C言語によるプログラミング 1
・序論およびgccについて

A Pedestrian Approach
to
the C Programming Language

目 次

1 序論および gcc について **1-1**

1.1 はじめに 1-1
1.2 gcc のつかいかた 1-1
1.3 ソースコード再訪 1-3
1.4 g++の操作の意味 1-4
1.5 他の OS 上での C プログラミング 1-6
1.5.1 序論 1-6
1.5.2 具体的方法 1-6
1.5.3 漢字コードの問題 1-8

図 目 次

表 目 次

1　序論および gcc について

1.1　はじめに

1）なにをやるのか？

この実習では、C 言語によるプログラミングの実習をおこなう。ここであつかうのは、「標準 C プログラミング」とよばれているもののうちのほんの初歩の部分である。「標準 C プログラミング」はどの OS(Operating System、基本ソフト) で C プログラミングをおこなう場合でも必須となる基礎のコースである。

2）プログラミング言語をつかわなければプログラムをつくることができないのか？

プログラミング言語をまったく、あるいはほとんどつかわなくてもプログラムをつくることはできる。そのためのソフトがあるからである。しかし、たとえかんたんなプログラムでも、いちどは自分でつくると、プログラミングでなにをやっているかの理解が深まるかとおもう。まぁ、料理のようなものじゃないかな？

3）ウインドウ・プログラミングとの関係

さて、MacOS や Windows8/Windows7/Vista などの OS、あるいは UNIX+ X-window system(Linux などのパソコン用の UNIX もふくむ) では、ウインドウを開いたり閉じたり、動かしたり、アイコン (icon) やマウス (mouse) をつかった操作で人間とコンピュータがやりとりをする。このような GUI(Graphical User Interface) 仕様の OS 上で動く応用ソフト (Application soft) をつくるには、ウィンドウやマウスの操作をふくむようにプログラムを記述しなければならない。具体的には、OS にくみこまれている GUI にかんする関数群をどうつかうか、とういうことになる。とうぜん、この部分のプログラムのしかたは OS に依存することになる。[1]これとは対照的に、「標準 C プログラミング」は、GUI の操作とは関係のない、プログラムの骨子にあたる部分を担当する。

「ウィンドウ・プログラミング」＝「標準 C プログラミング」＋「ウィンドウやマウスの操作をふくむようなプログラム」

というところであろうか？

具体的には、『端末』でのプログラム起動やファイル操作の標準的な環境であるターミナル・ウィンドウにおいて、文字入力をつかってコンピュータとやりとりするようなプログラムをあつかうことになる。すなわち、われわれのおこなうのは、CUI(Character User Interface) 環境でのプログラミングである。無骨な方法かもしれない。しかし、それゆえに、素朴で強靭、本質的である。

1.2　gcc のつかいかた

この授業では、ソースプログラムを翻訳して機械にわたすコンパイル用のソフトとして、‘gcc’ (the GNU Compiler Collection) をつかっている。これには、C, C++, Java などのコンパイラその他がふくまれている。’g++’ は UNIX 系の OS 上 (Linux、さらに MacOS もふくむ) で広くつかわれているフリーソフトである。詳しくは、‘gcc’ のホームページ `http://g++.gnu.org/` を参照のこと。

『端末』は Windows 上で Linux のような操作を再現するソフトである (`www.` 端末`.com`)。したがって、g++もつかえるようになっている。(自宅、下宿でのパソコンでつかうためには、端末インストール時に g++もインストール項目にえらんでおく必要がある。)

[1]ただし、Java は、各 OS 用の仮想機械を介することによって、おなじプログラムをべつの OS 上で走らせることができるようにしている。

それでは、これをつかって実習してみよう。端末 (Linux の類似品) での実習であるから、しばしば [1] も参照する。

［準備 1］**端末を起動する**

「アプリケーション」→「システムツール」→「端末」
とすると端末のウィンドウが出る。

［準備 2］**プログラム専用ディレクトリー (フォルダ) をつくる**

C の実習のときは、専用のディレクトリーをホームディレクトリーのなかにつくっておくと便利である。ここでは、すでに授業用のディレクトリとしてつくった、『Documents』ディレクトリーで代用する。

つぎに、この Cbox のなかに移る。

```
cd Cbox
```

いま、Cbox にいることは

```
pwd
```

というコマンド (print working directory) をつかうと、「/」からの径路 (絶対パス [2]) が出る。

```
/homer/se/14/rp00.../Documents
```

のような表示がでるはずである。

この Cbox のなかで、以下の、editor(編集ソフト) 『emacs』などによるソースプログラム作成、gcc によるコンパイル、さらに、結果の実行の作業をおこなえばよい。

［い］**ソース・プログラムをつくる**

『TeraPad 』で 「hello.c」という名のファイルをつくる。その内容は以下のとおりオリジナルな参考文献 [5] の初めにある例 [6] をほんのすこし変形したものである。

```
/**************    hello.c **********************
#include <stdio.h>

main()
{
      printf("Hello!!!\n");
}
```

この後、もちろん、保存 (save file) する必要がある。C のソースファイルの拡張子は「.c」である。

［ろ］**コンパイルする**

［い］でつくったファイルをコンピュータの理解できるように翻訳する。この操作をコンパイルという。コマンドは、プログラム名の後にファイル名を指定して、

```
g++ hello.c
```

となる。

［は］**実行形式のファイルの確認**

［ろ］の結果できたファイルを実行形式のファイルという。うまくでき上がっているか、調べてみよう。

```
ls
```

として、

```
a.out hello.c
```

となっていればよい。

```
ls -F
```

とすると

```
a.out* hello.c
```

というように、実行形式のファイルにはアスタリスクがつく。

試しにファイルの中味の見たい人は、emacs で見てみるとよい。とんでもないことになってるぞ!!!

［に］**プログラムの実行**

できあがったプログラムを走らせる（実行する）。

```
./a.out
```

と入力する。うまく走れば、ターミナル・ウィンドウ (端末端末) に

```
Hello!!!
```

と出力されるはず。

［ほ］**コンパイル・オプション**

初期の設定ではつねに a.out という名の実行形式のファイルができる。したがって、二つの異なるソースプログラムをコンパイルすると、はじめの実行形式のファイルは上書きされる。これを避けるために、とくに実行形式のファイル名を指定したいときは、たとえば

```
g++ hello.c -o hello
```

とする。

実行は

```
hello
```

うまく走れば、ターミナル・ウィンドウに

```
Hello!!!
```

と出力されるはず。

1.3 ソースコード再訪

一段落したところで、ソースプログラムを見ていくことにする [6]。

●主関数

C 言語のプログラムは、関数の集まりから成っている。その元締めが主関数 (main function) である。これは、プログラムのなかにひとつだけ存在する。プログラムは、主関数から始まり、そのなかでいろいろな関数を順次呼び出していく。`main()` の箇所がその始まりである。これは、数学の

$y = f(x)$ に似た書き方である。ただ、いまの場合、入力に当たる部分がないので空白になっている。これに関しては、関数の章で述べる。関数の中身 (定義) は、{ }の内側に記述される。

●標準出力

main 関数内に入ると、printf 関数を呼び出している。すなわち、

```
printf("Hello!!!\n");
```

である。この printf 関数は、二重引用符「" "」に囲まれた文字列を入力 ($y = f(x)$ の x に相当) として、それをそのまま端末 (端末端末) に出力 ($y = f(x)$ の y に相当) する (標準出力)。この関数は、標準ライブラリーのなかの入出力にかんする関数群を収めるライブラリー<stdio.h>に入っている。1 行めの#include <stdio.h>は、この関数と関連をつけるための記述である。

結局、ここでは、「Hello!!!という文字列を端末端末に印字する」というもっとも単純なことを、プログラムをつかっておこなっている。「\n」は何かと疑問の向きもあろう。「\n」を抜くとどうなるか、あるいは、「\n」を複数個入れるとどうなるか、を実験するとわかる。すなわち、それは、「改行 (Enter key)」をソースコードで記述するときの表記法である。

また、行末の「; (セミコロン)」は、日本語の「。(句点)」に相当する。つまり、ひとつの文章の終わりを意味する。これを忘れるのは文法上の誤りである。したがって、コンパイル時にしっかりエラーがお出ましになる。これは、案外、よくあるエラーであるので注意されたい。

1.4 g++の操作の意味

つぎに、ソースコードとコンパイラ (g++) との関係を、前節のプログラム「hello.c」を例にとって、みていこう。g++の意味をみていくということである。たとえていうと、以下のようになる。(この部分、名誉 TA 松本庄司さんの記述を部分的に拝借している。) 外国語のわからない日本人が外国人になんらかを頼むという状況をかんがえよう。その日本人はまず要件を紙に日本語でかき、つぎにだれかにそれを翻訳してもらう。そうしてできあがった外国語のメモを外国人わたす。g++のやってることことといまのたとえの対応はつぎのようになる。

```
hello.c   ……      日本語のメモ ( C Program source )

g++       ……      翻訳屋       ( C コンパイラ )

a.out     ……      外国語       ( 実行形式 )

CPU(コンピュータの頭)  外国人
```

コンパイル

g++

このコマンドの意味は g++(翻訳屋) に hello.c というメモを翻訳して a.out という外国語のメモになおすということ。この操作をふつうコンパイル (compile) とよんでいます。コンパイルの操作をしてくれるプログラムがコンパイラー (compiler) である。この『g++』では、『g++』そのものの名をあてている。(これが Linux のやりかたでしたね。「g++出でよ！」ということである。)

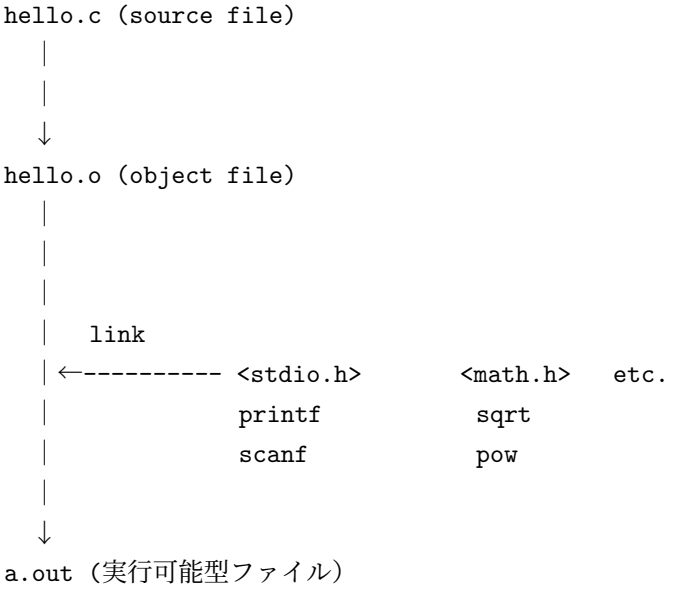
『g++』の操作は、さらに、標準入出力のライブラリ<stdio.h>ですでに定義されている関数 printf を呼んできて hello.o (object code、hello.c を機械語に変換したもの) にリンクさせている。

（図書館から必要な文献を借りてくるようなものである。）

実行 (RUN)

a.out というメモを直接 CPU に渡す。a と実行形式のファイル名そのものを入力するとプログラムが実行される。

以上を模式的にあらわせばつぎのようになる。



デバッグ（Debug)

プログラムがいつもすんなりと走るとはかぎらない。エラーの表示からおかしいとおぼしきところをさがそう。大雑把に分類して、エラーには 3 通りある [7]。すなわち、

（い）ソース・プログラム自体の文法的ミス (syntax error)

コンパイル時にエラー表示が、その箇所の行数とともに、出る。ただ、その行そのものに誤りがあるとはかぎらない。その付近があやしいということを意味する。

エラー表示例

(※　Vine Linux の g++での例)

```
[matsu@localhost Documents]$ g++ hello.c
hello.c: In function ‘ int main() ’:
hello.c:5: error: expected ‘ ; ’ before ‘ return ’
```

この例では、ソースコードの 5 行目あたりにエラーがあるということである。いったいどこにあるのか。急げ、探偵君！

このエラーは、表示からもわかるように、前述した「; (セミコロン)」をつけるのをわすれたときの例である。

（※　表示例は、RAINBOW の gcc のものとはちがうかもしれないが、御容赦ねがいたい。）

(ろ) プログラムの要求するもの自体がおかしいとき。メモリーが確保できなかったときなど、である。プロセッサがプログラムのあるステップに意味をあたえることができない、という状態 (意味上の誤り、semantic error) である。この場合、実行時にエラー表示がでることもある。しかし、エラー表示が出ないときもあるので要注意である。
（※ コンパイル時にエラーの出ることも、原理的にはあり得る。）
エラー表示例
(※　Vine Linux の gcc での例)

```
[matsu@localhost Documents]$ a
正の整数をあたえてください。
13
浮動小数点演算例外です
```

これは、割り算で除数が 0 となるような場合をふくむときのエラーの表示である。
さらに、

(は) 解き方が数学的におかしいことや、目的のものとはちがうものを求めるプログラムをつくってしまうこともある。たとえは、球の体積をもとめるところを、まちがって表面積を計算してしまうような場合である。巷で問題になっている誤りもほとんどこの型 (論理的な誤り、logical error) である。この場合は、もちろん、エラー表示は出ない。つねに検算して確認するように努めるのがいちばんであろう。やはり、最後は人間ですな。

ちゃんと走るまで忍耐ですぞ !!!
まぁ、気長にやってください。

1.5　他の OS 上での C プログラミング

1.5.1　序論

Linux の場合、ふつうにインストールすると、g++も自動的にインストールされているので、自然にプログラミングができる環境にある。しかし、自宅や下宿でもプログラミングをしたいという向きもあろう。このとき、ひとつの方法は、もちろん、Linux じたいを仕入れることである。いまひとつは、、Windows や Mac の上で動く C コンパイラをつかうことである。もちろん製品 (商品) もあるが、重宝なフリーソフトがいくつか出ている (シェアウェアもある) ので、これを利用すればよい。これらの C コンパイラをもちいれば、この授業であつかっているような基本的な内容 (標準 C プログラミング) にかんしては、Linux の場合とほぼおなじようにプログラミングをおこなうことができる。

1.5.2　具体的方法

Windows 編オンライン・プリント『フリーソフトのダウンロード、つかいかた 』も参照のこと。窓の杜 (http://www.forest.impress.co.jp/) や Vector(http://www.vector.co.jp/) などから、適当

なものを見つければよい。

(i)Windows の場合

○『端末』をダウンロードするときに、『g++』も選択して入手する。

『端末』の Web Page`www. 端末.com` に行けばよい。

○窓の杜から探す

・Borland C++Compiler

直接の URL は、Borland 社のページ

(http://www.borland.com/jp/products/cbuilder/freecompiler.html)。

○ Vector から探す

・LSI C-86

LSI-C のページ (http://www.vector.co.jp/vpack/filearea/win/prog/c/lsic/index.html) の DOS 用 LSI-C フォルダのなかにある。

ソースコードは、適当な編集ソフトをつかってつくる。実行結果は、『コマンドプロンプト』のウィンドウに表示される。

これらを GUI 環境でつかうためには、以下のソフトのいずれかを併用すればよい (ほかにおなじような機能をもつソフトも載っている)。

C 言語のページ (http://www.vector.co.jp/vpack/filearea/win/prog/c/index.html)

にある

・CPad for Borland C++Compiler

あるいは、上記の LSI-C のページにある

・C 言語を始めよう！Ver.1.1.1.7

などがある。

GUI 環境では、ソースコードの編集もこれらのソフトでできる。また、コンパイル、実行もメニューでおこなうことができる。

(ii)Mac の場合

・g++

MacOS は、UNIX を基盤にしているので、g++がつかえる。OS に添付されている X-code の DVD をインストールすれば、自動的に g++もインストールされる。(あるいはすでにインストールされているかもしれない。)

つかいかたは、端末の場合とほとんどまったくおなじである。すなわち、Mac の「ターミナル」ウィンドウが、われわれの Linux での「端末端末」に相当する。ここで、「g++」でのコンパイル、「a」、「hello」等の実行の操作をすればよい。

また、ここでやっているように、C プログラム用に「Cbox」などのフォルダをつくり、そのなかで作業すると便利である。

ソースコード作製には、適当な編集ソフトをつかえばよい.。X-code の DVD をインストールしたときに「emacs」もインストールされているから、これを使用してもよい。ただし、この場合、GUI が効かないこともあるかもしれない。そのときは、もうひとつ「ターミナル」ウィンドウを出せば解決する。この点、御注意を!!!

(※ この項目にかんして、名誉 TA 奥村さん、高尾さん、および Mac ユーザーの（以前の）受講生の教えを受けた。)

1.5.3　漢字コードの問題

これは、プログラミングのソースファイルにかぎったことではないのだが、Linux 上でつくったファイルとそれ以外の OS 上でつくったファイルとは、OS が日本語を認識するしかた (§ **??** 漢字コード) の異なることがある。これを変換する方法、すなわち、Mac, Windows $\longleftrightarrow$ Linux の漢字コードの変換方法については、[8] に記してある。だが、保存するときに、どの OS でも漢字コードを「utf-8」にしておくと漢字変換の必要もなく文字化けを気にすることもないので便利であろう。

参考文献

[1] 授業資料『端末と Linux の手引き (*An InformalIntroduction to* 端末 *and Linux*』)

[2] [1] の § 3.5。

[3] [1] の § 4.1。

[4] [1] の § 4.2。

[5] B. W. カーニハン & D. M. リッチー (石田 晴久 訳) 『プログラミング言語 C・第 2 版 (The C programming Language)』(共立出版、1989 年)。

[6] [5] の § 1.1。

[7] L. ゴールドシュレーガー & A. リスター (武市・小川・角田　訳)『計算機科学入門・第 2 版 (Computer Science)』、(近代科学社、2000 年) の §2-2。

[8] [1] の § 5.5。

[9] 坂村 健『痛快 コンピュータ学』(集英社文庫、2002 年)。

[10] [5] の § 2.1-§2.4。

[11] J. May & J. Whittle (武舎 広幸 訳) 『Symantec C++ for Macintosh トレーニングブック (Symantec C++ for The Macintosh)』(翔泳社、1994 年) 第 1 章。

[12] D. マーク『Windows ではじめる C プログラミング』(星雲社、2001 年)。

[13] 授業資料『C 言語によるプログラミング 3 (*A Pedestrian Approach to the C Programming Language*)』。

[14] [1] の § 5.3。

[15] [5] の § 1.2。

[16] 『*Abrégé d'histoire des mathématiques*』ed L. Dieudonné (Hermann, Parris, 1978). [17] に引用されている。

[17] R. Rammal, G. Toulouse and M. A. Virasoro, Rev. Mod. Phys. **58**, 765-88 (1986).

[18] [5] の pp 45-6、および pp 234-5。

[19] S. ハルトマン & J. ミクシンスキー（州之内治男・前田功雄 訳）『ルベーグ積分入門』（サイエンス社、1984 年）巻末付録: 訳者補足 2。

[20] 坂村 健『大人のための「情報」教科書』（数研出版、2003 年）。

[21] 和田 秀男『計算数学』（朝倉出版、2000 年）、第 1 章。

[22] 都倉 信樹『新版 情報工学』（放送大学教育振興会、1999 年）、第 3 章。

[23] 西村 めぐみ『図解でわかる Linux のすべて』(日本実業出版社、2000 年)、§ 6.1。