

SISTEMI OPERATIVI

Esercitazione 2

1 Editor di testi

Sul sistema sono disponibili buoni editor come *kate* (preferibile) e *gedit* che sono adeguati per le esigenze del corso, a parte l'incapacità di *gedit* nella versione disponibile in laboratorio (a causa della obsoleta distribuzione OpenSuse) di scrivere nel filesystem `/mnt/Z`.

Tuttavia, se si intende esplorare il famoso editor Emacs/XEmacs, lo si può avviare dallo shell con il comando *emacs* & (oppure *xemacs* se sul sistema utilizzato è installata la versione XEmacs) M

Per cominciare a scrivere, dal menu Files selezionare Open File... e, nel prompt che compare nella parte bassa della finestra, scrivere il nome del file che si vuole creare (ad esempio prova.txt). In modo più rapido, anziché usare la barra dei menu, si può usare la combinazione di tasti: `Ctrl^x Ctrl^f`

A questo punto scrivere qualcosa nel file appena creato, e quindi salvarlo. Le alternative per quest'ultima operazione sono:

- dal menu Files selezionare Save Buffer
- usare la combinazione di tasti: `Ctrl^x Ctrl^s`

Fare pratica con la barra dei menu (in particolare con i menu Edit, Search e Buffers) e soprattutto con le combinazioni di tasti, ad esempio:

spostamento a fine riga: `Ctrl^e`

spostamento a inizio riga: `Ctrl^a`

ricerca di una stringa nel testo: `Ctrl^s <stringa>`

tutorial: `Ctrl^h t`

2 Script

Utilizzare l'editor di testi per creare un certo numero di file contenenti parole a caso; salvare i file con estensioni diverse (ad esempio *.txt*, *.dat*, ecc.).

Successivamente copiare il seguente script, che è una variante di quello visto a lezione (salvare con il nome *sposta*):

```
#!/bin/sh

if (test $# -ne 2) then
    echo "Uso: $0 estensione direttorio"
    exit -1
fi

if (test ! -d $2) then
```

```

    echo "Il direttorio $2 non esiste, adesso lo creo"
    mkdir $2
fi

echo "Dimmi la parola che devo cercare: "
read parola

for i in *.$1
do
    echo "Esamino il file $i"
    if (grep $parola $i) then
        echo "Sposto $i in $2"
        mv $i $2
    fi
done

```

Lo script serve a spostare in un direttorio (specificato dall'utente) tutti i file con una certa estensione (anch'essa specificata dall'utente) e che contengono una certa parola (che lo script chiede all'utente).

Eseguire lo script con il comando: *sh sposta <estensione> <direttorio>*

(ad esempio: *sh sposta txt dirprova*)

Se il direttorio specificato dall'utente non esiste, viene creato dallo script.

2.1 Estensione dello script

Si aggiunga allo script precedente la possibilità di invocarlo con due eventuali opzioni *-s* e *-n* (ad es. *sposta [-s] [-n] <estensione> <direttorio>* : la prima opzione consente all'utente di specificare se desidera la sovrascrittura (*-s*) dei file già contenuti nel direttorio destinazione, mentre la seconda opzione gli permette di specificare che la parola introdotta da tastiera deve (*default*) oppure non deve (*-n*) essere presente nei file esaminati.

3 Compilazione dei programmi

Esercizio 1 (scrivere il seguente programma C e salvarlo come *hello.c*):

```

#include <stdio.h>

int main()
{
    printf("hello world!\n");

    return 0;
}

```

Dallo Shell, utilizzare *gcc* nel seguente modo: *gcc -o hello hello.c*

L'opzione *-o* seguita da *hello* comunica a *gcc* che deve creare un file eseguibile chiamato *hello*.

Verificare che il programma funziona, scrivendo nello Shell: *./hello*

Se non si scrive *-o filename*, il compilatore *gcc* crea un eseguibile chiamato *a.out*.

In generale, la sintassi del comando gcc è: *gcc <opzioni> <filename>*

Oltre a *-o*, le opzioni più comuni sono:

- *-c* per creare il file oggetto anziché l'eseguibile: *gcc -c hello.c*
(verificare che il comando precedente genera il file oggetto *hello.o*)
- *-Wall* genera tutti i messaggi di warning che gcc può fornire
- *-pedantic* mostra tutti gli errori e i warning richiesti dallo standard ANSI C
- *-O -O1 -O2 -O3* servono per definire il livello di ottimizzazione (dal più basso al più alto)
- *-O0* per non avere nessuna ottimizzazione
- *-g* per un successivo debugging

Per creare l'eseguibile a partire dal file oggetto: *gcc hello.o -o hello*

Quest'ultima operazione prende il nome di linking, che in generale consiste nella risoluzione dei simboli tra programmi e l'inclusione di eventuali librerie.

Esercizio 2 (scrivere il seguente programma C e salvarlo come *squareroot.c*):

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

int main()
{
    double m, n;

    m = 12345;
    n = sqrt(m);
    printf("The square root of %f is %f.\n", m, n);

    return 0;
}
```

Verificare il fallimento del comando: *gcc -o squareroot squareroot.c*

Verificare il successo del comando: *gcc -o squareroot squareroot.c -lm*

Non basta infatti includere nel sorgente il file header *math.h*, serve l'opzione *-lm* per indicare al gcc di effettuare il linking della libreria matematica.

4 Debugging

Una applicazione per il debugging è *gdb*, che può essere messa in esecuzione dallo Shell o, meglio ancora, dall'editor Emacs.

Esercizio 3:

```
#include <stdio.h>

int main()
```

```

{
    double m, n;

    m = 1234.5678;
    n = 9.999;

    m = n;
    printf("m = %f, n = %f\n", m, n);

    return 0;
}

```

Il programma (per comodità chiamato *prova.c*) deve essere compilato con l'opzione *-g*:
gcc -g -o prova prova.c

A questo punto, da Emacs, utilizzare la combinazione di tasti *Alt^x* seguita da *gdb*. Alla comparsa, nella parte bassa della finestra, del messaggio *Run gdb (like this): gdb* premere invio. La finestra si divide in due parti: in quella superiore dovrebbe esserci il codice dell'ultimo programma editato (*prova.c*), in quella inferiore il prompt del *gdb*. Per una consultazione rapida delle possibilità offerte da questa applicazione, utilizzare il comando *help* nel prompt del *gdb* stesso.

Esercitazione con i comandi più comuni:

1. Caricamento del programma target: *file prova*
2. Esecuzione del programma: *run*
3. Le 10 righe attorno alla stringa *main*: *list main*
4. Inserire un breakpoint in corrispondenza di *main*: *break main* o *b main*
5. Informazioni sui breakpoint creati: *info breakpoints*
6. Se ora si utilizza il comando *run*, il programma viene eseguito fino al breakpoint
7. Esecuzione passo-passo: *step* (per ripetere basta premere invio)
8. Continuare l'esecuzione interrotta: *cont*
9. Ripetere i punti 2..6
10. Inserire un watchpoint per monitorare una variabile (*m*, ad esempio): *watch m*
11. Continuare l'esecuzione interrotta: *cont*
 Si noti che questa volta l'esecuzione non arriva fino in fondo, ma si ferma appena il valore di *m* cambia; per continuare nuovamente, fino al successivo cambiamento del valore della variabile, ripetere il comando *cont* (basta premere invio).

5 Argomenti da riga di comando

Per passare informazioni alla funzione *main()* si utilizzano in genere gli argomenti della riga di comando, scritti nello shell dopo il nome del programma da eseguire. Il C prevede l'uso di

due speciali parametri: *argc* e *argv*, utilizzati per ricevere gli argomenti della riga di comando. Il parametro *argc* è un intero che contiene il numero di argomenti che si trovano nella riga di comando. Il suo valore è sempre almeno pari a 1 in quanto il nome del programma viene considerato come primo argomento. Il parametro *argv* è un puntatore a un array di puntatori a caratteri. Ogni elemento di questo array punta a un argomento della riga di comando.

Esercizio 4:

```
#include <stdio.h>

int main(int argc, char **argv)
{
    int i;

    printf("This program is %s and his number of arguments is %d.\n", argv[0], argc);
    for (i = 1; i < argc; i++)
    {
        printf("The argument number %d is %s.\n", i, argv[i]);
    }
    return 0;
}
```

Esiste anche un terzo parametro che consente di fornire al programma tutte le variabili d'ambiente: *envp*.

Esercizio 5:

```
#include <stdio.h>

int main(int argc, char **argv, char **envp)
{
    char *p;

    while (p = *envp)
    {
        printf("%s.\n", p);
        envp += 1;
    }

    return 0;
}
```

6 Funzioni per operare sulle variabili d'ambiente

Dallo shell, creare la variabile d'ambiente *INCLUDE* assegnandole il valore *\$HOME/include* (la sottodirectory *include* del proprio direttorio utente):

export INCLUDE=\$HOME/include

Ciascun programma messo in esecuzione nello shell è un processo figlio dello stesso processo shell, da cui eredita le variabili d'ambiente. Un programma C può operare sulle proprie variabili d'ambiente grazie ad alcune funzioni fornite dalla Standard Library:

```
#include <stdlib.h>
char *getenv(const char *name);
int putenv(char *string);
```

Esercizio 6:

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

int main()
{
    char *path;

    /* ottieni e mostra il valore della variabile d'ambiente INCLUDE */
    path = getenv("INCLUDE");
    if (path != NULL)
    {
        printf("INCLUDE=%s\n", path);
    }

    /* imposta un nuovo valore per INCLUDE */
    if (putenv("INCLUDE=/usr/local/src/include") != 0)
    {
        printf("putenv() failed setting INCLUDE\n");
        return -1;
    }

    /* ottieni e mostra il valore della variabile d'ambiente INCLUDE */
    path = getenv("INCLUDE");
    if (path != NULL)
    {
        printf("INCLUDE=%s\n", path);
    }

    return 0;
}
```

7 Compressione dei file

Per comprimere un file, si può utilizzare il comando: *gzip <nomefile>*

In tal modo viene creato il file compresso *<nomefile>.gz*

Per decomprimere un file *.gz*, utilizzare: *gzip -d <nomefile>.gz*

Per comprimere N file in un unico *.gz*, bisogna anzitutto creare un archivio tar:

tar -cvf <nomearchivio>.tar <nomefile1> <nomefile2> .. <nomefileN>

In tal modo viene creato il file *<nomearchivio>.tar*, che è una concatenazione dei file originari.

A questo punto si può utilizzare: *gzip <nomearchivio>.tar*

per ottenere il file compresso *<nomearchivio>.tar.gz*

Per comprimere un direttorio si opera come nel caso di N file, ma invece di specificare al comando *tar* N nomi di file si specifica il nome del direttorio target.

Per ricavare i file o il direttorio originari da un *.tar*, si utilizza la seguente sintassi:

tar -xvf <nomearchivio>.tar

Per spiegazioni esaustive, fare riferimento ai manuali di *gzip* e *tar*.