Laboratorio di Sistemi Operativi
22 settembre 2015
Compito

Si risponda ai seguenti quesiti, giustificando le risposte.

1. (a) Cosa si intende con ridirezione da linea di comando (altrimenti nota come "here document")?
   (b) Si faccia un esempio di utilizzo della ridirezione da linea di comando.

Soluzione:

(a) Con l'espressione "ridirezione da linea di comando" (i.e., *here document*) si intende la possibilità di fornire in input ad un comando od uno script del testo "catturato" dallo standard input (i.e., dalla tastiera) come se provenisse da un file.

(b) Ecco un esempio di *here document*:

```
> wc <<fine_esercizio
? se non so rispondere
? a questa domanda
? forse mi conviene ritirarmi...
? fine_esercizio
 3 11 69
```

Praticamente quanto segue l'operatore `<<` assume il ruolo di marcatore di fine input: la shell tiene traccia di tutto quanto viene digitato fintanto che non incontra il marcatore su una linea isolata. A questo punto fornisce il testo raccolto (escluso il marcatore) come input al comando (in questo caso `wc`) come se provenisse da un normale file di testo, per farlo processare.

2. Qual è l'effetto dei seguenti comandi?

   1. `ls -l | grep '^-'`
   2. `n=‘wc -l registro.txt | cut -d' ' -f1‘` (si supponga che l'output di `wc -l registro.txt` sia la stringa `10 registro.txt`);
   3. `echo $n` (dove `n` è la variabile inizializzata nel punto precedente).

   Attenzione: nel punto 2 gli apici esterni sono dei *backquote* (apici rovesciati).

Soluzione:

1. Il comando `ls -l | grep '^-'` seleziona dall'output di `ls -l` soltanto le linee che iniziano con il carattere `-`. Quindi vengono stampate a video soltanto le linee che corrispondono a file ordinari (i.e., non directory, link ecc.) in quanto il trattino è il carattere che li contraddistingue nell'output di `ls -l`.
2. Il comando `n=‘wc -l registro.txt | cut -d' ' -f1‘` opera una *command substitution* assegnando alla variabile `n` il risultato di `wc -l registro.txt` cut -d' ' -f1|, ovvero, `10` in questo caso (infatti `cut`, utilizzando come separatore di campo lo spazio, seleziona il primo campo della stringa prodotta da `wc -l registro.txt`, ovvero, texttt10).
3. Il comando `echo $n` stampa a video il valore della variabile `n` assegnato al punto precedente, ovvero, `10`.

3. Si predisponga uno script della shell `line_ith.sh` che prenda come argomento sulla linea di comando il percorso di un file di testo ed un intero e compia le seguenti azioni:
   (a) controlli il numero degli argomenti forniti, terminando la propria esecuzione nel caso il numero sia diverso da due;
   (b) controlli che il percorso fornito come primo argomento corrisponda ad un file e sia leggibile dall'utente (altrimenti l'esecuzione deve terminare);

(c) controlli che il numero intero fornito come secondo argomento sia un numero compreso fra `1` ed il numero di linee del file corrispondente al primo argomento;

(d) tra le righe del file passato come primo argomento stampi su standard output quella corrispondente al numero passato come secondo argomento.

Ad esempio il comando

```
> line_ith testo.txt 5
```

deve emettere su standard output la quinta riga del file `testo.txt`.

Suggerimento: si ricordi che `tail -n +k f` stampa le linee di `f`, a partire dalla `k`-esima, mentre `head -n +k f` stampa le prime `k` linee di `f`.

Soluzione:

Esempio di soluzione:

```
#!/bin/bash

if test $# -ne 2
then
  echo "Utilizzo dello script: $0 <file> <n. di linea>"
  exit 1
fi

if ! test -f $1 -a -r $1
then
  echo "$1 non è un file oppure non è leggibile."
  exit 2
fi

num_linee=`wc -l $1 | cut -d' ' -f1`

if ! test $2 -ge 1 -a $2 -le $num_linee
then
  echo "$2 non è un numero compreso fra 1 e $num_linee."
  exit 3
fi

tail -n +$2 $1 | head -n +1

exit 0
```

4. Sia data la seguente struttura ricorsiva in C per la rappresentazione di alberi binari:

```
struct bintree {
  int val;
  struct bintree *left;
  struct bintree *right;
};
```

dove `val` rappresenta il valore del nodo, mentre `left` e `right` puntano, rispettivamente, al figlio sinistro ed al figlio destro (tali puntatori assumono il valore `NULL` quando non esistono i rispettivi figli).

Si scriva il codice di una funzione avente il seguente prototipo:

```
void raddoppia(struct bintree *root);
```

che raddoppi il valore del membro `val` di ogni nodo dell'albero binario puntato da `root`.

Soluzione:

Esempio di soluzione:

```
1 void raddoppia(struct bintree *root) {
2   if(root!=NULL) {
3     root->val *= 2;
4     raddoppia(root->left);
5     raddoppia(root->right);
6   }
7 }
```

5. Si consideri il seguente programma `triangle.c`:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  int i,j,n=atoi(argv[1]);
  for(i=0; i<n; i++) {
    for(j=0; j<i+1; j++)
      printf("*");
    printf("\n");
  }
}
```

Una volta compilato, l'eseguibile accetta un intero sulla linea di comando e stampa un triangolo di asterischi nel modo seguente:

```
> ./triangle 5
*
**
***
****
*****
```

Si vuole ottenere lo stesso effetto con un altro programma `triangle_thread.c` che genera `n` thread figli (dove `n` è il valore dell'intero passato sulla linea di comando) facendo stampare una linea ad ogni figlio:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <fcntl.h>
#include <sys/types.h>

void *print_row(void *n) { // funzione che stampa una linea di n asterischi
  int i;
  sleep(random()%5+1); // attende da 1 a 5 secondi
  for(i=0;i<((int)n);i++)
    printf("*");
  printf("\n");
```

```
}

int main(int argc, char *argv[]) {
  int i,j,n=atoi(argv[1]);
  pthread_t *thr;
  thr=(pthread_t *)calloc(n,sizeof(pthread_t)); // alloco memoria per un array di n thread id

  for(i=1; i<=n; i++) {
    if(pthread_create(&thr[i-1],NULL,print_row,(void *)i)!=0) { // creo il thread i-esimo
      perror("Errore nella creazione del thread.\n");
      exit(1);
    }
  }

  for(i=1;i<=n;i++)
    pthread_join(thr[i-1],NULL); // attendo la terminazione dell'i-esimo thread figlio

  free(thr);                    // libero la memoria allocata per il vettore di thread id

  return 0;
}
```

(a) Il programma `triangle_thread.c` funzionerà correttamente, ovvero, produrrà in output il triangolo di asterischi? Perché?

(b) Se si è risposto negativamente alla precedente domanda, modificare (si può cancellare/modificare/aggiungere codice a piacimento) il programma in modo che rispetti la consegna.

Soluzione:

(a) Il programma `triangle_thread.c` non funzionerà correttamente in quanto la funzione `print_row()` introduce dei ritardi casuali prima di stampare gli asterischi di una linea. Inoltre, anche eliminando tali ritardi, non è affatto detto che il sistema esegua i singoli thread nell'ordine in cui sono stati creati.

(b) È possibile apportare diverse correzioni (eliminare i ritardi e rendere i thread FIFO oppure usare una variabile che rappresenti il turno del thread corretto ed utilizzare mutex e condition variable per sincronizzare le stampe delle singole linee). Tuttavia la soluzione più semplice consiste nell'attendere la terminazione del thread appena creato prima di procedere alla creazione del successivo. Per far ciò, è sufficiente eliminare il ciclo `for` finale con i join ai vari thread ed introdurre una chiamata a `pthread_join` all'interno del primo ciclo `for` come segue:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <pthread.h>
5  #include <fcntl.h>
6  #include <sys/types.h>
7
8  void *print_row(void *n) { // funzione che stampa una linea di n
       asterischi
9    int i;
10   sleep(random()%5+1); // attende da 1 a 5 secondi
11   for(i=0;i<((int)n);i++)
12     printf("*");
13   printf("\n");
14 }
```

```
15
16 int main(int argc, char *argv[]) {
17   int i,j,n=atoi(argv[1]);
18   pthread_t *thr;
19   thr=(pthread_t *)calloc(n,sizeof(pthread_t)); // alloco memoria
         per un array di n thread id
20
21   for(i=1; i<=n; i++) {
22     if(pthread_create(&thr[i-1],NULL,print_row,(void *)i)!=0) {
          // creo il thread i-esimo
23       perror("Errore nella creazione del thread.\n");
24       exit(1);
25     }
26     pthread_join(thr[i-1],NULL); // attendo la terminazione dell'
          i-esimo thread figlio
27   }
28
29   free(thr);                          // libero la memoria allocata
       per il vettore di thread id
30
31   return 0;
32 }
```