますがnoと答えてください。

ほかの引数の意味。

第一引数はファイル名の指定で*.cppとすれば拡張子がcppのファイルをマークできます。

第二引数はユーザー名の指定でrootと指定すればrootのファイルがマークできます。

第三引数はグループ名の指定。

第四引数はパーミッションの指定です。

マークには反転マーク(toggle mark)と単なるマークオンがあります。

toggle markの使い方の例。

まず全ての*.cppファイルをマークする

```
glob_mark '*.cpp', '*', '*', '*'
```

toggle mark? no

次に所有者がrootのファイルを取り除く

```
glob_mark '*.cpp', 'root', '*', '*'
```

toggle mark? yes

次にmから始まるcppファイルを取り除く

```
glob_mark 'm*.cpp', '*', '*', '*'
```

toggle mark? yes

などという使い方ができます。

逆にtoggle mark? noの使い方は

まず全ての*.cppファイルをマークする

```
glob_mark '*.cpp', '*', '*', '*'
```

toggle mark? no

次に全ての*.cファイルをマークする

```
glob_mark '*.c', '*', '*', '*'
```

toggle mark? no

です。

ほかにも

glob mark (f)ile ファイルのみのワイルドカードマーク

glob mark (a)ll ディレクトリなども含むワイルドカードマーク

glob mark (d)irectory ディレクトリのみワイルドカードマーク

glob mark (e)xecutable 実行ファイルのみワイルドカードマーク

glob mark (r)eadable 読み込み可能ファイルのみワイルドカードマーク

glob mark (w)ritable 書き込み可能ファイルのみワイルドカードマーク

glob mark (s)ymmlink シンボリックリンクファイルのワイルドカードマーク

regex mark (F)ile 正規表現によるマーク

mark (c)lear マークのクリア

mark a(l)l files ファイルのみのマークの反転

mark a(L)l 全てのファイルのマーク反転

mark with c(o)mparison 2画面時左右のファイルを比較して同じファイルのみマーク

一括マークができるのはカレントディレクトリだけじゃんか！カレントディレクトリ以下すべての*.cppファ

イルに操作が行いたい

Fによるファイル検索の仮想ディレクトリを使えば可能です。F->allでカレントディレクトリ以下のすべてのファイルを表示できます。このあと*のマークメニューのグロブによるマークで*.cppとしてcppファイルを選択すればいいでしょう。Fは他にも、あるサイズ以下のファイルを表示したり、ある更新日のファイルを表示したりできます。

ファイルの編集がしたい！UNIXではプログラミングやシステム管理でテキストファイルの編集を多用するし、

編集したいファイルの上にカーソルを移動して、F4かeを押してください。環境変数\$EDITORで指定したエディタが立ち上がるはずですが、この場合vimやemacs -nwなどコンソール上で動くものだと、編集途中にファイル操作などがしたくなったらCTRL-Zでサスペンドすることができます。サスペンドするとmfile2のファイラー画面の下に番号とサスペンドしたファイルのファイル名が表示されるので、もう一度エディタ画面に戻りたい場合はその番号のキー(1なら1キー)を押せば、またエディタ画面に戻ることができます。

X-Windowシステム上のエディタ(gvim, xemacs, geditなど)を使用したい場合はカスタマイズする必要があります。それは後述。

いちいちファイルを探すのが面倒だ、やっぱりシェルでファイル名を入力の方が早い

そのためにmfile2にはファイルのインクリメンタルサーチが実装されています。fか/キーを押してください。そして目的のファイル名を入力すれば、そのファイルにカーソルが移動します。Enterキーで確定です。migemoつきでmfile2をコンパイルしていれば日本語ファイルもローマ字で検索することができます。日本語ファイルを含むコマンドをシェルから起動する場合とても大変ですがmfile2を使えば、とても簡単に日本語ファイルを扱えます。

コピー、移動、削除、編集、以外のファイル操作は？

ファイルの作成はF10かnです。%r%Q mark_new_fileと表示されるのでの間に作りたいファイル名を入力すればファイルが作成されます。

ディレクトリの作成も同様にF9かkで行えます。

ファイル名の改名はファイル名を変えたいファイルの上にカーソルを持って行ってF2かrです。するとコマンドラインに以下のように表示されるので
%Q%r rename '改名前のファイル名', '改名後のファイル名'
を入力してEnterを押してください、このときCTRL-Wによる後方単語削除が便利につかえると思います。

マークしてF2かrを押すと正規表現による一括ファイル改名が行えます。
試しにアクティブディレクトリにa.cpp, b.cppというファイルを作ってください。
a.cpp, b.cppをマークしましょう。
そしてF2かrを押してください。
gsub_rename //, ""
と表示されたはずですが。
正規表現をご存知でない方はUNIXでは正規表現を多用するので、このさい勉強しておきましょう。
/(.+).cpp/, "\\1.c"
と入力すると確認用のページが表示されるので、ちゃんとa.c, b.cに改名されているか確認してください。Yesを選択するとa.cppがa.cにb.cppがb.cに改名されたのがわかると思います。
Rubyを利用してこの機能を作成しているので正規表現はRubyに準拠しています。詳細はRubyのページや本を見てください。

ファイル操作は便利にできることは分かった、でも、それ以外の処理をするにはやっぱりシェルが必要だ！

ご安心を、mfile2はシェルを内蔵していて、カスタマイズ可能で高機能な補完機能がついたコマンドラインがあります。hかF10を押すとコマンドラインに移行します。そこで好きなコマンドを入力してEnterを押せば、そのコマンドが実行されます。

TABを押せば補完候補が上の方に表示されますが、さらにTABキーを押せば補完候補をカーソルキーで選択できます。

カーソル下のファイルに何かのコマンドを実行したい場合、そのファイルにカーソルを合わせてF11かxを押せば、便利です。(ほとんどはファイル実行なんのですが、こういう利用法もあります)
マークした複数のファイルにコマンドを実行したい場合はファイルをマークしてからENTERを押してください。
%mと表示されますが、%mはマークされた複数のファイルに変換されるマクロです。
cat %mならマークされた全てのファイルの内容が全て表示されるはずですが。

「最近のシェルは高機能だからなあ。ヒストリ検索とかもあんの？」

あります。検索したい語を入力したあとCTRL-Rを入力すれば、その語を含むコマンドライン履歴が表示されメニューから選択できます。

Rubyスクリプトによる処理の自動化

Meta+x(Escを押してからすぐx、もしくはAlt + x)を押してください。そうすると\$EDITORが立ち上がります。そこにRubyによるスクリプトを書けば処理を自動化することができます。

試しにマークした全てのファイルのファイル名の拡張子の前に連番を付ける処理を書いてみましょう。

まず連番を付けたいファイルをマークします。そしてMeta+xを押してRubyスクリプト編集用のエディッタを立ち上げましょう。で、以下を入力します。

```
n = 1
adir_mark.each do |f|
  ret = (f.scan /(.) \.(.+)/)[0]
  File.rename f, ret[0] + n.to_s + ret[1]
  n += 1
end
```

エディッタを保存して終了するとスクリプトを実行して良いか聞いてくるのでYesと答えてください。正しく実行されるはずですが、a.txt, b.txt, c.txt --> a1.txt, b2.txt, c3.txtなど。
adir_markはmfiler2組み込みの関数で、マークされたファイルのファイル名の配列を返します。scanなどはRubyを勉強して理解してください。そんなに難しいことはしていません。
処理が気に入った場合、オリジナルのコマンドとしてmfiler2に登録しましょう。次のカスタマイズの章で登録の仕方を教えます。

カスタマイズ

mfiler2をカスタマイズするには/usr/local/etc/.mfilerをホームディレクトリにコピーして\$HOME/.mfilerを編集します。

エディッタをgvim, xemacs, geditなどXウィンドウ上のものを使いたい場合は\$HOME/.mfilerに

```
keycommand NOMETA, KEY_e, "", "shell('%q%o xemacs %f &', '%f')"  
keycommand NOMETA, KEY_F4, "", "shell('%q%o xemacs %f &', '%f')"
```

を追加してください。

keycommandはキーバインドを設定するコマンドでNOMETA, KEY_eはMetaキー無しのエキーという意味で、""はカーソル下の拡張子はなんでもOKという意味。

"shell('%q%o xemacs %f &', '%f')"は実行するコマンドです。

%qはコマンド待ちをしないマクロ

%oはコマンド実行後マークをクリアしないマクロ

%fはカーソル下のファイル名に変換されるマクロです。

shellはシェルコマンドを実行するコマンドです。

この場合eキーを押せばxemacsが起動します。

オリジナルコマンドを追加する。登録する関数がset_renbanだとするとset_renbanの定義をmfiler内に追加して

mfiler末尾の

```
defmenu("my_commands",  
  "(1) set_number_at_head", KEY_1, "set_number_at_head()",  
  "(2) del_number_at_head", KEY_2, "del_number_at_head()",  
  );  
  
keycommand NOMETA, KEY_y, "", "menu('my_commands')"
```

を

```
defmenu("my_commands",
```

```
"(1) set_number_at_head", KEY_1, "set_number_at_head()",  
"(2) del_number_at_head", KEY_2, "del_number_at_head()",  
"(3) set_renban", KEY_3, "set_renban"  
);
```

```
keycommand NOMETA, KEY_y, "", "menu('my_commands')"
```

としてください。
yを押せばメニューが追加されているはずですが