第 1 回 プログラミング基礎ゼミ

ゼミ担当者　：　金美和，澤田淳二，長野林太郎
指導院生　：　佐野正樹，實田健
開催日　：　2002 年 5 月 10 日

ゼミ内容: 本ゼミでは，研究を快適に進めるためのプログラミングについて紹介します．知的デザイン研究室では，Cambria や Gregor といった PC クラスタを用いて多くの計算を行うため，計算結果をまとめ，編集する方法を習得することはとても有益であります．詳しい内容としては，ヘッダファイルの作り方，makefile の作り方，シェルスクリプトの書き方，Perl による編集方法についての実習を行います．

1　実習の準備

まず forte にログインして下さい．

```
ssh アカウント名@202.23.147.73
```

次に子ノードを指定します．子ノードには 001(〜004)-101(〜109) があるので，各自好きな子ノードを選択して下さい．

```
rsh forte001-101
```

続いて，自分のホームディレクトリの下に新しいディレクトリ (test) を作成し，そこに,"kim/program"をコピーして下さい．

```
mkdir test
cp -r /home/kim/program test
```

2　作成するプログラムについて

2.1　プログラムの内容と構成

今回作成するプログラムは，式 (1) で表された目的関数 $f(x,y)$ の値を最小にする x,y をランダムサーチを用いて探索するものです．

$$f(x,y) = x^2 - 8x + y^2 - 6y + 18 \qquad (1)$$

この関数は，Fig. 1 のような形状になっています（この図では等高線のみ表示しています）．図中で，×印の部分が最適解になっています．

作成するプログラムは次の 3 つのソースファイルから構成されています．

main.c	ランダムサーチのメインルーチン
gen_rand.c	乱数発生ルーチン
func.c	目的関数ルーチン

このプログラムは乱数の種を引数で与えることにより，その種で乱数を初期化し，x,y ともに，−100〜100 の間の乱数を発生させることでランダムサーチを行います．それを一定回数分繰り返し，最後に乱数の種と最適解とその値を出力しています．出力形式は Fig. 2 のようになります．

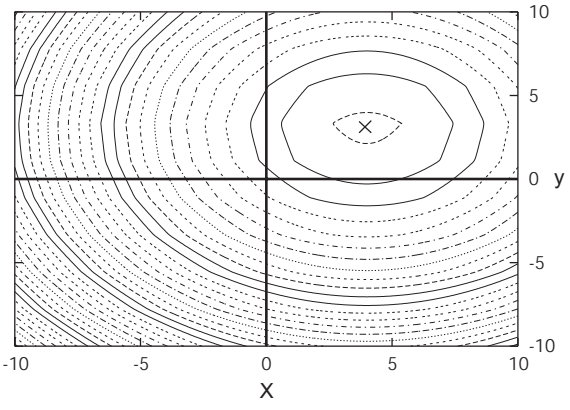

Fig. 1　目的関数の形状

```
seed,x,y,f
90,4.065563,3.084983,-6.988479
```

Fig. 2　プログラムの出力形式

Fig. 2 のように各値をコンマで区切ったものを CSV[1] 形式と呼びます．この形式で結果を出力しておけば，出力結果を Excel に取り込んで編集できるので，非常に便利です．プログラムでは，与えられた乱数の種に応じて，“rs_[seed].csv”（[seed] は，与えた乱数の種の値）というファイルを作成し，そのファイルに結果を出力しています．こうすることで，後から見たときにどの種を用いてプログラムを走らせたかがわかります．

2.2　ヘッダファイルの使い方

先ほど説明しましたように，今回作成するプログラムは複数のソースファイルで構成されています．プログ

[1] Comma Separated Value

ラムが一つのソースファイルだけでなく，複数のソースファイルで構成されるようになると,「別のソースファイルに書かれた関数を使う」とか「プログラム全体で使用する変数を定義する」ということが必要になります．その場合に，そのような関数の呼び出し方法や変数定義を記述しておくものがヘッダファイルです．今までにもプログラムを書くときには，

```
#include <stdio.h>
```

という行を書いていたと思いますが，この stdio.h がヘッダファイルです．この中に画面表示のための printf 関数の呼び出し方法などが書かれているため，プログラム中で printf 関数が使えたわけです．

ヘッダファイルを，<>で囲むと，ヘッダファイルはデフォルトの検索パスから検索されます．””で囲んだ場合は，カレントディレクトリから検索されます．よって，自分で作成したプログラム固有のヘッダファイルは，”” で囲むようにしてください．

今回は，main.c から gen_rand.c や func.c で定義された関数を使うために，rsearch.h というヘッダファイルを作成し，その中に関数の呼び出し方法を記述しておきます．こうすることで，コンパイラはどのように関数呼び出しを行えばいいかがわかります．こうしないと，コンパイラはデフォルトの呼び出し方法を用いようとするので，実行しても正しい結果が得られなくなります．

3 プログラムのコンパイル

ソースファイルが書けたとして，次にそれらをコンパイルし，実行ファイルを作ります．その場合，次のようなコマンドを入力する必要があります．

```
gcc -c main.c
gcc -c gen_rand.c
gcc -c func.c
gcc -o rsearch main.o gen_rand.o func.o
```

プログラムのソースコードが一つのファイルのうちは，単純にコマンドラインからコンパイラを用いてプログラムをコンパイルしても問題はありません．しかし，プログラムが複数のソースコードで構成されていると，それぞれのソースコードをまずコンパイルしてオブジェクトファイルを作成し，最後にオブジェクトファイルをリンクして実行ファイルを作成するという手順が必要となります．プログラムにバグがあってソースファイルを編集した場合，どのファイルが更新されたかに応じて，コンパイルを行い，リンクをしなおさなければいけません．プログラムを構成するファイルが多くなってくると，その作業がより煩雑になります．その都度，構成ファイルすべてをコンパイルすれば間違いはないのですが，その場合，更新されていないファイルもコンパイルされるために時間の無駄が生じます．ほかにも，毎回コマンドラインからコンパイラに指定するオプションを入力するのも手間がかかります．

そこで，プログラムの依存関係とプログラムのコンパイル方法を記述することで，更新されたファイルに応じて自動的にソースファイルをコンパイルし，最終的な実行ファイルを作成するためのプログラムが，make と呼ばれるプログラムです．

make を使うためには，プログラムの依存関係を記述した “makefile” というものを記述する必要があります．makefile は，

```
ターゲット: 依存ファイル群……
    作成方法
```

のように作りたいファイル（ターゲットと呼びます）とその構成ファイル（依存ファイルと呼びます）を ‘:’ で区切って記述します．ターゲットの指定は必ず，行の先頭から書いてください．次の行には，ターゲットをどのように作ればいいかを書きます．この行はタブなどで先頭に空白を空けて記述します．作成方法を省略すると，デフォルトの作成ルールが用いられます．依存ファイルのどれかのタイムスタンプがターゲットよりも新しい場合に，依存ファイルが更新されたと認識され，ターゲットが作り直されます．

今回作成した makefile を Fig. 3 に示します．

```
rsearch: main.o gen_rand.o func.o
    gcc -o rsearch main.o gen_rand.o func.o
main.o: main.c rsearch.h
gen_rand.o: gen_rand.c rsearch.h
func.o: func.c rsearch.h
```

Fig. 3　makefile の内容

Fig. 3 では，rsearch を作るためには，main.o, gen_rand.o, func.o が必要であり，作成するには，“gcc -o rsearch main.o gen_rand.o func.o” というコマンドを実行すればよい，ということが書かれています．

この makefile は Fig. 4 のような木構造のイメージで，最終目標である rsearch を作るためにはどのような依存関係があるかが記述されています．Fig. 4 では，あるターゲットはその子となっているファイルが依存しているということを示しています．

makefile をソースファイルと同じディレクトリに用意した上で，コマンドラインから，

```
$ make
```

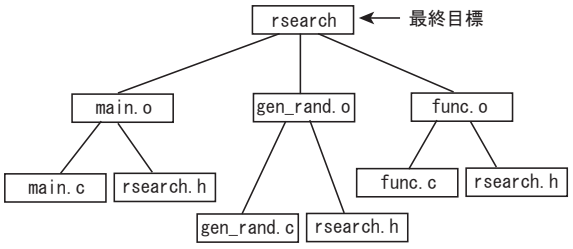

Fig. 4　makefile の依存関係イメージ

と入力すると，make が依存関係と更新状況を調べて自動的に最終目標である rsearch を作成してくれます．

4　プログラムの実行

ではプログラムの実行を行います．ここでは引数を 10,30,50,70,90 として”rsearch”を 5 試行実行していただきます．

```
./rsearch 引数
```

最後に各自”ls”と入力して，CSV ファイルが保存されているか確認してください．

4.1　シェルスクリプトの活用

ここでは，シェルスクリプトを用いて複数回プログラムを実行をさせる方法を紹介します．通常，研究で使用するプログラムはここで例として挙げているソースよりもずっと複雑で長いため，1 試行に時間を要します．よって複数の試行を行うとき，1 試行の結果を待って次を実行するのはとても面倒です．そこでシェルスクリプトを用いて複数の試行をまとめて実行させます．Windows のバッチファイルのように，コマンドを登録してまとめて実行できるのがシェルスクリプトの特徴です．このゼミで用いたシェルスクリプトの内容は以下の通りです (Fig. 5)．

```
#!/usr/bin/bash

seed=0

 while [ $seed -lt 100 ]
 do
   ./rsearch $seed
   seed=‘expr $seed + 20‘
 done

 ls
```

Fig. 5　shell のソース

”#!/usr/bin/bash”では処理するシェルのパスを指定します．シェルスクリプトには sh, bash，ksh など種類があり (Table1)，ここでは bash を指定しています．

Table 1　シェルスクリプトの種類

sh	最初に使われはじめたので，どの UNIX にも必ず載っている．
csh	プログラミン機能が C 言語に似ているため csh と呼ばれる．インターフェイス機能が評価されている．
tcsh	csh の全ての機能を持ちさらにコマンドライン編集機能やスペル訂正機能，コマンド補完機能などを持つ．
ksh	ベル研の David Korn により開発されたシェルスクリプト．フリーではない．
bash	bsh と互換性を持たせ，機能を強化したシェルスクリプト．Linux の標準シェルになっている．
zsh	bash 互換で色々なシェルの機能を取り入れさらに独自機能追加をしたシェル．最も新しいシェルスクリプトである．

4.2　shellscript の特徴

shellscript 最大の特徴はコマンドをそのままプログラムに記述できることです．fig1 では"./rsearch $seed"や”ls”がこのコマンドに相当します．参考として上記の内容のプログラムを C 言語で記述すると次のようになります (Fig. 6)．fig5 と fig6 を比べるとシェルスクリプトの利便さを分かっていただけると思います．

```
int seed
char cmd[80]

seed=0

 while(seed < 100) {
   sprintf(cmd,"./rsearch %d",seed);
   system(cmd);
   seed = seed + 20;
 }
system("ls");
```

Fig. 6　C 言語で記述した shell のソース

4.3 windows ユーザへ

シェルスクリプトは UNIX におけるシェルプログラムであるため，windows ユーザは，シェルスクリプトを利用するには Cygwin[2]というアプリケーションをインストールする必要があります．Cygwin を install すると WINDOWS の DOS 画面や CGI プログラムから有る程度 UNIX コマンドを使うことが出来るようになります．

5 CSV ファイルの Perl を用いた編集

5.1 CSV ファイルのの編集

プログラムの実行を行うと，多数の CSV ファイルが出力されます．これは，Windows 上では Fig. 7 のような Excel ファイルの形式になります．

これまでで，複数のプログラムのコンパイルや実行を簡単に行う方法を学びました．しかし，実際に実験を行う際は，実験データが出力されると，そのデータを編集する必要性が生じます．具体的には実験に応じてデータの平均値・最小値・中央値などを求める必要があります．そのような値をデータを見ていちいち手作業や電卓で計算すると，非常に煩雑になります．また，データが複数のファイルにまたがるため，データの編集のためにすべてのファイルを開かなくてはいけません．そこで，ここではそのようなデータの編集をプログラムを用いて自動的に行う方法を学びます．なお，データ編集に用いるプログラミング言語としては，ここでは Perl を用います．Perl を用いる理由は，Perl が CSV ファイルのようなテキストを処理するのに非常に向いている言語だからです．

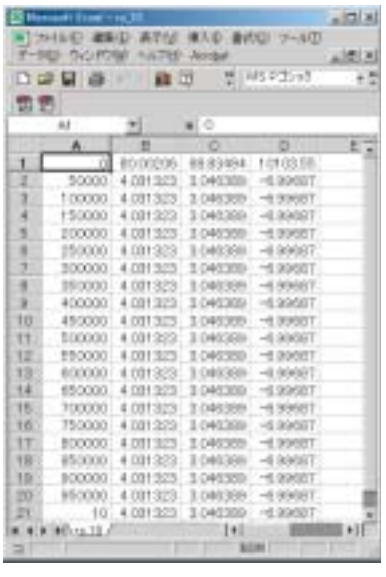

Fig. 7　Windows 上における CSV ファイル

[2]http://www.cygwin.com/

5.2 用いる Perl プログラム

ここでは，Perl のプログラムの方法を解説します．ただし，ここでは Perl 言語の中身については触れません．Perl は非常に多機能な言語でありさまざまな処理を行うことができますので，ぜひ勉強をしてみてください．

今回は，いままでに出力された 5 つのファイル (rs_10.csv〜rs_90.csv) の結果部分のみ (この場合は，最後の行) を出力し，それの平均を求めるプログラムを例に行います．以下に，用いた Perl のプログラムソースと，その内容の簡単な解説を示します．実際のプログラムを書くときの参考にしてください．なお，複雑なプログラムを書くこともできますので，それについては参考文献を見てください．

```
#!/usr/bin/perl

while($dummy = <rs_[0-9]*\.csv>){
$datafile[$i++] = $dummy;
print $datafile[$i-1]."\n";
}

$file_num=$i;

$tseed=0; $tx=0; $ty=0; $tf=0;

open(OUTCSV,">rs_ave.csv");
for($i=0; $i<$file_num; $i++) {
open(INFILE,$datafile[$i]);
while(<INFILE>) {
if(eof) {
($seed,$x,$y,$f) = split /,/;
$tseed += $seed; $tx += $x;
$ty += $y; $tf += $f;
print OUTCSV "$seed,$x,$y,$f";
}
}
}
print OUTCSV "$tseed,$tx,$ty,$tf,total\n";
$tseed /= $file_num; $tx /= $file_num;
   $ty /= $file_num; $tf /= $file_num;
print OUTCSV "$tseed,$tx,$ty,$tf,ave";
```

以下では，プログラムの内容の簡単な解説を行います．

```
#!/usr/bin/perl
```

これは，Perl という言語の中のどの部分を用いるか，というものを記述したものです．これは，わかりにくい場合は，Perl プログラムには必ず載せるという「おまじな

い」みたいなものと思ってもらってかまいません．

```
while($dummy = <rs_[0-9]*\.csv>){
$datafile[$i++] = $dummy;
print $datafile[$i-1]."\n";
}
```

Perlにおいては，$からはじまるものは変数として扱われます．なお，C言語のように変数の宣言は必要ありません．ちなみに@ではじまるものは，配列を表します．

また1行目の< >内で用いられているのは正規表現です．ここでは，”rs_1桁以上の数字”の形のものはすべて配列に格納されます．なお，配列に格納されたファイルは，実行時に3行目の print 文で表示されるようになっています．

```
open(OUTCSV,">rs_ave.csv");
for($i=0; $i<$file_num; $i++) {
open(INFILE,$datafile[$i]);
while(<INFILE>) {
if(eof) {
($seed,$x,$y,$f) = split /,/;
$tseed += $seed; $tx += $x;
$ty += $y; $tf += $f;
print OUTCSV "$seed,$x,$y,$f";
        }
    }
}
```

1行目の open の部分はCSVファイルに出力するための出力ファイルを OPEN するものです．OUTCSV は出力ファイルへのファイルハンドル名[3]です．これによって，ここでは実行結果が rs_ave.csv というファイルに出力されます．なお，ここの部分は変数を用いることもできます．3行目の open は，逆に入力ファイルを OPEN するもので，INFILE が入力ファイルのファイルハンドル名を表します．6行目の split という命令は，”,”を区切り文字として，4つに分割を行う命令で，C言語にはない命令です．

5.3 プログラムの実行

Perl を実行するには，コマンドラインに”perl ファイル名”と入力してください（大文字と小文字は区別します．以下同じ）．今回は，sample.pl というファイルですので，以下のように記述します．

```
perl sample.pl
```

実行すると，出力ファイルが作成されるはずです．”ls”を用いてファイルができていることを確認してください．

[3]ファイルに関するアドレスを示したもの

5.4 Windows から実行する場合

ここでは，Windows 上から Perl を実行する方法を説明します．

ここでは，作成した Perl 言語の Windows 上の実行方法を説明します．Perl 言語を実行するためには，Perl 言語の処理系が必要になりますが，個人のコンピュータに Perl の言語をインストールするには手間がかかります．そこで，ssh を用いて forte に移動し，そこで Perl プログラムの実行を行います．

この部分における手順の概要は以下のようになります．

1. WinSCP を用いて，forte にファイルをアップロードする．
2. SSH を用いて，forte 内で Perl プログラムを実行する．
3. WinSCP を用いて，forte から出力ファイルをダウンロードする．

この手順のイメージ図を Fig. 8 に示します．以下，それぞれの手順について説明します．

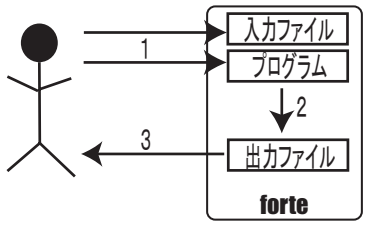

Fig. 8　ファイル実行のイメージ図

5.4.1 WinSCP を用いたファイルのアップロード

1. WinSCP をダブルクリックして起動する．
2. WinSCP の接続先を Fig. 9 のように，forte (forte.doshisha.ac.jp) に設定する．

Fig. 9　WinSCP の接続先の設定

3. ホームページにファイルをアップロードする方法と同様に，forte に「Perl のプログラム」（普通は秀丸エディタなどで記述します．拡張子は`**.pl` にしてください）と「編集を行いたいデータファイル群」の２つをアップロードします（適当にフォルダを設けて行うとわかりやすいです）．

もう一度，SCP を使いますので，ここで WinSCP を閉じないでください．

5.4.2　Perl プログラムの実行

1. コンピュータから SSH を起動します．
2. 接続先を Fig. 10 のように直接入力をして，forte (forte.doshisha.ac.jp) に設定します．

Fig. 10　SSH の接続先の設定

3. ユーザ名とパスワードを聞かれるので，入力をします．
4. 通常と同じく Fig. 11 のように起動しますので，Perl プログラムをアップロードしたフォルダまで移動します．

Fig. 11　SSH の起動

5. 移動したら，コマンドを用いて Perl を実行します．（UNIX コマンドを使いなれている人は，ファイルの修正を行うときに vi などを使ってファイルの修正を行うと，デバッグが簡単になります．コマンドは，”`vi` ファイル名”として起動します．）

5.4.3　出力ファイルのダウンロード

1. WinSCP を起動していた場合は，いったんフォルダを移動して再び戻ると，出力ファイルが生成されているのがわかります．それをダブルクリックすると，出力ファイルを自分のパソコンにダウンロードすることができます (Fig. 12)．

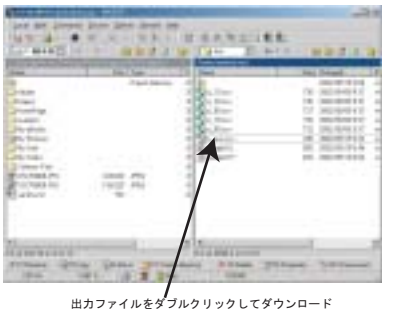

Fig. 12　出力ファイルのダウンロード

2. ダウンロードしたら，ファイルを見て，出力結果を確認する．Windows を用いている場合は，Excel 形式で CSV ファイルが出力されるので，グラフなどの作成も容易である（Fig. 13）．

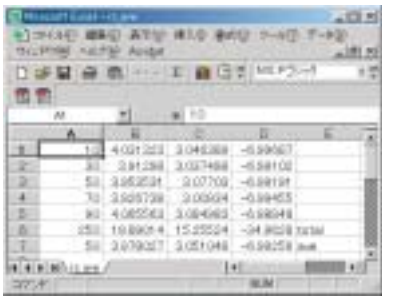

Fig. 13　出力ファイルのようす (Windows)