Laboratorio di Sistemi Operativi
08 settembre 2014
Compito

Si risponda ai seguenti quesiti, giustificando le risposte.

1. L'utente `pippo` esegue il comando `ls -l` nella sua home directory, ottenendo il seguente output:

   ```
   ...
   drw-r--r--    2 pippo  users    68 Sep  2 11:42 tmp
   ...
   ```

   Successivamente tenta di eseguire il comando `cd tmp`, ma questo fallisce, riportando il messaggio d'errore `-bash: cd: tmp: Permission denied`.

   (a) Perché il comando `cd tmp` fallisce?

   (b) Cosa deve fare l'utente per riuscire a spostarsi nella directory `tmp`?

   Soluzione:

   (a) L'errore è dovuto al fatto che anche per l'owner (l'utente `pippo`) manca il permesso di esecuzione.

   (b) Per rimediare è sufficiente digitare (ad esempio) il seguente comando:

   ```
   chmod u+x tmp
   ```

2. Si completi il seguente script della shell `bash` che deve leggere dallo standard input un flusso di testo (linea per linea), producendo in output un istogramma di asterischi (`*`), in modo che la lunghezza di ogni linea di quest'ultimo rappresenti il numero di caratteri contenuto nella linea corrispondente del testo passato in input. Ad esempio, se il file di testo `prova.txt` contiene 4 linee di 7, 2, 1, 5 caratteri rispettivamente, l'istogramma prodotto deve essere il seguente:

   ```
   *******
   **
   *
   *****
   ```

   Il codice da completare è il seguente:

   ```
   IFS='\n'  # imposta il separatore di linea per il comando read
   while read -r linea # -r considera eventuali backslash come normali caratteri
   do
     ...
   done
   ```

   Soluzione:

   Esempio di soluzione:

   ```
   IFS='\n'
   while read -r linea
   do
     lunghezza=‘echo $linea | wc -c‘
     #echo $lunghezza
     i=0
     while test $i -lt $lunghezza
     do
       echo -n '*'
       i=$[$i+1]
     done
     echo
   done
   ```

3. Si mostri qual è l'output prodotto dai seguenti comandi:
   (a) `echo ababab | sed '1,$y/ab/yz/'`
   (b) `echo ababab | sed '1,$s/ab/_/'`
   (c) `echo ababab | sed '1,$s/ab/_/g'`

Soluzione:
(a) `yzyzyz`
(b) `_abab`
(c) `___`

4. Si progetti uno script della shell `rect.sh` che prenda in input sulla linea di comando due interi positivi e stampi un rettangolo di asterischi secondo lo schema dell'esempio seguente (prendendo quindi le misure dei lati come argomenti sulla linea di comando):

```
$ ./rect.sh 5 3
*****
*   *
*****
```

Soluzione:
Esempio di soluzione:

```
if test $# -ne 2
then
  echo 'Utilizzo dello script: rect.sh <m> <n>'
  exit 1
fi

if ! test $1 -gt 0 -a $2 -gt 0
then
  echo 'Le dimensioni del rettangolo devono essere degli interi >0'
  exit 2
fi

x=$1
y=$2

while test $y -gt 0
do
  while test $x -gt 0
  do
    if test $y -gt 1 -a $y -lt $2
    then
      if test $x -gt 1 -a $x -lt $1
      then
        echo -n " "
      else
        echo -n "*"
      fi
    else
```

```
      echo -n "*"
    fi
    x=$[$x-1]
  done
  x=$1
  y=$[$y-1]
  echo
done

exit 0
```

5. Classificare come vere o false le seguenti affermazioni (per quelle false giustificare le risposte):
   (a) Nel linguaggio C si definisce una struttura ricorsiva, specificando un membro di tipo puntatore, che fa riferimento ad una struttura dello stesso tipo di quella in cui è contenuto.
   (b) La system call `fork()` viene usata per generare *nuovi thread* all'interno del processo che la invoca.
   (c) Le system call della famiglia `exec` consentono di lanciare in esecuzione un nuovo processo a partire da un programma esterno, dopodiché il processo originale (ovvero, colui che ha invocato la system call di tipo `exec`) continua la propria esecuzione.
   (d) Una `pthread_cond_signal()` va eseguita soltanto se c'è un thread in stato di attesa (ovvero, che ha eseguito una `pthread_cond_wait()`) sulla corrispondente condition variable. Altrimenti la segnalazione viene "persa", dato che non c'è nessun thread in attesa di essa.
   (e) Le socket possono essere utilizzate come un meccanismo di comunicazione tra processi (sia che questi risiedano sulla stessa macchina che su macchine diverse connesse in rete).
   (f) La comunicazione stabilita tramite socket è di tipo bidirezionale.
   (g) Una socket in uso è solitamente legata ad un indirizzo e la natura di quest'ultimo dipende dal dominio di comunicazione della socket.

Soluzione:

(a) Vero.

(b) Falso. La system call `fork()` crea un nuovo processo, copiando il chiamante (padre).

(c) Falso. Questa famiglia di system call sovrascrive il processo chiamante, sostituendone il codice con quello del programma esterno.

(d) Vero.

(e) Vero.

(f) Vero.

(g) Vero.

6. Scrivere un programma C che esegua ad intervalli di *5 secondi* il comando `ps -ef`. Il programma deve continuare la sua esecuzione fintanto che l'utente non prema la combinazione di tasti `CTRL-C` (pressione contemporanea di `CTRL` e `C`). A quel punto prima di terminare l'esecuzione il programma deve visualizzare su standard output il numero di cicli compiuti.

Soluzione:

Esempio di soluzione:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>

int stop;

/* prototipo della funzione per la gestione del segnale SIGINT */
void catchint(int);

int main() {
  pid_t pid;
  int count=0;
  stop=0;
  static struct sigaction act;
  /* registrazione dell'handler */
  act.sa_handler = catchint;
  /* eventuali altri segnali saranno ignorati
     durante l'esecuzione dell'handler */
  sigfillset(&(act.sa_mask));
  sigaction(SIGINT, &act, NULL);
  while(!stop) {
    sleep(5);
    switch(pid=fork()) {
      case 0:
        execl("/bin/ps","ps","-ef",(char *)0);
        perror("Exec fallita!");
        break;
      case -1:
        perror("Fork fallita!");
        return 1;
      default:
        waitpid(pid,NULL,0);
        count++;
    }
  }
  printf("Terminazione; sono stati eseguiti %d cicli\n",count);

  return 0;
}

void catchint(int signo) {
  stop=1;
}
```