Caselle riservate

Ex. 1	
Ex. 2	
Ex. 3	
Ex. 4	
Ex. 5	
Ex. 6	
Tot.	

Sistemi Operativi

Compito d'esame

06 Luglio 2018

Matricola _____ Cognome _____ Nome _____

Docente: ◯ Quer ◯ Sterpone

Non si possono consultare testi, appunti o calcolatrici a parte i formulari distribuiti dal docente. Risolvere gli esercizi negli spazi riservati. Fogli aggiuntivi sono permessi solo quando strettamente necessari. Riportare i passaggi principali.
Durata della prova: 100 **minuti.**

1. Implementare un programma concorrente multi-thread in grado di generare un albero di thread di altezza N. Ciascun thread a livello dispari dell'albero (1, 3, 5, etc.) deve generare 2 thread. Ciascun thread a livello pari dell'albero (2, 4, 6, etc.) deve generare 1 unico thread. In pratica quindi, il thread iniziale (al livello 1) deve generare 2 thread, mentre i 2 thread generati (al livello 2) devono generare ciascuno 1 thread, e i 2 thread generati (al livello 3) devono generare ciascuno 2 thread, e cosí via. I thread all'ultimo livello dell'albero devono stampare il proprio identificatore di thread.

   Il valore di N viene ricevuto dal programma sulla riga di comando.

   Si ricordano i seguenti prototipi:

```
int pthread_create (pthread_t *tid, const pthread_attr_t *attr, void
                    *(*startRoutine)(void *), void *arg);
void pthread_exit (void *valuePtr);
int pthread_join (pthread_t tid, void **valuePtr);
```

2. Un programma richiama una funzione ricorsiva di nome `recur`, procedendo secondo lo schema seguente:

```
...
main (int argc, char *argv[]) {
  ...
  recur (...);
  ...
}

void recur (...) {
}
```

Dato che la funzione ricorsiva `recur` puó avere tempi di esecuzione decisamente elevati, il programmatore desidera forzarne la terminazione dopo un certo numero di secondi. Si chiarisca come sia possibile utilizzare i segnali e le system call ad essi associati, per forzare la terminazione della funzione `recur` dopo un numero di secondi ricevuto dal `main` sulla riga di comando. Si osservi che è possibile tanto aggiungere funzioni quanto modificare la funzione `recur` oppure quella principale.

3. Si illustrino le caratteristiche delle *pipe* per la comunicazione e la sincronizzazione tra processi.

   Tramite un tratto di codice C, si illustri l'utilizzo di una pipe nel caso essa sia utilizzata per trasferire delle stringhe di lunghezza variabile tra due processi. Il codice dovrà includere le operazioni di creazione dei processi coinvolti nonché quelle di apertura e chiusura delle pipe.

4. Uno script BASH riceve due parametri al momento dell'esecuzione: il nome di un file e un intervallo di tempo (in secondi). Il file contiene un numero indefinito di nomi di processi in ragione di un nome per ciascuna riga del file. Lo script deve verificare a intervalli di tempo pari a quello specificato sulla riga di comando se i processi indicati nel file sono ancora in fase di esecuzione all'interno del sistema. Si visualizzi il nome di tutti i processi in esecuzione nonché il numero dei processi zombie (cioè nello stato Z). Lo script venga eseguito sino a quando nessuno dei processi riportati nel file risulta in esecuzione.

   Si ricorda che il comando `ps -el` fornisce un output simile al seguente:

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 46340 -      ?        00:00:01 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:00 kthreadd
0 R  1000  7295  2344  0  80   0 -  7229 -      pts/17   00:00:00 ps
...
```

5. Due file contengono un numero di righe indefinito. Nel primo ciascuna riga contiene due orari con formato:

   `HH:MM:SS HH:MM:SS`

   mentre il secondo contiene un unico orario su ciascuna riga.

   Si scriva uno script AWK in grado di:

   - ricevere il nome dei due file sulla riga di comando.
   - verificare che su ciascuna riga del primo file il primo orario preceda il secondo, visualizzando un messaggio di errore in caso contrario.
   - per ogni intervallo orario non negativo contenuto nel primo file, calcolare quanti orari del secondo file ricadono in esso. Visualizzare tale informazione al termine dell'esecuzione.

   Ad esempio se i due file sono i seguenti:

Primo file		**Secondo file**			**Output**
10:10:10	13:12:24	11:11:11	10:10:10	13:12:24	2
15:00:25	16:23:25	12:12:12	15:00:25	16:23:25	1
17:00:21	17:00:19	13:13:13	17:00:21	17:00:19	negativo
09:00:00	11:00:00	14:14:14	09:00:00	11:00:00	0
		15:15:15			

   il terzo intervallo è negativo e va segnalato, mentre nel primo ricadono due ore, nel secondo una e nel quarto nessuna. L'output è quindi quello indicato a destra della figura precedente.

6. Il comando `ls -la` fornisce il seguente risultato:

```
total 796
drwxrwxr-x  2 quer quer   4096 Jun 27 12:38 .
drwx------ 10 quer quer   4096 Jan 18 15:11 ..
-rw-rw-r--  1 quer quer 106815 Jun 27 12:15 20180706.pdf
-rw-rw-r--  1 quer quer   6865 Feb 22 10:21 20180706.tex
...
```

   Chiarire il significato dei campi e dei valori riportati dal comando.

   Si chiarisca inoltre, con l'aiuto di ausili grafici, il significato dei seguenti termini: directory entry, i-node, hard-link e soft-link. Si riportino infine i comandi per creare hard- e soft-link.