UNIX 上の C 言語プログラム入門 テキスト

於 東京経営短期大学 平成 17 年度 IT 活用セミナー

平成 17 年 8 月 23 日 (火)

担当 神保 雅人

内　　容

はじめに

東京経営短期大学では，平成 17 年度より経営総合学科が開設されましたが，それに伴い，教育用情報システムの更新が行われました。この新システムでは，端末は最新 iMac で，Mac OS X は Mac G5 サーバからのネットブートとし，この端末から Windows サーバや Linux サーバに OS 起動メニューのアイコンクリックのみで接続できる方式を開発しました。

ここに至るには，先ず，平成 4 年度に開学した際に導入された富士通製の汎用機を平成 8 年度のシステム更新で日立製（HP の OEM）のワークステーションに置き換え，端末 20 台のワークステーション室を設けました。更に，その後のシステム更新ではダウンサイジングを推し進め，デスクトップ Linux をデュアルブートで 20 台の PC に導入したり，PC による Linux サーバを導入したり，というように様々な試みをしてきました。今回のシステムでは，3 室あるコンピュータ演習室のどこからでも Linux サーバにアクセス可能にしました。

また，経営総合学科のカリキュラムにはネットワーク時代に相応しい科目も多数取り入れられ，Linux サーバを利用する頻度が増えました。これに合わせて，従来は MS-DOS や Windows 上で行ってきた経営情報学科のプログラミング教育も，平成 17 年度の 2 年次前期配当科目のプログラミング言語演習は Linux 上で vi エディタを用い，gcc で翻訳編集を行うという，オープンソースのみの構成で行いました。

この講座は，担当者の上述のような経験を生かして，日頃は Windows に慣れ親しんでいらっしゃる方々に，オープンソースによる C 言語プログラミング教育の一端を経験していただこうという趣旨で設けられました。

1．ログインとパスワードの変更

iMac の OS 起動メニューで Linux のアイコンをクリックして，暫くすると SUSE Linux のログインダイアログが表示されます。ここで，セッションタイプが FVWM となっていることを確認した上で，配布されたメモ用紙に書かれたユーザ名とパスワードとを入力してください。このパスワードは暫定的なものですから変更が必要です。

無事にログインが完了すると，ターミナルエミュレータ（端末の機能を真似するソフトウェアの意）の窓が開いた状態となっています。ここでコマンドを入力します。

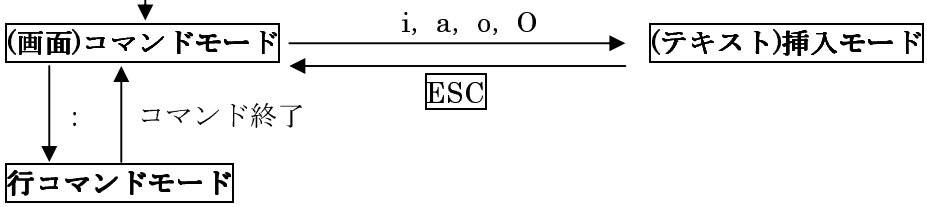
パスワードを変更するコマンドは passwd です。このコマンドを打ち込んで Enter キーを打つと，現時点でのパスワードの入力，新規に自分で設定したいパスワードの入力，新規のパスワードを覚えているかどうかの確認のための再入力をそれぞれ促すガイドコメントが出てきます。これらに従って，入力を行ってください。

なお，短すぎるパスワード，辞書にある単語を用いたパスワード，現時点のものと殆ど変わらないパスワードの場合，警告が出て，変更が出来ません。

ここで，一旦ログオフして，新規パスワードで再度ログインしましょう。

2. vi の基本的な使い方

・起動　　ｖｉ　ファイル名



・挿入モードへの切り替え

ｉ　カーソルの前に文字挿入　　ａ　カーソルの後ろに文字挿入

ｏ　カーソルの位置に後ろの行を挿入　　Ｏ　カーソルの位置に前の行を挿入

・カーソル移動コマンド

ｋ　上の行に移動　　ｊ　下の行に移動

ｌ　右の文字に移動　　ｈ　左の文字に移動

・削除コマンド（基本）

ｘ　カーソルの後ろの文字を削除　　Ｘ　カーソルの前の文字を削除

・保存終了コマンド

：ｗｑ　　（行コマンドモードに入り，書き出して抜け出す）

・行の連結コマンド

Ｊ　カーソルのある行と次の行とをつなげる

・取り消しコマンド

ｕ　直前の操作を取り消す

・保存しない終了

：ｑ！　　（行コマンドモードに入り，保存せずに抜け出す）

・カット，コピー，ペースト

ｄｄ　カーソルのある行を削除してカットバッファに移す

ｙｙ　カーソルのある行の文字列をカットバッファにコピーする

ｐ　カットバッファの内容をカーソルから後ろにペーストする

Ｐ　カットバッファの内容をカーソルより前にペーストする

（ｘやＸで削除した文字もカットバッファに移る）

・すばやい移動

ｊ，ｋ，ｌ，ｈの前に繰り返しの回数を指定する　（例　５ｌ）

移動したい列の番号に続けて縦線を用いる　（例　５｜）

０　行の先頭に移動

＄　行末に移動

ｆ[文字]　カーソルのある行での後方検索，検索後カーソルは文字の上　（例　ｆ０）

Ｆ[文字]　カーソルのある行での前方検索，検索後カーソルは文字の上　（例　Ｆｇ）

ｔ[文字]　カーソルのある行での後方検索，検索後カーソルは文字の左　（例　ｔ０）

Ｔ[文字]　カーソルのある行での前方検索，検索後カーソルは文字の右　（例　Ｔｇ）

Ｇ　最終行へ移動

[行番号]Ｇ　行番号で指定した行に移動　（例　１Ｇ）

・行番号の表示

：ｓｅｔ　ｎｕｍｂｅｒ

・行番号の非表示

：ｓｅｔ　ｎｏｎｕｍｂｅｒ

・マークと移動

ｍ[文字]　マーク（印）したい場所にカーソルを合わせ，ｍに続けてａ～ｚまでの１文字を打つ（例　ｍａ）

移動はバッククォート（‘）に続けて，マーク時に指定した文字を打つ　（例　‘ａ）

シングルクォート（’）に続けて，マーク時に指定した文字を打つとマーク行の先頭に移動

・削除コマンド（応用）

ｄＧ　カーソルの位置からファイルの終わりまでの文字を削除

ｄ１Ｇ　カーソルの位置からファイルの先頭までの文字を削除

ｄ＄　カーソルの位置から行末までの文字を削除

ｄ０　カーソルの位置より前の文字を行の先頭まで削除

ｄｆ[文字]　カーソルの位置から検索した文字までを削除

ｄｗ　カーソルの位置から１単語分の文字を削除

・置換コマンド（コマンドモードで利用するもの）

ｒ　カーソルの位置の１文字を置き換える（ｒを打った後に置き換えたい文字を打つ）

Ｒ　カーソルの位置から打ち込んだ分の文字を置き換える（置き換え終了は[ESC]キー）

ｃｗ　カーソルの位置から１単語分の文字を置き換える（置き換え終了は[ESC]キー）

・グローバルな置換（ｅｘエディタコマンドの利用）

：１，＄ｓ/[元の文字列]/[新しい文字列]/ｇ

［解説］‘：’を打つと切り替わる「行コマンドモード」とは，実際には「ｅｘエディタ」というラインエディタのモードで，スクリーンエディタとしての「ｖｉエディタ」の基本的な部分を担っている。

１，＄は先頭行から最終行，ｓはsubstitute（置き換え），ｇはglobal（全体）を意味する。（もしも，最後の/ｇを付けなければ，それぞれの行で最初に見つかった元の文字列のみが，新しい文字列へ置き換えられる処理が実施される。）

3. C言語の簡単なプログラム

最初に，mkdir cworkと入力して，作業用ディレクトリcworkを作成します。次に，cd cworkと入力してカレントディレクトリをcworkに移します。そこで，下に示したC言語プログラムをviで作成しましょう。ファイル名はsample.cとします。プログラムが出来上がったら，保存してviを終了させます。

```
#include <stdio.h>

#define NINZU 5

void hyouji(char [][],int []);

int main(void)
{
    int i,ten[NINZU];
    char name[NINZU][20];

    for (i=0;i<NINZU;i++)
    {
        printf("%d番目の氏名を入力してください：",i+1);
        scanf("%s",name[i]);
        printf("%sさんの点数を入力してください：",name[i]);
        scanf("%d",&ten[i]);
    }

    hyouji(name,ten);

    return 0;
}

void hyouji(char str[][20],int p[])
{
    int i;

    printf("\n番号： 氏名                 点数\n");

    for (i=0;i<NINZU;i++)
        printf("%4d： %-20s%4d\n",i+1,str[i],p[i]);
}
```

プログラムの内容は，5人分の氏名と点数とをキーボードからの入力で受け取り，一覧を画面に表示するものです。プログラム本体のmain関数では，繰り返し処理の

for 文を制御する整数型の変数 i，人数分の点数を格納する整数型の配列 ten，人数分の氏名を格納する文字型の配列 name が宣言されて用いられています。

また，main 関数でそれぞれの配列の各要素にデータが格納された後に，画面への表示処理をひとまとめにしたユーザ定義関数 hyouji を呼び出し，それらのデータを渡して，処理を行っています。

4. gcc による翻訳と実行

作成した sample.c は人間が理解できる高級言語に従って記述されたプログラムです。コンピュータがそこに書かれたことを実行するには，コンピュータに直接指令を行える機械語に翻訳してやらなければなりません。この翻訳の作業は，コンパイラと呼ばれるソフトウェアで行います。ここでは，オープンソースのコンパイラ gcc を利用しましょう。

gcc の基本的な使い方は，gcc ソースファイル名 です。但し，この書き方では，翻訳された実行形式のプログラムファイル名はいつも a.out となってしまい，色々なプログラムを作成していくには，不便です。また，プログラムに誤り（エラー）があっても何も知らせてくれません。そこで実用的な使い方としては，いくつかのコンパイルオプションを組み合わせて，次のように指定します。

```
gcc -Wall -O3 -o sample sample.c
```

ハイフンに続くのがコンパイルオプションで，-Wall は Warning all，-O3 は Optimize level 3，-o は output file を指しています。最後にソースファイル名が来るので注意が必要です。また，この場合の実行形式のプログラムファイル名は単に sample で，Windows の様な.exe といった拡張子は付けないのが UNIX 系の OS の習慣です。

5. 基本情報技術者試験の問題

次に挙げる，平成 17 年度春期基本情報技術者の午後の問題について，その解答を埋めるとともに，この問題で作成された関数を呼び出す main 関数の例を付け加えたプログラム jouhou.c を，現在作業中のディレクトリから見て 2 つ上のディレクトリ（../../で表す）に置いてあります。

cp ../../jouhou.c . によってカレントディレクトリ（.で表す）にコピーして，gcc で翻訳して実行してみましょう。

平成 17 年度 春期 基本情報技術者 午後 問題

問 6　次の C プログラムの説明及びプログラムを読んで，設問に答えよ。

〔プログラムの説明〕

関数 print_string は，欧文ピッチ処理（文字固有の字幅に従って字送りする）を行って印刷するとき，単語の途中で改行されないように英文を出力するプログラムである。

(1) 関数 print_string の引数は，次のとおりである。

line_w	1 行の行幅（ポイント数）
str_list	出力する英文を構成する単語の配列 （最後の要素には，NULL が格納されている）
char_list	出力する単語を構成する文字とその文字幅（ポイント数）のリスト（構造体 CHARPROF の配列）
space_w	空白文字の文字幅（ポイント数）

(2) 文字幅は，文字ごとに次に示す構造体 CHARPROF で定義される。

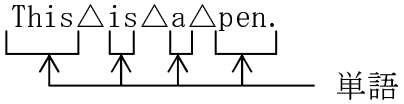
```
typedef struct { char char_p; /* 文字 */
                 int  char_w; /* 文字幅（ポイント数） */
               } CHARPROF;
```

(3) 単語幅は，単語を構成する各文字の文字幅の和である。単語幅を求めるために，関数 word_width を用いる。

(4) 単語を出力しようとしたときに，1 行の行幅を超える場合は，その単語が次の行の先頭になるように出力する。ただし，どの単語幅も行幅を超えることはない。

(5) 単語は，空白を含まない文字列である。単語と単語の間には，1 文字の空白文字を出力する。ただし，行の最後に出力する単語の後には，空白文字は出力しない。

(例)



注　△は空白を示す。

〔プログラム〕

```
#include <stdio.h>

typedef struct { char char_p; /* 文字 */
                 int  char_w; /* 文字幅（ポイント数） */
               } CHARPROF;

void print_string(int, char *[], CHARPROF *, int);
int word_width(char *, CHARPROF *);

void print_string(int line_w, char *str_list[],
                  CHARPROF *char_list, int space_w) {
    int cur_w = 0, str_w, idx;

    for (idx = 0; [ a ]; idx++) {
        str_w = word_width(str_list[idx], char_list);
```

```
        [ b ];
        if (cur_w == str_w)     /* 最初の単語？ */
            printf("%s", str_list[idx]);
        else {
            cur_w += space_w;
            if (cur_w <= line_w)
                printf(" %s", str_list[idx]);
            else {
                [ c ];
                printf("\n%s", str_list[idx]);
            }
        }
    }
    putchar('\n');
}

int word_width(char *str, CHARPROF *char_list) {
    int print_w = 0, idx;

    while (*str != '\0') {
        for (idx = 0; [ d ]; idx++);
        print_w += char_list[idx].char_w;
        str++;
    }
    return print_w;
}
```

設問　プログラム中の［ d ］に入れる正しい答えを，解答群の中から選べ。

a に関する解答群

ア idx <= line_w　　イ idx < line_w

ウ idx + 1 < line_w　　エ str_list[idx] != NULL

オ str_list[idx] = NULL　　カ str_list[idx] == NULL

b，c に関する解答群

ア cur_w += space_w　　イ cur_w += str_w

ウ cur_w = 0　　エ cur_w = space_w

オ cur_w = space_w + str_w　　カ cur_w = str_w

d に関する解答群

ア *str != *char_list[idx].char_p

イ *str != char_list[idx].char_p

ウ *str == *char_list[idx].char_p

エ *str == char_list[idx].char_p

オ str != char_list[idx].char_p

カ str == char_list[idx].char_p