自分の計算機環境を作り上げる
－フリーソフトで使いやすく－

医療技術短期大学部　髙梨一彦
takanasi@cc.hirosaki-u.ac.jp

1．はじめに

子どもの頃からずっと不思議に思っていましたが、大工さんはどうしてあんなにうまく道具を使って仕事をする－例えば鋸をまっすぐに引く－ことが出来るのでしょうか。昨年亡くなった大工の叔父（からだったと思うのですが）から聞いた話ですが、これには秘密がありました。曰く、「自分の癖を知ってそれを打ち消すように道具に手を入れる」のだそうです。例えば、鋸を引くと右側に曲がってしまうという人の場合、鋸の目立てをする時に普通に引くとやや左側に曲がるように刃を研ぐのだそうです。これは、鑿（のみ）や鉋（かんな）にしても同じで使う本人向けに多少の微調整をしているということでした（使う人それぞれの環境をカスタマイズする、ということになるでしょうか）。確かにその叔父の鉋を使って板を削らしてもらったことがありましたが、とても使いにくかったことを覚えています。もっともこれ、腕のよしあしというか、使いこなす技術の有無が大きく影響していたのでしょう。

計算機を使う時にもこのことは当てはまるように思います。ワークステーション上である人はX Window systemをばりばり使いこなしているでしょうし、またある人はvt100端末でカタカタとキーを叩いていたりします。私はXもたまに使いますが、mnews、ftp、mailx*1など、CUIでしか使いません。マルチに仕事をする際には、vt100端末をいくつも切り替えて使います。

前置きが長くなりましたが、自分の計算機の使用環境を自分で作り上げるために今回は、フリーソフトのコンパイルを取り上げることにしました。UNIX関係のソフトは、フリーなものがたくさんあります。*2このフリーソフトは、パソコンとは状況が違って基本的にソースコードしか提供されていません。ソースコードとは、ご存じのように実行できる形(バイナリ)ではなく、そのコードを用いてインストールするマシン向けにバイナリを作るという作業が必要です。パソコンの場合、たいていは即使える実行形式のみの配布というのが多いですが、UNIXではこの作業（コンパイルとインストール）を自分でやらなくてはなりません。なぜ七面倒くさいこのような仕組みになっているのかというと、ソースを提供することによって使う人が不具合を直せる、

＊1　新しいシステムに標準なのにこれを使うと（mailx -Hなど）画面が化けるという不具合があります。早急に何とかしてください。

＊2　代表的なものはGNUツールでしょう。GNUについては例えば、http://www.gnu.org/japan/などをご覧ください。後に述べるgccやpatch、makeなど主要なツールの根幹をなしているものはこのGNUによって生み出されています。これらのツールはほぼどのUNIXでも同じように使えるようにインプリメントされています。もし、入っていない場合はシステム管理者にお願いして一式インストールしてもらいましょう。ただ、管理者によっては、言い出しっぺが協力しなくてはならないこともあります。もっともこういうところがUNIXのおもしろさなのかもしれません。

ウイルスなどの心配がない、ソースを読むことによって勉強が出来る、などなど多くの利点があるからです。ただ、私のようなパソコンから入った素人には、このコンパイルという作業がとても大変でした(いや、今も大変です)。しかし、フリーソフトで環境を作り上げるとパソコンほど小回りはきかないものの、そこそこの環境が出来上がります。その一端をご紹介したいと思います。なお、C言語のコンパイル・リンクの仕組みとか、ソースコードにおける*.cや*.hなどのファイルがどうのという詳しい話はすべて省略させていただきます。もし詳しく知りたい方は、適当な参考書で調べてください。

総合情報処理センターのマシンの入れ替えで、NECのEWSから、HP(Hewlett Packard)製のサーバーに置き換わりました。まだいろいろと安定していないところもあるようですが、*3だいぶ使えるようになって来ました。owani8で使っていたいくつかのソフトをコンパイルしたので、それについて報告いたします。ソフトはfd、ish、lha、unzip、lessの5つです。

以下の記述に際して注意していただきたい点があります。まず、これらすべてを利用者が自前でコンパイルして使うということは、基本的に自由ですが、もし共通性の高いものであれば、システム管理者にお願いしてインストールしてもらうのが筋です。そうすれば、使いたいフリーソフトが最初から使えますし、新しく利用する人にとっても利便性が高くなるからです。ですがもしもすべてのユーザがこのようにし始めたらマシンのload averageは跳ね上がり、ディスク容量も多く消費してしまいますのでシステム管理者は困ってしまうでしょう。私がこれから書くのは、**「こうしましょう」と積極的にすすめているのではなく、使いたい人はどうぞ、ただし自己責任ですので何があっても保証はしません**、ということです。

2．フリーソフトについて

都合上、いつくかのレベルに分けます。なお、以下のソフトは、より新しいものが出ている可能性がありますので、ご注意下さい。

2.1 何もいじる必要のないソフト

1)fd clone

fdというのは、パソコン上でファイル管理ソフトとして有名なFDというソフトのクローンです。使ってみれば分かりますが、パソコンのFDとそっくりです。キー割り付けも設定ファイル（.fdrc）をいじればパソコンと同じようにできますし、ファイル名の最後にlzh、tar、gzなどといったものが付いているバイナリファイルの中味を他のソフトを利用して見ることができます。なかなかの優れものです。

a)準備

○FD-1.03j.tar.gzを探して来ます。archieを使ってサイトを探し、ftpでファイルを持ってくる

---

*3 さすがに今は起きなくなりましたが、ホームディレクトリが突然見えなくなったりすることがあり、困りました。またユーザに黙ってサーバーをネットワークから切り離して作業するなど、止めていただきたいところです。新システムの稼働間もないので内情は分からないではありませんが、何も知らないユーザを路頭に迷わせるようなことはしてほしくないものです。

という手順は、三上聖治先生の書かれたHIROINにありますので参照して下さい。*4

○作業するディレクトリを作ります。

これは、自分のホームディレクトリでもいいのですが、ファイルが結構ありますので、たとえば~/workなどというディレクトリを作成した方がよいかもしれません。

○環境の確認

これで私はハマりました。tsugaruの場合、pathの設定がうまく合っていなかったらしくコンパイル出来なくて泣きました。ちなみに私は以下のように設定しています。その道の方から「カレントディレクトリをpathに含めるとは何事か」というお叱りを受けるかもしれませんが、私の環境ということでお許し下さい。shellはtcshを使っています。なお、Cコンパイラとしては、gccやシステム標準のccがありますが、どちらでもよいと思います。

```
---------------------------------------------------------
setenv  PATH
/opt/bin:/usr/local/bin:/usr/contrib/bin:/usr/local/X11R6/bin:/usr/convex/bin:/bin:
/usr/bin:
/usr/ccs/bin:/etc:/usr/contrib/bin:/usr/lib:/usr/bin/X11:/usr/dt/bin:/sbin:/usr/sbin:
/opt/imake/bin:
/hm/shome/itan/stf/takanasi/tools:.
---------------------------------------------------------
```

b)ファイルの展開

適当なディレクトリでFD-1.03j.tar.gzを展開します。

```
        gzip -d FD-1.03j.tar.gz
        tar -xv FD-1.03j.tar
```

すると、FD-1.03jというサブディレクトリが出来ます。

c)コンパイルとインストール

カレントディレクトリをFD-1.03jに移して、make。

```
        make CC=gcc
---------------------------------------------------------
cp config.hin config.h
gcc  -DCCCOMMAND='"gcc"' -o mkmfsed mkmfsed.c
./mkmfsed  >  mkmf.sed
sed -f mkmf.sed Makefile.in  >  Makefile.tmp  ||¥
(rm -f Makefile.tmp; exit 1)
make SHELL=/bin/sh -f Makefile.tmp
<中略>
gcc -DFD=1 -DHPUX=1 -DDEFRUNCOM='"' /etc/fdrc'"'    -o fd main.o term.o pat
hname.o libc.o input.o shell.o info.o dosdisk.o dosemu.o rockridg.o builtin.o parse.
o kanji.o file.o apply.o archive.o
```

*4　三上聖治，1994，今更（に）UNIXへ・・・…落ちこぼれパソコンユーザのinternet利用入門…，HIROIN，4，37-50.あるいはhttp://dash.cc.hirosaki-u.ac.jp/koho/3/mikami.txt

```
tree.o command.o browse.o -lcurses
./kanjicnv -s -c fd.man fd.1
gcc -DFD=1 -DHPUX=1 -DDEFRUNCOM='"/etc/fdrc'"'    -o mkunitbl mkunitbl.c
./mkunitbl fd-unicd.tbl
make[1]: Leaving directory `/hm/shome/itan/stf/takanasi/work/FD-1.03j'
```

すると上記のようなメッセージが出ます。どうやらコンパイルはうまく行きました。./fdと叩くとめでたくfdが起動します（画面はパソコンのFDそっくりで、DOSの画面を見ているような気分になります）。このfdというバイナリファイルを例えば、~/toolsなどというディレクトリに入れておき、pathをそこに通しておくとコマンドラインからfdとするだけで使えるようになります。なお、標準のccだとコンパイルはうまく行くものの、「No TERMCAP prepared」というエラーが出てうまく動きませんでした。HP-UXがらみの問題だと思うのですが、ここらへんになるともう私には不可解な世界でして、誰か専門家に解説していただきたいのですが・・・。

2)ish

かの有名なishコンバータのUNIX版です。UNIX系では、バイナリをテキスト変換するにはuudecodeやuuencodeがありますが、パソコンではこれです。PPP接続でftpの使える今ではさすがに活躍の機会はないのですが、以前モデムでつないでいた時にファイルを転送する手段はテキストしかなく、とてもお世話になりました。

a)準備

○ish-1.11.tar.gzを探してftpで持ってくる

○作業するディレクトリを作る：省略

○環境の確認：省略

b)ファイルの展開

適当なディレクトリでish-1.11.tar.gzを展開します。

```
gzip -d ish-1.11.tar.gz
tar -xv ish-1.11.tar
```

すると、ish-1.11というサブディレクトリが出来ます。

c)コンパイルとインストール

カレントディレクトリをish-1.11にして、make。

```
make CC=gcc
```

```
gcc -O   -c crc.c -o crc.o
gcc -O   -c decode.c -o decode.o
gcc -O   -c encode.c -o encode.o
gcc -O   -c ish.c -o ish.o
ish.c: In function `main':
ish.c:65: warning: return type of `main' is not `int'
gcc -O   -c jis7.c -o jis7.o
```

```
gcc -O    -c jis8.c -o jis8.o
gcc -O    -c sjis.c -o sjis.o
gcc -O    -c njis.c -o njis.o
-n Loading ish ...
done
```

---

warningが出ていますが、大丈夫のようです。*5./ishと叩くと、かつてパソコンで見慣れた画面が出てきます。あとはfdと同じでここで出来たバイナリファイルishを~/toolsにでも放り込んでおけばよいようです。環境の設定はありません。なお、標準のccでも別なWarningメッセージが出ますがコンパイルはうまく行くようです。

2.2 ちょっと手をかけるもの

1)lha

これも説明するまでもないアーカイバの有名どころです。Mac版もあります。Windowsを使っている方でも知らないところでお世話になっていたりします。そのUNIX版です。

a)準備

○lha-114c.tgzを探してftpで持ってくる

○作業するディレクトリを作る：省略

○環境の確認：省略

b)ファイルの展開

適当なディレクトリでlha-114c.tgzを展開します。

```
gzip -d lha-114c.tgz
tar -xf lha-114c.tar
```

すると、lha-114cというサブディレクトリが出来ます。そのサブディレクトリに移動してMACHINES.eucというファイルを読みます。その中に次のような記述があります。

社名	Hewlett Packard
マシン	HP9000 s700
OS	HP-UX 8.05,8.07
Compiler	cc
変更事項	SWITCHES に -DUSESTRCASECMP を追加。Shift-JIS を使用する場合はさらに -DEUCを削除。
備考	hpux

---

tsugaruはHP-UXマシンですので、おそらくはこのSWITCHESを一部変更すればよいだろう

---

*5 本当はこのwarningの意味の解説をするべきなのでしょうが、プログラミングに疎い私としては荷が重いのでここでは何も触れません。

と予想がつきます。ただ、ＯＳのバージョンが11.00と新しいのでそのまま当てはまるかどうかは怪しいのですが、ともかくやってみましょう。変更するのはこのサブディレクトリにあるMakefileの中のSWITCHESです。中を見て行くと、ありました。

```
----------------------------------------------------------
#################################################################
# Makefile for LHa topdir
#                Mar. 2, 1992, written by Masaru Oki.
<中略>
#CC                          = cc
CC                           = gcc
SWITCHES          = -DNEED_INCREMENTAL_INDICATOR ¥
-DTMP_FILENAME_TEMPLATE="¥"/tmp/lhXXXXXX¥""
#MACHINE          = -DSYSTIME_HAS_NO_TM -DFTIME -DEUC
MACHINE           = -DSYSTIME_HAS_NO_TM _DEUC _DFTIME
<後略>
----------------------------------------------------------
```

適当なエディタでこのMakefileを開いて-DTMP_FILENAME_TEMPLATE="¥"/tmp/lhXXXXXX¥""の後に-DUSESTRCASECMPを加えます。そして保存します。ただし、よく見るとCCはgccになっていますが、たぶん大丈夫だろうという希望的な予測の下にそのままにしておきます。

c)コンパイルとインストール

カレントディレクトリで、make。

```
        make
----------------------------------------------------------------
make all in src...
make[1]: Entering directory `/hm/shome/itan/stf/takanasi/work/lha-114c/src'
gcc -O2 -fstrength-reduce -fomit-frame-pointer -DNEED_INCREMENTAL_INDICATOR -
DTMP_
FILENAME_TEMPLATE="¥"/tmp/lhXXXXXX¥"" -DUSESTRCASECMP -DSYSTIME
_HAS_NO_TM -DEUC -D
FTIME    -c lharc.c -o lharc.o
lharc.c: In function `main':
lharc.c:156: warning: return type of `main' is not `int'
gcc -O2 -fstrength-reduce -fomit-frame-pointer -DNEED_INCREMENTAL_INDICATOR -
DTMP_
FILENAME_TEMPLATE="¥"/tmp/lhXXXXXX¥"" -DUSESTRCASECMP -DSYSTIME
_HAS_NO_TM -DEUC -D
FTIME    -c lhadd.c -o lhadd.o
<中略>
gcc -O2 -fstrength-reduce-fomit-frame-pointer -DNEED_INCREMENTAL_INDICATOR-DTMP_
```

```
FILENAME_TEMPLATE="¥"/tmp/lhXXXXXX¥"" -DUSESTRCASECMP -DSYSTIME
_HAS_NO_TM -DEUC -D
FTIME   -c patmatch.c -o patmatch.o
gcc -o lha lharc.o lhadd.o lhlist.o lhext.o header.o append.o crcio.o dhuf.o extract.
o huf.o larc.o maketbl.o maketree.o shuf.o slide.o util.o patmatch.o
make[1]: Leaving directory `/hm/shome/itan/stf/takanasi/work/lha-114c/src'
make all in man...
make[1]: Entering directory `/hm/shome/itan/stf/takanasi/work/lha-114c/man'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/hm/shome/itan/stf/takanasi/work/lha-114c/man'
```

またもやwarningが出ていますが、致命的なエラーでないので大丈夫のようです（もし致命的なエラーだとコンパイルが途中で止まりますのでご安心ください）。あとは前のソフトと同じでlhaというバイナリファイルをどこかのディレクトリに放り込んでpathを通せば使えるようになります。システム標準のccでも別なWarningメッセージが出ますがコンパイルはうまく行くようですし、使えるようです。

2)unzip

これはzipファイルの展開を行うソフトです。パソコンでも使用されていて有名です。

a)準備

○unzip540.tar.gzを探してftpで持ってくる

○作業するディレクトリを作る：省略

○環境の確認：省略

b)ファイルの展開

適当なディレクトリでunzip540.tar.gzを展開します。

```
gzip -d unzip540.tar.gz
tar -xf unzip540.tar
```

すると、unzip-540というサブディレクトリが出来ます。そのサブディレクトリに移動してINSTALLというファイルを読みます。その中に次のような記述があります。

```
To compile UnZip, UnZipSFX and/or fUnZip (quick-start instructions):
==========================================
(1) Unpack everything into a work directory somewhere, and make sure you're
    in the main UnZip directory (the one with this file in it).
    * (See note below concerning line termination format used in the source
      distribution)
(2) Copy the appropriate makefile into the current directory, except under
    OS/2.
(3) Run your "make" utility on the makefile (e.g., "nmake -f makefile.msc").
```

(4) Try out your new UnZip the way you would any new utility: read the
docs first.
<後略>

要するにOSがUNIXの場合、サブディレクトリunixの中にあるMakefileをカレントディレクトリ（unzip-540）にコピーすればよいようです。

c)コンパイルとインストール

カレントディレクトリで、make。

make

おやおや何かメッセージが・・・。

If you're not sure about the characteristics of your system, try typing "make generic". If the compiler barfs and says something unpleasant about "timezone redefined," try typing "make clean" followed by "make generic2". If, on the other hand, it complains about an undefined symbol _ftime, try typing "make clean" followed by "make generic3". One of these actions should produce a working copy of unzip on most Unix systems. If you know a bit more about the machine on which you work, you might try "make list" for a list of the specific systems supported herein. (Many of them do exactly the same thing, so don't agonize too much over which to pick if two or more sound equally likely.) Also check out the INSTALL file for notes on compiling various targets. As a last resort, feel free to read the numerous comments within the Makefile itself. Note that to compile the decryption version of UnZip, you must obtain the full versions of crypt.c and crypt.h (see the "WHERE" file for ftp and mail-server sites). Have a mostly pretty good day.

自分の使っているシステムについて詳しく分からなければ、"make generic"とせよ、もし「timezoneの設定が二重になっている」とコンパイラがメッセージを出せば、"make clean"の後に"make generic2"とせよ、「_ftimeというシンボルが定義されていない」とコンパイラが言ったら"make clean"と打ってから"make generic3"とせよ、と書かれています。そしてUNIXのシステムでは、"make list"とすれば、このソフトをコンパイルする上でのシステム名が表示される、と書かれているようです。そこで

make list

Type "make <system>", where <system> is one of the following:

generic generic2 generic3 generic_zlib generic_shlib

386i 3Bx 7300 7300_gcc aix aix_rt amdahl amdahl_eft apollo aviion
bsd bsdi bsdi_noasm bull coherent convex cray cray_opt cyber_sgi
dec dnix encore eta freebsd gcc gould hk68 hp hpux isc isc_gcc isi
linux linux_dos linux_noasm linux_shlib linux_shlibz lynx minix
mips next next10 next2x next3x nextfat osf1 pixel ptx pyramid
qnxnto realix regulus rs6000 sco sco_dos sco_sl sco_x286 sequent sgi
solaris solaris_pkg stardent stellar sunos3 sunos4 sysv sysv_gcc
sysv6300 tahoe ti_sysv ultrix vax v7 wombat xenix xos

For further (very useful) information, please read the comments in Makefile.

なるほど、makeの後にsystem名を付ければいいらしい。たぶんsystem名はtsugaruの場合、hpuxだろう。そこで次のようにすると・・・

make hpux

```
cc -c -O -I.  unzip.c
cc -c -O -I.  crc32.c
<中略>
cc -c -O -I.  unix/unix.c
cc: "unix/unix.c", line 928: warning 604: Pointers are not assignment-compatible.
cc: "unix/unix.c", line 928: warning 563: Argument #2 is not the correct type.
cc: "unix/unix.c", line 989: warning 604: Pointers are not assignment-compatible.
cc: "unix/unix.c", line 989: warning 563: Argument #2 is not the correct type.
cc: "unix/unix.c", line 1025: warning 604: Pointers are not assignment-compatible.
cc: "unix/unix.c", line 1025: warning 563: Argument #2 is not the correct type.
cc -o unzip unzip.o crc32.o crctab.o crypt.o envargs.o explode.o extract.o fileio.
o globals.o inflate.o list.o match.o process.o ttyio.o unreduce.o unshrink.o zipin
fo.o unix.o -s
/usr/ccs/bin/ld: (Warning) At least one PA 2.0 object file (unzip.o) was detected.
 The linked output may not run on a PA 1.x system.
cc -c -O -I.  funzip.c
<中略>
cc -c -O -I.  -DSFX unix_.c
cc: "unix_.c", line 928: warning 604: Pointers are not assignment-compatible.
cc: "unix_.c", line 928: warning 563: Argument #2 is not the correct type.
cc: "unix_.c", line 989: warning 604: Pointers are not assignment-compatible.
cc: "unix_.c", line 989: warning 563: Argument #2 is not the correct type.
```

```
rm -f unix_.c
cc -o unzipsfx unzipsfx.o crc32.o crctab.o crypt.o extract_.o fileio.o globals.o
inflate.o match.o process_.o ttyio.o unix_.o -s
/usr/ccs/bin/ld: (Warning) At least one PA 2.0 object file (unzipsfx.o) was detect
ed. The linked output may not run on a PA 1.x system.
```

またしてもwarningが出ていますが、致命的ではないのでまあよしとします。どうやらちゃんと出来たようです。このunzipというバイナリを上のものと同じように好きなディレクトリに入れてpathを通せば出来上がりです。なお、gccでコンパイルするとやはりいつくかのwarningが出るようですが、どうにかコンパイルは終わり、同じようなバイナリファイルが出来ます。

2.3 かなり手をかけないといけないもの

ここからは、パワーユーザに助けていただかないととても手が出せません。たまたま私がやったらうまくいった（と思われる）ものを紹介します。less-332という画面表示のためのソフト（バックスクロールの出来るmore）です。tsugaruではmoreでも^uなどでバックスクロール出来ますが、通常のmoreにはバックスクロール機能はありません。また、patchが出ていて日本語が表示できるなど改良されています。システム標準のmoreでも不満はないのですが、他の計算機の環境ではlessが使えるようにしているので、環境の一貫性を持たせるためにあえてやってみました。この先は、なぜエラーなのか考えるのが楽しみな人を除いてはかなりマニアックな世界ですので、あまり深入りはしない方が身のためでしょう。

a)準備

○次のファイルを探してftpで持ってくる

```
less-332.tar.gz
less-332-iso241.patch.gz
less-332-iso242-243.patch.gz
less-332-iso243-244.patch.gz
less-332-iso244-245.patch.gz
less-332-iso245-247.patch.gz
less-332-iso247-248.patch.gz
```

○作業するディレクトリを作る：省略

○環境の確認：省略

b)ファイルの展開

適当なディレクトリでless-332.tar.gzを展開します。

```
gzip -d less-332.tar.gz
tar -xf less-332.tar
```

すると、less-332というサブディレクトリが出来ます。そこに2番目以降のファイルをコピーします。カレントディレクトリをless-332にして

```
gzip -d *.gz
```

とします。

次にpatch当てをします。*6具体的には次のようにします。

```
cat *.patch | patch
```

すると以下のようなメッセージが出ます。

```
patching file `Makefile.aut'
patching file `Makefile.dsb'
patching file `Makefile.dsg'
<中略>
patching file `version.c'
Hunk #1 FAILED at 631.
1 out of 1 hunk FAILED -- saving rejects to version.c.rej
patching file `multi.c'
patching file `unify.c'
patching file `version.c'
Hunk #1 FAILED at 633.
1 out of 1 hunk FAILED -- saving rejects to version.c.rej
patching file `unify.c'
patching file `version.c'
Hunk #1 FAILED at 634.
1 out of 1 hunk FAILED -- saving rejects to version.c.rej
patching file `charset.c'
patching file `help.c'
patching file `less.hlp'
patching file `less.nro'
patching file `multi.c'
patching file `multi.h'
patching file `opttbl.c'
patching file `unify.c'
patching file `version.c'
Hunk #1 FAILED at 635.
1 out of 1 hunk FAILED -- saving rejects to version.c.rej
patching file `multi.c'
patching file `unify.c'
```

*6 「パッチを当てる」と言います。ツギハギですね。誰かの書いたソフトに対してソースそのものを書き換え、改変した部分だけを独立してファイルにしておき、それをあるやり方で加える（あるいは削除する）ということを行うことです。それ専用のソフトがdiffやpatchです。フリーソフトとして流通しているものは、たいていGNUのdiffを使って作成されていることが多いようです。

```
patching file `version.c'
Hunk #1 FAILED at 638.
1 out of 1 hunk FAILED -- saving rejects to version.c.rej
```

表示のようにいくつかFAILEDと出るのですが、これはversionの管理の部分らしいので、目をつむります。*7version.c以外の部分はどうやらうまくpatchが当たったようです。

c)コンパイルとインストール

サブディレクトリless-332にあるINSTALLを読めば、configureをした後、makeをすればいいようです。そこで

```
configure
```

とすると次のようなメッセージが出ます。

```
creating cache ./config.cache
checking for gcc... gcc
checking whether the C compiler (gcc  ) works... yes
<中略>
checking for regcomp... using POSIX regcomp
decide to enable the MSB of non ASCII characters
updating cache ./config.cache
creating ./config.status
creating Makefile
creating defines.h
```

どうやら、これで準備完了です。いよいよmake。

```
test ! -f stamp-h || CONFIG_FILES= CONFIG_HEADERS=defines.h ./config.status
touch stamp-h
gcc -I. -c  -g -O2 main.c
gcc -I. -c  -g -O2 screen.c
In file included from /usr/include/sys/stream.h:39,
                 from screen.c:86:
/usr/include/sys/user.h:334: parse error before `physical'
/usr/include/sys/user.h:376: parse error before `}'
make: *** [screen.o] Error 1
```

---

*7　本当はこの理由とか詳しく調べないとまずいのですが、本体とはあまり関係ないので無視しました。

ところがエラーになってしまいました。*8patch当てに失敗したのかと思いましたが、エラーが出ているのは、screen.cです。これはきちんとpatchが当たっているものですので、patch当ての失敗ではないようです。とするとエラーの起きているstream.hの中味を見てコメントアウトするなり、あるいはuser.hを編集するなりする必要がありそうです。

後者については、その中味まで追求して編集して直すところまで手を入れるべきでしょうが、システム管理者ではないし，OSの標準で持っているheaderは少なくともメーカーが責任を持っているのですから、素人がいじれるわけがありませんし、いじってはいけません。困ってしまいました。仕方ないので振り出しに戻ります。詳しい説明は省略しますが、この場合、screen.cの中でstream.hがincludeされて、その結果としてuser.hがincludeされているようです。とすれば、「このstream.hをincludeしないように出来れば何とかなるのではないか」と考えることにします。configureした際に出力された結果を眺めてみると、確かにstream.hの有無を調べています。ということは、これがないものとコンパイラオプションでコンパイラに見せかければ、ないなりにコンパイルを終了するはずです。このconfigureの結果はdefines.hというファイルに書き出されますが、その中を見ていくと

```
/* Define if you have the <sys/stream.h> header file.  */
#define HAVE_SYS_STREAM_H 1
```

という部分が見つかります。事の詳細は省略しますが、この最後の数字の１は、システムにstream.hがあるということを示しています。ここを0にすれば、おそらくはstream.hがないものとして処理されるに違いありません。さっそく直して、makeします。すると

```
gcc -I. -c  -g -O2 main.c
gcc -I. -c  -g -O2 screen.c
gcc -I. -c  -g -O2 brac.c
<中略>
gcc -I. -c  -g -O2 multi.c
gcc -I. -c  -g -O2 unify.c
gcc  -o less main.o screen.o brac.o ch.o charset.o cmdbuf.o command.o decode.o e
dit.o filename.o forwback.o help.o ifile.o input.o jump.o line.o linenum.o lsystem.
o mark.o optfunc.o option.o opttbl.o os.o output.o position.o prompt.o search.o si
gnal.o tags.o ttyin.o version.o    multi.o unify.o -lPW -lgen    -lncurses
gcc -I. -c  -g -O2 lesskey.c
gcc  -o lesskey lesskey.o version.o
```

*8　実はこのソフトの場合、ごく最近に行われたOSのpatch当て以前にはちゃんとコンパイルが出来ていました。確証はないのですが、そのpatchを当てた部分に問題があるのではないかと思われます。

```
gcc -I. -c  -g -O2 lessecho.c
gcc  -o lessecho lessecho.o version.o
```

となって、めでたくエラーなしでコンパイルが出来ました。あとは同じように適当なディレクトリに入れてpathを通せばいいのですが、実はこのソフトはシステム管理者によって/usr/local/binにインストールされていました。分かっていれば、こんなに苦労しなくてもよかったのです。pathに/usr/local/binを加えておけば済むことでした。

3．最後に

というわけで、一通り見てきましたが、いかがでしたでしょうか。コンパイルの話ばかりで、実際に使ったところまで書けませんでしたが、そこそこに便利な環境を作り上げることが可能であるということがお分かりになられると思います。ほしいと思うようなソフトは案外フリーでnet news（例えばfj.sourcesなど）に流れていたりするものです。また、口コミで使っている人から教えてもらうこともあります。ちょっと注意しているとおもしろいものが見つかるかもしれません。そういった情報もユーザ間で交換して利用環境をよりよいものに出来るのではないでしょうか。これこそUNIXの「協調と奉仕の精神」だと思います。

それから、このような作業に慣れて来るとどこをどう見てエラーにならないようにするか分かるようになる人もいるようですが、私はとてもそこまで行かないのでnet newsを見たり、一生懸命にpatchファイルを探します。それでも分からないことが多くあります。正直言って誰に聞けばいいのか困っています。新システムに関しては、総合情報処理センターに聞けばいいと思うのですが、何せOSが新しすぎてフリーソフトが前のシステムと同じように動かない（コンパイル出来ない？）らしく、メールシステムはそこそこ安定してきたようですが、まだまだ先は長いようです。一利用者としてはもう少しシステムを安定させてほしいし、いろいろなソフトを使いたいと思います。すでにシステムを入れ替えて公式には2ヶ月が過ぎようとしているのですから、管理者の方々にはもう少し頑張っていただきたいところです。

最後は文句とも愚痴ともつかなくなってしまいましたが、計算機は「使ってナンボ」のものです。どんどん使って要求をセンターに出すことがやはり、環境の向上につながることだと思いますので、これからも一利用者として総合情報処理センターに計算機システムへのいろいろな要求を出していきたいと思います。

○謝辞

ソフトのコンパイルやOSに関して有用なアドバイスをいただいた総合情報処理センターの須藤勝弘さん、またこの文章を書くきっかけを与えていただいた附属病院医療情報部の三上聖治先生に感謝いたします。