# 大津真＝X は UNIX でサーバで》
# UNIX の定番エディタ - Vi と Emacs

UNIX 系 OS で使用可能な代表的なフリーエディタに Vi と Emacs があります。さて、どちらがすぐれたエディタ？突き詰めると宗教戦争になりかねませんから、ここではふれないことにしましょう。

さすがに今では素の Vi が使われることは少なくなり、Vi クローンと呼ばれる Vi の機能強化版が主流です。代表的な Vi クローンとして VIM が有名です。Vim は早くから多言語に対応し、最近のバージョンでは GUI も備えています。

Emacs に関しては、GNU プロジェクトによる本家の Emacs の開発も進められていますが、Emacs の開発の遅さに業を煮やして分派した XEmacs という流派があります。XEmacs はメニューバーやツールバーなどの GUI にいち早く対応し、その使い勝手のよさからユーザ数としては本家 GNU Emacs をしのぐ勢いです。なお、Emacs 系エディタは Emacs Lisp と呼ばれる Lisp 言語を使用して自由にカスタマイズできることが人気の秘密ですが、両者の開発が進むにつれて互換性が乏しくなってきているのが残念です。

◇VIM

http://www.vim.org

◇Emacs

http://www.gnu.org/directory/emacs.html

◇XEmacs

http://www.xemacs.org

## Mac OS X で Vi と Emacs を使用するには

Mac OS X には最初から Vi(vi コマンド)と Emacs(emacs コマンド)が用意されています。ただし、どちらも Terminal 内での使用を前提にしているため、日本語が使えません。解決策としてはフリーX Window System である XFree86 をインストールしておくとい

う方法があります。

◇XFree86

http://www.xfree86.org/

その場合の手順の概略を次に示します。

1. XFree86 をインストール
2. Kterm などの日本語ターミナルをインストール
3. Canna などの日本語入力システムをインストール
4. Vi 派の方は、Vi の拡張版である VIM をインストール
5. Emacs 派の方は、Emacs 21 あるいは XEmacs をインストール

XFree86 の最新版であるバージョン 4.2 は、Mac OS X に正式に対応しているためバイナリディストリビューションをダウンロードしたら後はインストーラの指示に従っていけば簡単にインストールできます。これまでは別途 XDarwin が必要であった、Aqua と共存可能な「ルートレースモード」も最初から組み込まれています。ただし、詳しくは述べませんが、XFree86 4.2 は日本語のロケールのサポートに難がありまともに機能しません(俗に恐怖の X Locale 問題と呼ばれています)。
そのため筆者の場合には、まだ XFree86 4.2 への移行には踏み切れず、旧バージョンの XFree86 4.1 を、日本語ロケールライブラリ libxpg4 と組み合わせて使っています。libxpg4 は、Darwin 関連情報のサイトを主催する井上氏が FreeBSD から移植したロケールライブラリです。

◇Darwin 関連情報のサイト

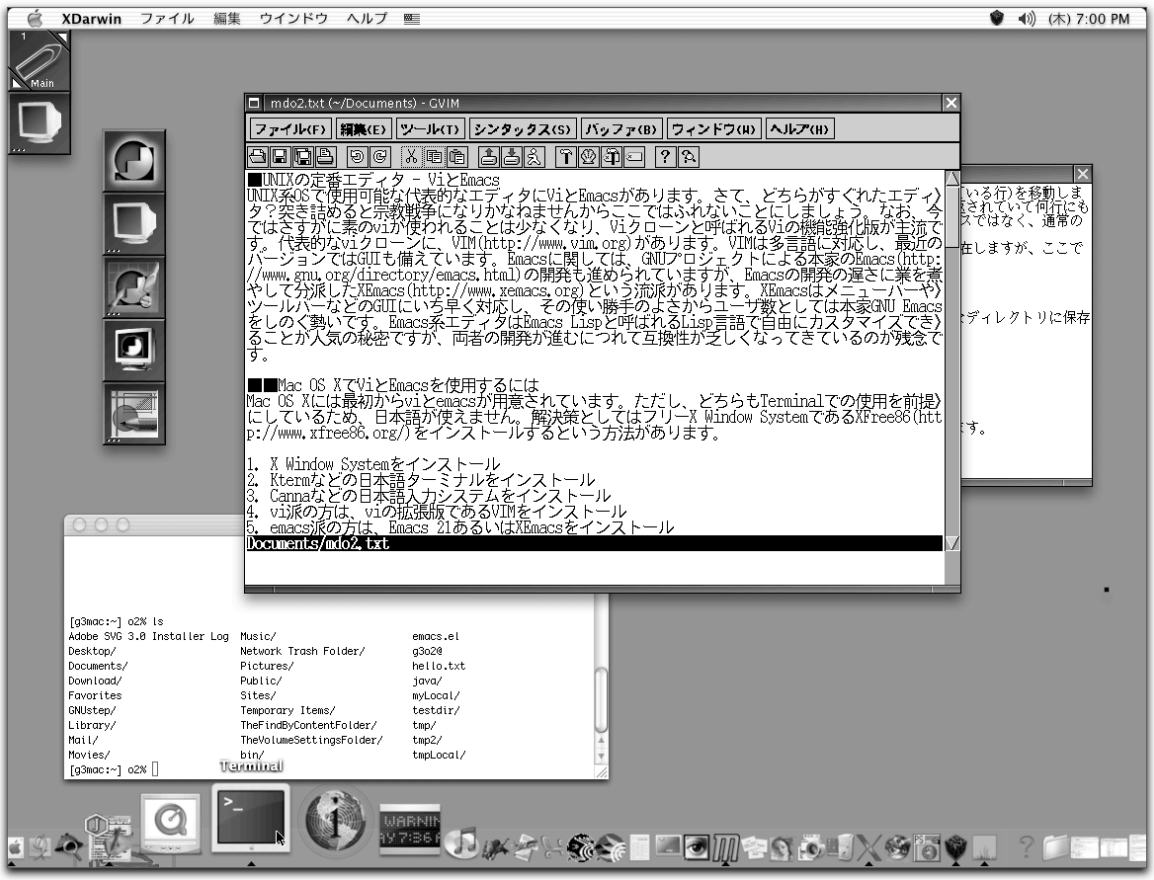
http://www.ab.wakwak.com/~tino/darwin/

次に、Mac OS X に X Window System をインストールし、その上で動作させた XEmacs と GVIM(GUI を備えた VIM)の画面を示します。

## XEmacs


Explorer ファイル 編集 表示 移動 お気に入り ツール ウィンドウ ヘルプ
(木) 6 55 PM
Main
emacs: MDO-o2-08.txt
File Edit View Cmds Tools Options Buffers Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Info Compile Debug News
MDO-o2-08.txt
Mac OS XでViとEmacsを使用するには
Mac OS Xには最初からviとemacsが用意されています。ただし、どちらもTerminalでの使用を前提にしているため、日本語が使えません。解決策としてはフリーX Window System であるXFree86(http://www.xfree86.org/)をインストールするという方法があります。
1. X Window Systemをインストール
2. Kterm などの日本語ターミナルをインストール
3. Canna などの日本語入力システムをインストール
4. vi派の方は、viの拡張版であるVIMをインストール
5. emacs派の方は、Emacs 21あるいはXEmacsをインストール
最新版のXFree86 4.2は、Mac OS Xに正式に対応し、インストーラの指示に従っていけば簡単にインストールできます。また、Aquaと共存するルートレースモードも最初から組み込まれています。ただし、詳しくは述べませんが、XFree86 4.2は日本語のロケールのサポートに難がありまともに機能しません。そのため筆者の場合には、旧バージョンのXFree86 4.1を、日本語ロケールライブラリlibxpg4と組み合わせて使っています。libxpg4は、Darwin関連情報のサイト(http://www.ab.wakwak.com/~tino/darwin/)を主催する井上氏がFreeBSDから移植したライブラリです。
次に、Mac OS XにX Window Systemをインストールし、その上で動作させたXEmacsとGVIM(GUIをそなえたVIM)の画面を示します。
fig
fig
かんなJa/SJIS:t-----XEmacs: MDO-o2-08.txt (Text)----L5--C77--9%--
[g3mac:~] o2% ls
Adobe SVG 3.0 Installer Log
Desktop/
Documents/
Download/
Favorites
Temporary Items/
TheFindByContentFolder/
TheVolumeSettingsFolder/
bin/
emacs.el
かんなJa/EUC--**-XEmacs: *shell* (Shell:run)----L6--C0--Top------
nsmail serikawa
rpm1.txt service2
rpm2.txt w3m-0.2.1-11.i38
emacs-21.4.3-3c7a4c86.dmp
Applications
Info
Run...
XTerm
Rxvt
Workspaces
Applications
Editors
Miscellaneous
Utils
Selection
Workspace
Appearance
Exit
XDarwin

GVIM




どちらも、プログラムのソースコードや設定ファイルの編集には十分すぎる機能をもっています。ただし、残念ながら OS X の日本語入力システムは使えず、Canna や SKK といった一昔前の入力システムを使用する必要があるため、日本語の入力は快適とは言い難いものがあります(もちろん好みにもよりますが)。筆者は「ATOK が使えないとちょっと...」と感じる柔なタイプな人間のため、原稿の作成などにはあまり使っていません(隣の Linux マシンには ATOK X をインストールし XEmacs を日常的に使用しています)。

## Carbon 版 Emacs 21

さて最近になって、Andrew Choi 氏による Emacs バージョン 21 の Carbon パッチ(「付録」UNIX の基礎知識参照）が登場しました。OS X の「ことえり」や「ATOK」が使用可能なため、快適な日本語入力が行えます。ただし、残念ながらインライン変換は行えません。

## Carbon で動く Emacs


ここでは Carbon 版 Emacs 21 のインストール方法を紹介しましょう。まだアルファ版ということで、不安定な側面もあるということですが、筆者の環境ではほぼ安定して使用できています。ただし、非同期のサブプロセスには未対応なため、Emacs のウリのひとつである、シェルやメールなどのすべての環境を Emacs 内で実現するといったことはできません(この問題を修正するパッチも存在するそうです)。

次にインストール手順を示します。

1. Emacs のサイトより、ソースの tar ボール(emacs-21.1.tar.gz)をダウンロードして、適当なディレクトリで展開します。

◇Emacs のサイト

http://www.gnu.org/directory/emacs.html

% tar xvzf emacs-21.1.tar.gz 〈return〉

emacs-21.1/

emacs-21.1/FTP

emacs-21.1/INSTALL

emacs-21.1/README

emacs-21.1/BUGS

〈以下略〉

これで、emacs-21.1 ディレクトリが作成され、ソースが展開されます。

2. Sourceforget の「Emacs 21.1 for Mac OS X and Classic」のサイトより、Gzip 形式で圧縮された Carbon パッチ(emacs-21.1-2-mac.patch.gz)をダウンロードして、解凍します。

◇「Emacs 21.1 for Mac OS X and Classic」のサイト

http://mac-emacs.sourceforge.net

% gunzip emacs-21.1-2-mac.patch.gz 〈return〉

3. emacs-21.1 ディレクトリに移動し、パッチを当てます

% cd emacs-21.1 〈return〉

% patch -p1 < ../emacs-21.1-2-mac.patch 〈return〉

patching file ChangeLog

patching file Makefile.in

patching file README

patching file configure

patching file configure.in

patching file etc/DEBUG

patching file etc/PROBLEMS

patching file lisp/ChangeLog

〈以下略〉

4 configure スクリプトを実行します。このとき X Window System をインストールしてある場合には「--without-x」オプションを指定してください。

```
% ./configure --without-x 〈return〉
creating cache ./config.cache
checking host system type... powerpc-apple-darwin5.2
checking for gcc... no
checking for cc... cc
checking whether the C compiler (cc ) works... yes
checking whether the C compiler (cc ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether cc accepts -g... yes
checking whether ln -s works... yes
checking how to run the C preprocessor... cc -E -traditional-cpp
〈以下略〉
```

5. コンパイルします。

```
% make 〈return〉
cd lib-src; make all ¥
  CC='cc' CFLAGS='-g -O2 ' CPPFLAGS='-fpascal-strings -fno-common -DMAC_OSX -I../mac/src '¥
  LDFLAGS='' MAKE='make'
cc -fpascal-strings -fno-common -DMAC_OSX -I../mac/src -DHAVE_CONFIG_H -I. -I../src -I/Users/o2/Download/emacs-21.1/lib-src -I/Users/o2/Download/emacs-21.1/lib-src/../src -fpascal-strings -fno-common -DMAC_OSX -I../mac/src -g -O2 -o test-distrib /Users/o2/Download/emacs-21.1/lib-src/test-distrib.c
〈以下略〉
```

6. sudo コマンド経由でインストールします。

```
% sudo make install 〈return〉
Password: ←パスワードを入力
cd lib-src; make all ¥
```

```
   CC='cc' CFLAGS='-g -O2 ' CPPFLAGS='-fpascal-strings -fno-common -DMAC_OSX -
I../mac/src       '￥
   LDFLAGS='' MAKE='make'
make[1]: Nothing to be done for `all'.
cd src; make all   ￥
〈以下略〉
```

以上で Emacs のバンドル(Emacs.app)が、emacs-21.1/mac ディレクトリに作成されます。Applications フォルダなどにコピーしておくといいでしょう。

## Emacs の設定

Emacs は起動時に~/.emacs.el ファイルを読み込みます。このファイルには Emacs Lisp 言語で設定を記述します。次に基本的な設定例を示します。

〈リスト〉~/.emacs.el の例

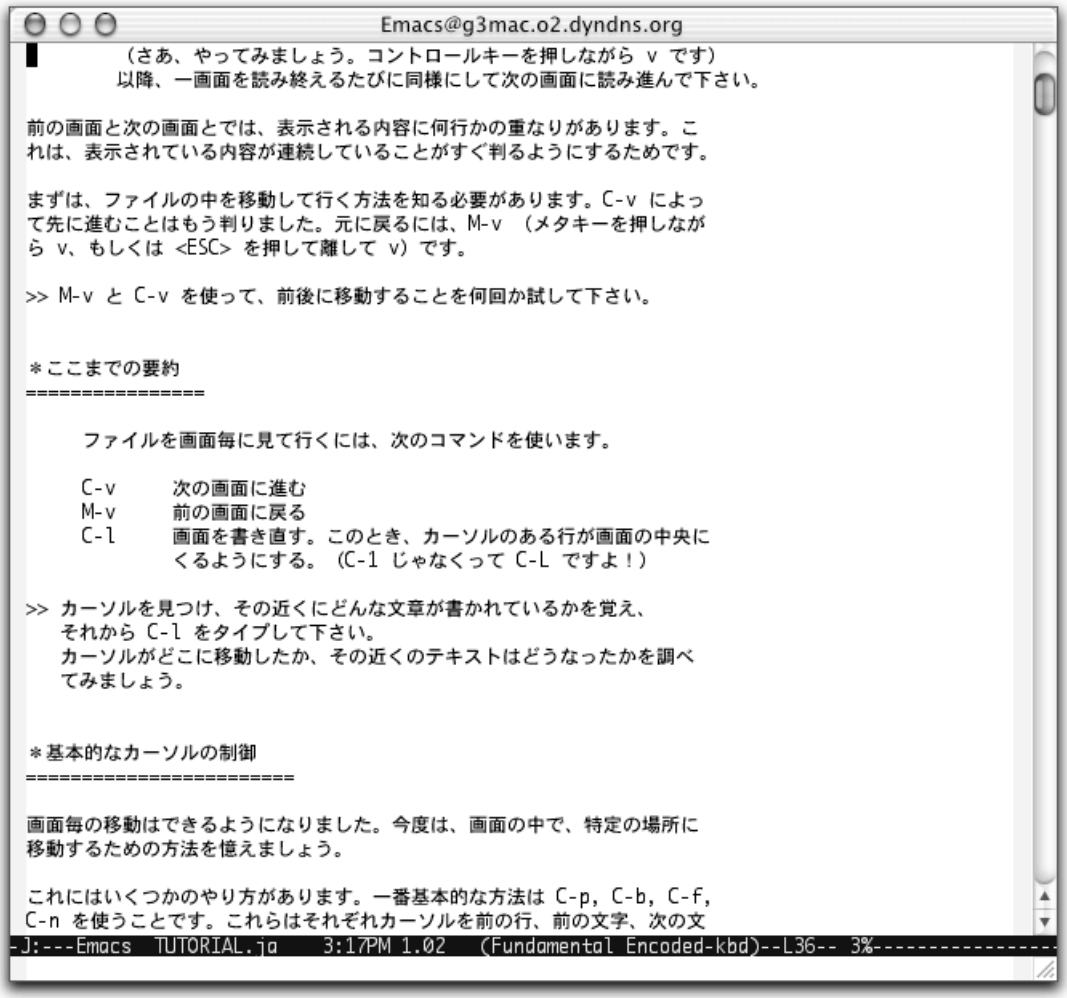
```
;; 日本語の設定(この例では Shift-JIS に設定している)
(set-language-environment 'japanese)
(set-keyboard-coding-system 'japanese-shift-jis-mac)
(set-terminal-coding-system 'japanese-shift-jis-mac)
(set-buffer-file-coding-system 'japanese-shift-jis-mac)

;; モードラインに時間を表示
(display-time)

;;モードラインの色を設定
(set-face-foreground 'modeline "yellow")
(set-face-background 'modeline "navy")
```

## Emacs の起動

設定が完了したら、Finder で Emacs のアイコンをダブルクリックして起動してみまし

ょう。Emacs の使い勝手はちょっとくせがあるので、まずはチュートリアルをやってみることをオススメします。「Help」メニューから「Emacs Tutorial (choose language)...」を選択してください。ミニバッファー(ウィンドウの一番下の行)に「Language:」と表示されるので、続けて「japanese」とタイプして return キーを押すと日本語のチュートリアルが起動します。

**Emacs の日本語チュートリアル**




●カーソルで物理行を移動するには

一般的なエディタでは、カーソルを縦方向に移動すると物理行(実際に見えている行)を移動しますが、Emacs の場合には論理行単位で移動します。つまり段落が行末で折り返されていて何行にも渡る場合に、カーソルが段落単位で移動してしまいます。プログラムのソースではなく、通常の文章を作成する場合にちょっと使いづらいと感じるでしょう。

この問題に対処する Emacs Lisp のプログラムはインターネット上にいくつか存在しま

すが、ここでは、モード単位の設定が可能な physical-line を紹介しましょう。

◇physical-line
http://www.taiyaki.org/elisp/physical-line/

1. physical-line.el をダウンロードし、「~/.site-lisp」など適当なディレクトリに保存します。

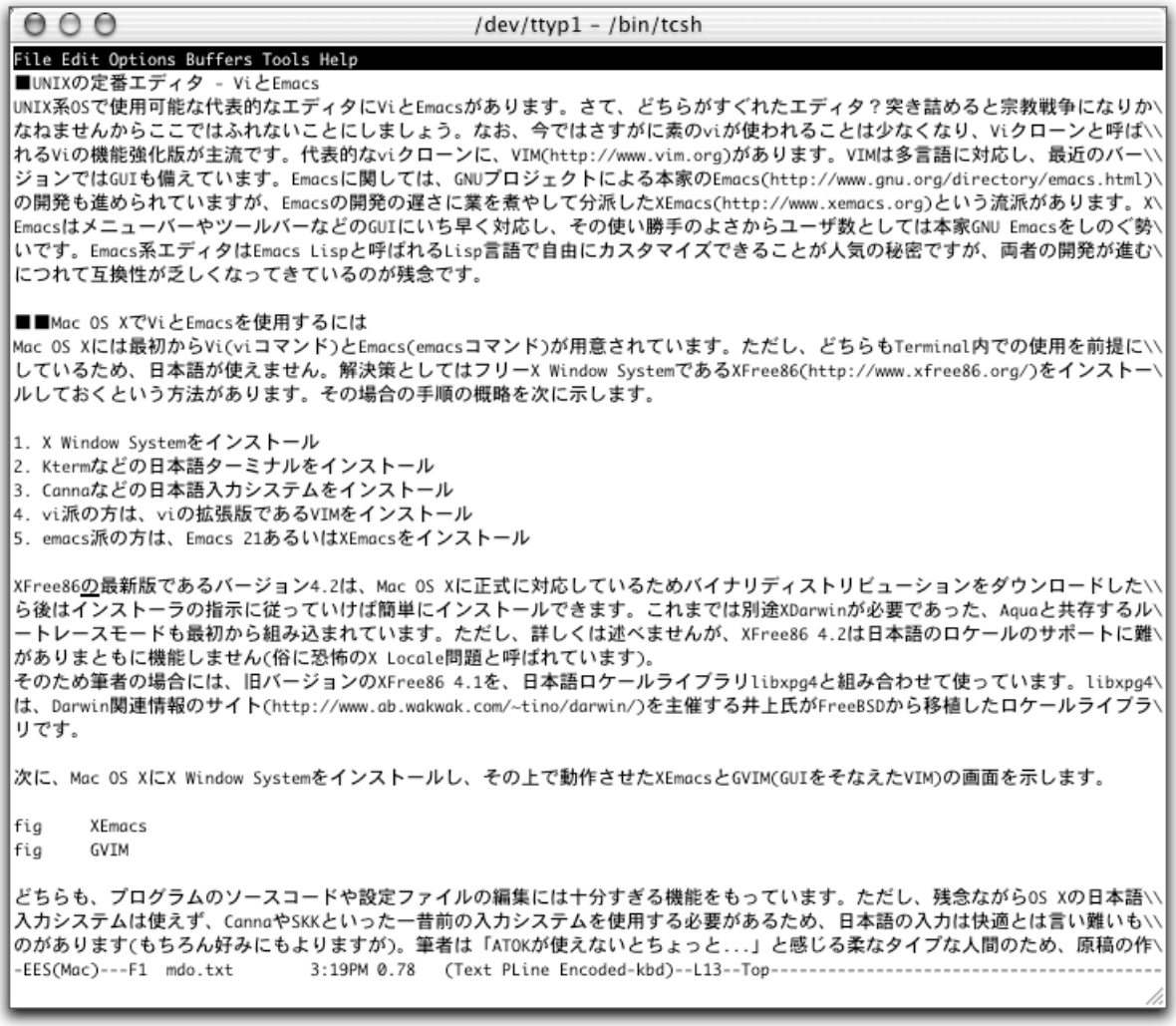
2. ~/.emacs.el に次のような行を加えます。
(load "~/.site-lisp/physical-line.el")
(add-hook 'text-mode-hook 'physical-line-mode-on)

以上で、Emacs を再起動すればカーソルが物理行単位で移動するようになります。

## JTerminal 上で Emacs を使う

メニューやスクロールバーは不要という上級者の方は、Kusama 氏による日本語 Terminal である JTerminal 内で Emacs を使うという手もあります。ことえりや ATOK による「インライン変換もどき」の変換も可能です。

## JTerminal で使う Emacs




◇JTerminal のオフィシャルサイト

http://www.tt.rim.or.jp/~kusama/

JTerminal 上で Emacs を起動するには次のようにします。

% emacs -nw 〈return〉

なお、JTerminal が対応している文字コードは日本語 EUC のみなので、Emacs の設定ファイル(~/.emacs.el)ではコーディングシステムの設定を「euc-jp」にしておきます。

```
(set-language-environment 'japanese)
(set-keyboard-coding-system 'euc-jp)
(set-terminal-coding-system 'euc-jp)
```

## パッチについて

元のソフトウエアと、それを元に修正したソフトウエアとの差分ファイルのことを「パッチ」と呼びます。元のソフトウエアにパッチを適用することを「パッチを当てる」といいます。「パッチ」とは日本語では洋服の継ぎ当てのような意味ですが、ソフトウエアに細かな修正を施すものと考えればいいでしょう。もちろん、パッチの数が膨大になりすぎると混乱を招くため、重要なパッチはバージョンアップ時にオリジナルのソースにマージされるのが普通です。
パッチには、ソースファイルに適用するソースパッチと、バイナリファイルに適用するバイナリパッチがあります。オープンソースソフトウエアの場合には通常パッチはソースパッチとして提供されます。

ソースパッチを当てるには patch コマンドを使います。パッチファイルのパスはリダイレクション「<」で指定することに注意してください。

patch -p 番号 < [パッチファイルのパス]

オプションの「-p 番号」は、パッチファイルを作成した環境と、パッチを当てている環境のディレクトリ構造を調節するために使用します。パッチファイル内には、

+++ dir1/file1

のようにソースのパスが指定されています。「-p0」と指定した場合には、パッチを作成した環境と同じディレクトリ構造と見なされパスがそのまま使われます。
「-p1」とした場合には、パッチファイル内で指定されているパスから最初のディレクトリ部分を取り除いて適用します。「dir1/file1」の場合には「dir1」が取り除かれ「file1」が対象と見なされ、カレントディレクトリにファイルがある状態でパッチが当てられます。
いずれにしてもディレクトリ構造が異なる場合にはエラーになるので、パッチに

README などが含まれていない場合、まず「-p0」を指定して実行してそれでエラーならば「-p1」を指定すると、たいていの場合うまくいきます。

## パッチの作り方

パッチを作成するにはファイルの差分を表示する diff コマンドを使用します。元のソースが保存されているディレクトリが「program」の場合、まずこれを「program.org」のような名前でコピーしておきます。

% cp -rp program program.org 〈return〉

ソースファイルの修正が完了したら、diff コマンドでパッチを作成します。

% diff -uNr program.org program > mypatch 〈return〉

ここで「u」はユニファイド差分形式と呼ばれる人間にわかりやすい形式の差分ファイルを作成するオプション、「Nr」はディレクトリ単位でパッチを作成するためのオプションです。
次に作成されたパッチである「mypatch」ファイルの中身を示します。

〈リスト〉パッチファイルの例

```
diff -uNr program.org/Hello.java program/Hello.java
--- program.org/Hello.java          Thu Feb 21 23:33:13 2002
+++ program/Hello.java    Thu Feb 21 23:36:21 2002
@@ -1,7 +1,8 @@
 class Hello {
         public static void main(String argv[]){
-                //String s;
-                System.out.println("Hello Worlid");
+                String s;
+                s = "Konnitiwa";
+                System.out.println(s + "Hello Worlid");
         }
 }
```

最初の行にパッチの作成に使用したコマンドが記述されます。2 行目、3 行目には変更前と変更後のソースファイルのパスが記述され、その後ろに変更部分のリストがその周囲を含めて表示されます。複数のソースファイルが変更されている場合には、その数だけ変更部分が記述されます。

ᔓᔓᔓᔓᔓᔓᔓこの項、以上ᔓᔓᔓᔓᔓᔓᔓ[大津真]ᔓᔓᔓᔓᔓᔓᔓ