学術情報センター

Linux 利用の手引き
(C 言語編)
コンパイル・実行・デバック

2010 年 7 月

学術情報センター

目次

１　概要

学術情報センターの実習室に導入されている UNIX(Linux)の利用方法を C 言語のプログラミングを用いて説明します。とくにセンターで特別に変更している点を中心に説明しますので、UNIX（Linux)や C 言語の詳細は市販の本を参考にして下さい。なお、Linux 手引き（基本編）にはログイン・コマンド・エディタ・ファイル等が記載されていますので該当する部分を参考にして下さい。

２　C 言語

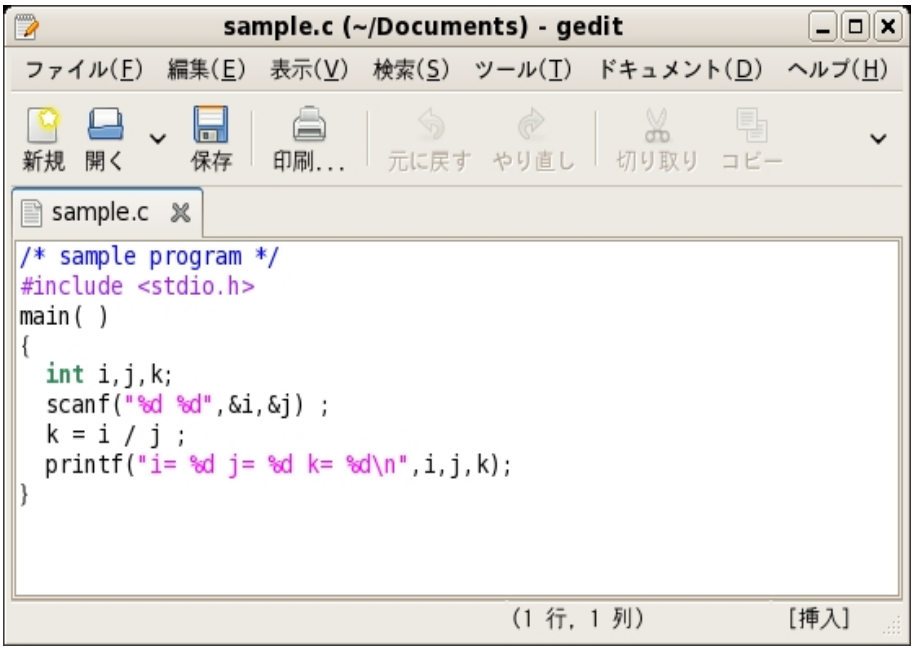
電子情報工学科や情報工学科のプログラミング学習に利用されており、C 言語は社会でも広く利用され、技術系では必ず習得しておかなければならない言語の１つです。とくに UNIX（Linux）システム開発環境では C 言語を中心により開発されているので、便利な道具が準備されています。また、オープンソースのシステム開発では C 言語を利用した開発が定着しています。

３　言語処理

Linux の C 言語は GCC と呼ぶ、オープンソースの世界で定評のあるコンパイラが利用できます。エディタも VI や Emacs の従来からのエディタに加え GEDIT とよぶ Windows システムのメモ帳ライクなエディタも準備されています。統合化された開発システムはありませんがシンプルな理解し易いコマンドがあります。 ここで C 言語のプログラムを処理する流れを説明し、機能や目的を解説します。

３．１エディタ（プログラム作成）

C 言語のプログラムをエディタを利用して入力します。



エディタは Vi または Emacs でも同様です。

補足　gedit はアプリケーションメニューー＞システムツールから実行できます。

３．２翻訳（コンパイル）

エディタを利用して入力されたプログラムはソースコードとも呼ばれて、人間にとって理解し易い形式で記述されています。このソースコードを計算機（機械）に処理できる形式（機械語）に変換することを翻訳（コンパイル）と呼びます。

例　sample.c をコンパイルする場合

```
a00% cc  sample.c
```

３．３実行

エラーがなかった場合はコンパイル結果として機械語のプログラムが a.out として出力されます。出力された a.out を実行するにはファイル名 a.out をそのままコマンドと同様に入力します。

```
a01% a.out
1 2
i= 1 j= 2 k= 0
a01%
```

なお、実行時にエラーが発生することもあります。これは 0 で割り算をして、計算できない場合や途中で異常終了する場合などです。このエラーを実行時のエラーと呼びます。

３．４エラーの場合

C 言語プログラムに何らかのエラーがある場合はコンパイル時にエラーメッセージを表示して機械語の実行ファイル(a.out)を出力しない時があります。この場合にはエラーメッセージを読んで該当する C 言語プログラムの部分を修正します。修正にはエディタを利用し、その後、翻訳し直します。

例　** の計算式が無効である場合

```
a01% cc sample.c
sample.c: 関数 `main' 内:
sample.c:7: `unary *' の引数として無効な型
a01% cat -n sample.c
     1  /*   sample program */
     2  #include <stdio.h>
     3  main( )
     4  {
     5        int i,j,k ;
     6        scanf("%d %d",&i,&j) ;
     7        k = i ** j ;
     8        printf("i= %d j= %d k= %d¥n",i,j,k) ;
     9    }
a01%
```

7 行目の計算式がおかしい。** は C 言語の文法上で許されない。

実行時のエラー例

```
a01% cc sample.c
a01% a.out
1 0
Floating exception
a01% cat -n sample.c
     1  /*   sample program */
     2  #include <stdio.h>
     3  main( )
     4  {
     5        int i,j,k ;
     6        scanf("%d %d",&i,&j) ;
     7        k = i / j ;
     8        printf("i= %d j= %d k= %d¥n",i,j,k) ;
     9    }
```

入力データが適当でなく'0'で割算を実行しようとした。

４　リダイレクションによる実行

UNIX ではシェルの機能を利用して入出力の切替えが容易にできます。入力するデータをファイルから読んだり、画面表示する出力をファイルに書き出すことが簡単にできる特徴を持っています。

例 input.d から入力する場合。（あらかじめ、エディタで入力データを作成してあること）

```
a01% a.out < input.d
i=  1 j=  2 k=  0
a01%
```

例 output.d に出力する場合。

```
a01% a.out > output.d
1,2
a01% cat output.d
i=  1 j=  2 k=  0
a01%
```

例 両方する場合。

```
a01% a.out < input.d > output.d
a01%
```

５　ライブラリの連結

C 言語から数学ライブラリ（sin,cos,sqr,...）や X window ライブラリ(図形作成）を連結するにはつぎのように行ないます。

数学ライブラリ

```
a01% cc sample.c -lm
```

Plotter ライブラリ

```
a01% cc sample.c -lpu
```

X-Window ライブラリ

```
a01% gcc -I/usr/local/X11R6/include -L/usr/local/X11R6/lib sample.c -lX11
```

５．１ライブラリーリンクエラー

連結する関数名を間違えるとライブラリエラーが発生し、実行形式が作成されない。また、リンクオプションを忘れるても同様のエラーが発生します。

例　sin 関数で　-lm オプションを忘れた場合

```
a01% cc sample.c
/tmp/ccWisvjh.o(.text+0xbd): In function `main':
: undefined reference to `sin'
collect2: ld はステータス 1 で終了しました
```

６．レポート作成時の注意

　課題提出する場合、C プログラムと実行結果をまとめてファイルを作り、それに考察や感想を追加するときには以下のように行います。

実行結果のファイル(kekka)作成

```
a01% a.out >  kekka
a01%
```

プログラム(program1.c)と実行結果(kekka)を１つのファイル(report09)を作成する。

```
a01% cat program1.c  kekka  >   report09
a01%
```

作成した ファイル(report09)に Emacs エディタなどを利用して考察や感想を追加して、レポートを作成して下さい。

印刷

```
a01% flp  report09
a01%
```

７．補足

Linux を利用する場合に操作方法だけでなく、関連する情報やシステム情報などを解説します。

７．１ マニュアルの表示

man コマンド

man 調べたいコマンド名を指定する。

例　date コマンド

```
a01% man date
date(1)                                                    date(1)
```

名前

date - システムの日付と時刻を表示・設定する

書式

```
date [-uR] [-d datestr] [-f datefile] [-r file] [-s datestr] [-I [time-
spec]]  [--date=datestr]  [--file=datefile]  [--iso-8601[=timespec]]
[--reference=file]  [--set=datestr]  [--rfc-822]  [--universal] [--utc]
[+format] [MMDDhhmm[[CC]YY][.ss]]

date [--help] [--version]
```

説明

date は引数を指定しないと、現在の時刻と日付を表示する（表示形式 は　‘%a %b %e %H:%M:%S %Z %Y’ となる。以下を参照のこと）。

：　：　：　：　：

また、コマンド名以外にも、システムコールやライブラリ、設定ファイルなどのマニュアルもあります。マニュアルはセクション毎に分けられているので、番号を指定して呼び出します。

1 コマンド（ls、cat、kterm など）
2 システムコール（C 言語の関数。open、fork など）
3 ライブラリ関数（C 言語の関数。printf、fopen、Tck/Tk、Xlib などの関数群）
4 デバイス・デバイスドライバ
5 ファイルフォーマット（uuencode などのファイル形式、hosts などの書式）
6 ゲーム
7 その他（環境変数の説明、groff の書式など）

例　C 言語の関数 fread を調べる

```
a01% man 3 fread

FREAD(3)                    Linux Programmer's Manual                    FREAD(3)
名前
       fread, fwrite - バイナリストリームの入出力
書式
       #include <stdio.h>
       size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);

       size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
説明       fread()  関数は stream ポインタで指定されたストリームから nmemb 個のデータを読
み込み、ptr で与えられた場所に格納する。個々のデータは size バイトの長さを持つ。
       fwrite() 関数は ptr で指定された場所から得た nmemb 個のデータを、 stream ポインタ
で 指定されたストリームに書き込む。個々のデータは size バイトの長さを持つ。
       これらの処理をロックせずに行いたいときは、 unlocked_stdio(3) を参照のこと。
              :              :              :              :
```

７．２ ファイルタイプの判定

file コマンドにより、ファイルタイプを表示できる。

```
a01% file a.out
a.out:     ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.6.9,
dynamically linked (uses shared libs), for GNU/Linux 2.6.9, not stripped
```

インテル 80386 の Linux2.6.9 用の実行できるプログラムと表示される。

７．３　機械語プログラムの逆アセンブル

objdump コマンドによりバイナリデータ（２進形式）を逆アセンブルして表示する。

```
a01% objdump -d a.out | more

a.out:     file format elf32-i386

Disassembly of section .init:

08048358 <_init>:
 8048358:       55                      push   %ebp
 8048359:       89 e5                   mov    %esp,%ebp
 804835b:       83 ec 08                sub    $0x8,%esp
 804835e:       e8 b1 00 00 00          call   8048414 <call_gmon_start>
 8048363:       e8 38 01 00 00          call   80484a0 <frame_dummy>
 8048368:       e8 23 07 00 00          call   8048a90 <__do_global_ctors_aux>
```

```
 804836d:       c9                      leave
 804836e:       c3                      ret
Disassembly of section .plt:

08048370 <sprintf@plt-0x10>:
 8048370:       ff 35 3c 9d 04 08       pushl  0x8049d3c
 8048376:       ff 25 40 9d 04 08       jmp    *0x8049d40
 804837c:       00 00                   add    %al,(%eax)
       ...

08048380 <sprintf@plt>:
 8048380:       ff 25 44 9d 04 08       jmp    *0x8049d44
```

７．４ 計算機の内部表現(ソースコードと実行形式)

コンピュータ上のファイルはテキスト形式とバイナリー（２進数）形式に分けられます。テキスト形式はC言語等のプログラムのソースコードなどで利用されており、文字情報で構成されています。文字コードはアスキーコード表で表され１６進では次のように表現します。

例　sample.c を１６進表示するには

```
a01% od -xc  sample.c
0000000 2a2f 7320 6d61 6c70 2065 7270 676f 6172
          /   *       s   a   m   p   l   e       p   r   o   g   r   a
0000020 206d 2020 6165 3939 3030 2031 e6c3 f4c9
          m               e   a   9   9   0   0   1     303 346 311 364
0000040 c0c2 bacf 2f2a 230a 6e69 6c63 6475 2065
         302 300 317 272   *   /  ¥n   #   i   n   c   l   u   d   e
0000060 733c 6474 6f69 682e 0a3e 616d 6e69 2028
          <   s   t   d   i   o   .   h   >  ¥n   m   a   i   n   (
0000100 0a29 0a7b 2020 6e69 2074 2c69 2c6a 3b6b
          )  ¥n   {  ¥n           i   n   t       i   ,   j   ,   k   ;
0000120 200a 7320 6163 666e 2228 6425 2520 2264
         ¥n           s   c   a   n   f   (   ″   %   d       %   d   ″
0000140 262c 2c69 6a26 2029 0a3b 2020 206b 203d
          ,   &   i   ,   &   j   )       ;  ¥n           k       =
0000160 2069 202f 206a 203b 200a 7020 6972 746e
          i       /       j       ;      ¥n           p   r   i   n   t
0000200 2866 6922 203d 6425 6a20 203d 6425 6b20
          f   (   ″   i   =       %   d       j   =       %   d       k
0000220 203d 6425 6e5c 2c22 2c69 2c6a 296b 0a3b
          =       %   d   ¥   n   ″   ,   i   ,   j   ,   k   )   ;  ¥n
0000240 0a7d
          }  ¥n
0000242
```

次に実行形式はコンピュータがそのまま実行できる形式で保存されるため、人間には理解しづらい形式になっています。OS のための制御領域と機械語（アセンブラ）・データ領域で構成されています。

```
a01% od -xc a.out | more
0000000 457f 464c 0101 0001 0000 0000 0000 0000
        177   E   L   F 001 001 001  ¥0  ¥0  ¥0  ¥0  ¥0  ¥0  ¥0  ¥0  ¥0
0000020 0002 0003 0001 0000 82f4 0804 0034 0000
        002  ¥0 003  ¥0 001  ¥0  ¥0  ¥0 364 202 004  ¥b   4  ¥0  ¥0  ¥0
0000040 07f0 0000 0000 0000 0034 0020 0007 0028
        360  ¥a  ¥0  ¥0  ¥0  ¥0  ¥0  ¥0   4  ¥0      ¥0  ¥a  ¥0   (  ¥0
0000060 001c 0019 0006 0000 0034 0000 8034 0804
        034  ¥0 031  ¥0 006  ¥0  ¥0  ¥0   4  ¥0  ¥0  ¥0   4 200 004  ¥b
0000100 8034 0804 00e0 0000 00e0 0000 0005 0000
          4 200 004  ¥b 340  ¥0  ¥0  ¥0 340  ¥0  ¥0  ¥0 005  ¥0  ¥0  ¥0
0000120 0004 0000 0003 0000 0114 0000 8114 0804
        004  ¥0  ¥0  ¥0 003  ¥0  ¥0  ¥0 024 001  ¥0  ¥0 024 201 004  ¥b
0000140 8114 0804 0013 0000 0013 0000 0004 0000
        024 201 004  ¥b 023  ¥0  ¥0  ¥0 023  ¥0  ¥0  ¥0 004  ¥0  ¥0  ¥0
0000160 0001 0000 0001 0000 0000 0000 8000 0804
        001  ¥0  ¥0  ¥0 001  ¥0  ¥0  ¥0  ¥0  ¥0  ¥0  ¥0  ¥0 200 004  ¥b
0000200 8000 0804 0504 0000 0504 0000 0005 0000
         ¥0 200 004  ¥b 004 005  ¥0  ¥0 004 005  ¥0  ¥0 005  ¥0  ¥0  ¥0
0000220 1000 0000 0001 0000 0504 0000 9504 0804
         ¥0 020  ¥0  ¥0 001  ¥0  ¥0  ¥0 004 005  ¥0  ¥0 004 225 004  ¥b
           :                                                   :
```

後、長いので省略

アスキーコード表

	00	10	20	30	40	50	60	70
00	NL	DE	SP	0	@	P	‘	p
01	SH	D1	!	1	A	Q	a	q
02	SX	D2	”	2	B	R	b	r
03	EX	D3	#	3	C	S	c	s
04	ET	D4	$	4	D	T	d	t
05	EQ	NK	%	5	E	U	e	u
06	AK	SN	&	6	F	V	f	v
07	BL	EB	’	7	G	W	g	w
08	BS	CN	(	8	H	X	h	x
09	HT	EM	)	9	I	Y	i	y
0A	LF	SB	*	:	J	Z	j	z
0B	HM	EC	+	;	K	[	k	{
0C	CL	→	,	<	L	¥	l	|
0D	CR	←	－	=	M	]	m	}
0E	SO	↑	.	>	N	^	n	￣
0F	SI	↓	／	?	O	_	o	DL