

システム工学概論

1. ソフトウェア開発環境の整備

山田泰司

`taiji@aihara.co.jp`

株式会社あいはら 研究開発チーム

開発環境の整備のポイント

マルチプラットフォーム (Unix, Mac, Windows)

GUI な統合開発環境 (Visual C++, C++ Builder など) は
不要

CUI なオープンソースプロダクト、ツールの組み合わせ
で開発環境を整備

Unix 端末 (Windows の場合は Cygwin を利用)、シェル

GNU make, gcc

エディタ (例えば、GNU emacs)

広く移植され、十分に枯れたツールなら、

プラットフォームが変わっても同様に開発ができる。

それぞれのツールの寿命が長く、ノウハウの変遷に振
り回されにくい。

make コマンドとは

プログラムソースファイルから実行ファイル等へのターゲット生成規則から、ソースファイルの変更日時（タイムスタンプ）がターゲットより新しい、もしくはターゲットがまだ生成されていない場合に、実行ファイル等のターゲットファイルを生成するユーティリティ

(例題 1) make コマンドの実践 1

以下のようなプログラミング言語 C プログラム、ファイル名「hello.c」があるとする。

```
#include <stdio.h>
int main(void)
{
    printf("hello\n");
}
```

make コマンドを使って実行ファイル「hello」を作成せよ。

(例題1)の解答

コマンドラインで

```
$ make hello
```

make コマンドの学び方

make コマンドのマニュアルを読む (MANPAGE)

```
$ man make
```

make コマンドのマニュアルを読む (GNU Info)

```
$ info make.info
```

make コマンドのデフォルトの規則と変数を知る

```
$ make -f /dev/null -p
```

make コマンドのデフォルトの規則と変数

```
$ make -f /dev/null -p | less
```

```
      :  
CC = cc  
      :  
LINK.c = $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)  
      :  
%: %.c  
      $(LINK.c) $^ $(LOADLIBES) $(LDLIBS) -o $@  
      :
```

```
hello: hello.c  
      cc hello.c -o hello
```

```
cc hello.c -o hello
```

(例題 2) make コマンドの実践 2

以下のようなプログラミング言語 C プログラム、ファイル名「hello2.c」があるとする。

```
#include <stdio.h>          /* printf */
#include <math.h>           /* sqrt */
int main(void)
{
    printf("sqrt(2)=%g\n", sqrt(2));
}
```

make コマンドを使って実行ファイル「hello2」を作成せよ。つまり、以下のような事を make コマンドで行なうには、という例題。

```
cc hello2.c -lm -o hello2
```

(例題2)の解答

コマンドラインで

```
$ make LDLIBS=-lm hello2
```

(例題3) 簡単なmakefileを書く

makefile を書いて (例題1)(例題2) を簡略化せよ。

(例題3)の解答

以下のような makefile、ファイル名「makefile」を用意する。

```
LDLIBS=-lm
```

```
SRC=\
```

```
hello.c \
```

```
hello2.c \
```

```
EXE=$(SRC:.c=)
```

```
all: $(EXE)
```

```
$ make
```

端末、コマンドライン、シェルについて

CUI ... キャラクタ (文字) ユーザーインターフェース

Unix(Linux, Solaris, Mac OS X) の**端末**で扱う

Windows の Cygwin **端末**で扱う

Windows の**コマンドプロンプト**や**DOS プロンプト**も
CUI の**一種**

Unix の**シェル** sh(ksh, bash, zsh) や csh(tcsh) **上で扱う**

Windows Vista の PowerShell

慣れれば作業時間が短い。作業内容が再現しやすい。作業内容の伝達がしやすい。

グラフィカル・シェルについて

GUI ... **グラフィカルユーザインターフェース**

Windows

Mac OS X ... Quartz

Unix(Linux, Solaris) ... GNOME, KDE, Motif, X11

直観的に作業することが出来る。しかし、作業を再現するにはほぼ同じだけの時間がかかる。作業内容の伝達が比較的面倒である。

CUIとGUIの融合した環境について

Unix(Linux, Solaris) ... GNOME, KDE 上のターミナル、
X11 上の xterm, kterm

Mac OS X ... ターミナル.app、X11 上の xterm, kterm

Windows ... Cygwin/X11 の xterm, kterm

GUI と CUI の両方を使えた方が、どちらの利点も生かすことができる。

Unix(Linux, Solaris)

Mac OS X ... テキスト
Emacs, emacs

Windows ... メモリ
emacs

ファイル編集環境、エディタについて@

エディタとは、コード
を取られず、とにかく
ストであり、慣れてい

もし慣れているものが

プログラミングの点で最

私の開発環境について...

1. Basic
2. DOS ... Turbo C
3. Solaris, Linux ... gcc, make, X11
- 参考 (1) @ 4. Windows 95 ... C++ Builder, Visual C++
5. Windows XP ... 同上、Cygwin, gcc, make, X11
6. Mac OS X ... gcc, make, X11, Xcode

私の開発してきたテーマについて...

1. ウィンドウシステムもどき
2. アセンブラ、仮想マシンエミュレータもどき
3. X11, PostScript グラフィックライブラリ

- 参考 (2)** @
4. 数値計算と可視化
 5. 商用グラフィカル数値計算アプリケーション
 6. スпам対策プログラム
 7. オープンソースのパッチ群 (gs, xdvi, ptetex, mew, smtpproxy, pop3proxy, wipe-out98)

この講義について

使い慣れたノートパソコンを持ち込んで構いません。

但し、マルウェア等の感染には気をつけて下さい。

成績の評価は、提出してもらう「プログラム自慢」で行ないます。

何度でも提出してもらって構いません。

提出はソースコードとビルドに必要なものを
taiji@aihara.co.jp まで。

プログラムには著作権があります。

実習の目的は、CUIでの開発環境を整え、慣れる事、

そして、オープンソースの活用が出来るようになる事、

最終的には、オリジナルのプログラムが書けるようになる事。

*シェルの学び方

csh(tcsh) 系は学ぶ必要はありません。なぜならスクリプト言語として欠陥が多く、メリットが Bourne shell(sh, ksh, bash, zsh) と比べて皆無だからです。そう言った理由から、これから学ぶのであれば Bourne shell を推奨します。

bash のマニュアルを読む (Linux, Mac OS X)

```
$ man bash
```

zsh のマニュアルを読む (Linux, Mac OS X)

```
$ man zsh
```

ksh のマニュアルを読む (Solaris)

```
$ man ksh
```

むしろ、使いながら慣れる事が重要でしょう。

* (例題3) シェルの実践 1

以下のようなファイル群があるものとする。

```
$ ls
```

```
result1.jpg result2.jpg result3.jpg result4.jpg
```

すべてのファイル名を「resultA ~ .jpg」へ変更せよ。

* (例題3) の解答

```
$ for f in *.jpg; do  
> mv "$f" "`echo $f | sed 's/result/resultA/'`"  
> done
```