Laboratorio di Sistemi Operativi
21 Giugno 2021
Compito

Si risponda ai seguenti quesiti, giustificando le risposte.

1. (6 punti) Si scriva uno script della shell `check_file_tree_size.sh` che prenda come argomento sulla linea di comando un percorso, controlli che corrisponda ad una directory e attraversi ricorsivamente il file system a partire da essa, stampando alla fine il numero dei soli file incontrati e del totale di byte occupati da essi.

   Esempio:

   ```
   ./check_file_tree_size.sh .
   Numero di file: 7
   Numero di byte: 52493
   ```

   Esempio di soluzione:

   ```
   f ! test $# -eq 1
   then
           echo "Utilizzo $0 pathname"
       exit 1
   fi

   num_file=0
   num_bytes=0

   if test -e $1 -a -d $1
   then
       lista=‘find $1 -print 2>/dev/null‘
       for i in $lista
       do
         if test -f $i
         then
           num_file=$[$num_file+1]
           dim=$(cat $i | wc -c)
           num_bytes=$[$num_bytes+$dim]
         fi
       done
   fi

   echo "Numero di file: $num_file"
   echo "Numero di byte: $num_bytes"

   exit 0
   ```

2. (6 punti) Si scriva una pipeline che stampi a video l'elenco (senza ripetizioni e ordinato lessicograficamente) delle shell di login assegnate agli utenti di sistema.

   Suggerimento: si ricorra al file `/etc/passwd` dove ogni linea corrisponde ad un account del sistema ed è una successione di campi separati dai due punti (`:`). Il campo relativo alla shell di login è il settimo.

   Esempio di soluzione:

   ```
   cat /etc/passwd | cut -d ':' -f7 | sort | uniq
   ```

3. (6 punti) Si considerino le seguenti dichiarazioni in C:

```
int *a=(int*)malloc(10*sizeof(int));
int b[10];
```

Si dica, per ognuno dei seguenti comandi, se è lecito oppure no (motivando la risposta):

1. a[4+5]=67;
2. *(a+4+5)=67;
3. b[10]=68;
4. *(b+10)=68;
5. a=b+1;
6. b=a+1;

Risposte:

1. a[4+5]=67; è corretto: in generale, a[i] è sinonimo di *(a+i) e a è un puntatore ad interi che punta ad un'area di memoria allocata dinamicamente per contenere 10 interi; pertanto a[4+5] punterà all'indirizzo di memoria dell'ultimo intero (a[9]);
2. *(a+4+5)=67; è corretto: a è un puntatore ad interi che punta ad un'area di memoria allocata dinamicamente per contenere 10 interi; pertanto a+4+5 punterà all'indirizzo di memoria del decimo intero (ovvero, l'ultimo);
3. b[10]=68; è scorretto in quanto b è il nome di un vettore di 10 interi e si sta tentando di accedere all'undicesimo;
4. *(b+10)=67; è scorretto: in generale, *(b+i) è sinonimo di b[i] (vedi punto precedente);
5. a=b+1; è corretto: a è un puntatore a interi a cui viene assegnato l'indirizzo del secondo elemento (di tipo intero) del vettore b;
6. b=a+1; non è corretto: il nome di un array non può apparire a sinistra di un comando di assegnamento in questo modo (è possibile assegnare soltanto i singoli elementi del vettore).

4. (4 punti) Si dica qual è l'output generato dal seguente programma C:

```
#include <stdio.h>

int main() {
  int i,j;
  for(i=2; i>=0; i--) {
    for(j=0; j<i; j++)
      printf(" ");
    for(j=0; j<(3-i)*2-1; j++)
      printf("*");
    printf("\n");
  }

  return 0;
}
```

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

Laboratorio di Sistemi Operativi
21 Giugno 2021
Compito

5. (10 punti) Si scriva un programma C `dice.c` che simuli il lancio di un dado. Il programma prende da linea di comando il numero di volte `n` che il punteggio massimo possibile (ovvero 6) deve uscire come esito del lancio. Dopodiché inizia a generare un numero pseudo-casuale compreso tra 1 e 6 ogni 5 secondi.

   Raggiunto il numero `n` di lanci con punteggio massimo, termina la propria esecuzione stampando il messaggio `Si sono verificati n lanci con punteggio massimo`.

   Tuttavia, prima di giungere a terminazione, se l'utente preme Ctrl-C (ovvero, preme contemporaneamente i tasti Ctrl e C), il programma termina immediatamente, stampando il numero di lanci con punteggio massimo eventualmente generati fino a quel momento.

   Esempio: se l'utente digita

   `./dice 7`

   e non preme `Ctrl-C`, il programma si fermerà dopo che il numero 6 sarà uscito per la settima volta, stampando

   `Si sono verificati 7 lanci con punteggio massimo.`

   Altrimenti uscirà al momento della pressione di `Ctrl-C` stampando il numero di volte che il numero 6 sarà uscito fino a quel momento.

   Suggerimento: si ricorda che, per generare dei numeri pseudo-casuali, è sufficiente utilizzare la funzione di libreria `random()` come segue:

```c
#include <stdlib.h>
#include <time.h>
...
srandom(time(NULL)); // per inizializzare il seme con la data/ora attuale, assicurandosi di generare sequenze diverse ad ogni esecuzione
...
long int r;
r=random(); // genera un numero pseudo-casuale con valore compreso tra 0 e RAND_MAX e lo assegna alla variabile r
```

Esempio di soluzione:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <signal.h>
#include <unistd.h>

int stop=0;

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

int main(int argc, char** argv) {
  long int max_count=atoi(argv[1]);
  long int max_current_count=0;
  srandom(time(NULL));
  signal(SIGINT, catchint);
  while(max_current_count<max_count && !stop) {
    long int r=random();
    int score=r%6+1;
    if(score==6)
```

```
22        max_current_count++;
23      printf("Punteggio dell'ultimo lancio: %d.\n",score);
24      sleep(5);
25    }
26
27    printf("Si sono verificati %ld lanci con punteggio massimo.\n",
          max_current_count);
28
29    return 0;
30  }
```