Sistemi Operativi – Lab 9 – 9.12.19 - A.A. 2019/2020 - Prof. L. Sterpone

Goal: Esercizi su script bash.

Esercizio 1 (tutorial bash):

Si scriva uno script che controlla che un dato file sia presente nei direttori del PATH corrente. Il nome del file è passato come primo e unico parametro. Quando il nome viene trovato nei sotto-direttori viene stampato a video il percorso in cui il file si trova.

```
Esempio riga di comando:

$myscript file

Contenuto dello script:
#!/bin/bash
case $# in
0) echo almeno un argomento; exit 1;;
esac
for i in `echo $PATH | tr ':' ' '`
do
if test -e $i/$1 #se esiste
then
     if test -f $i/$1
          then echo $i/$1 #se è un file
     fi
fi
done
```

Esercizio 2 (bash):

Scrivere uno script bash che controlli periodicamente ogni X secondi se l'utente U (X e U passati come argomenti) abbia eseguito il logout.

Esercizio 3 (bash):

Si realizzi uno script bash in grado di controllare lo stato della connessione ad internet. Lo script non riceve parametri e dovrà riportare in uscita un messaggio di stato (e.g., Connesso/Non connesso ad Internet).
Suggerimento: per verificare o stato della connessione usare ping www.google.com.

Sistemi Operativi – Lab 10 – 16.12.19 - A.A. 2019/2020 - Prof. L. Sterpone

Goal: Esercizi su script bash.

Esercizio 1 (bash):

Scrivere uno script in grado di trasformare ricorsivamente tutti i nomi di file contenuti in una directory sostituendo i caratteri passati come parametri da minuscoli in maiuscoli.

Suggerimento: attenzione alle operazioni critiche sul proprio file system!

Esercizio 2 (bash):

Si scrivano due script bash che si alternino all'infinito nell'esecuzione. Ogni script stampa una propria stringa a terminale. Fare in modo che la semplice pressione del tasto "s" arresti entrambi.

Esercizio 3 (bash):

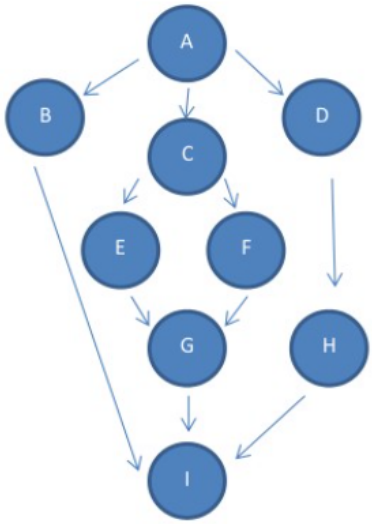
Scrivere uno script che calcoli il totale della memoria fisica utilizzata dai processi in esecuzione nel sistema all'istante del lancio e ne valuti il valore percentuale rispetto alla quantità di RAM installata nel sistema.

Sistemi Operativi – Lab 11 – 13.01.20 - A.A. 2019/2020 - Prof. L. Sterpone

Goal: Implementazione e uso dei semafori.

Esercizio 1 (tutorial semafori):

Implementare il grafo di precedenza illustrato nella figura seguente utilizzando i threads e il meccanismo di sincronizzazione basato sui semafori considerando i threads come non ciclici.



Tutorial:

```
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>

sem_t semB,semC,semD,semE,semF,semG,semH, semI;

void *tfA (void *arg) {

    sleep(1);
    printf("   A\n");
    sem_post(&semB);
    sem_post(&semC);
    sem_post(&semD);

}

void *tfB (void *arg) {

    sem_wait(&semB);
    sleep(2);
    printf("B  ");
    sem_post(&semI);

}

void *tfC (void *arg) {

    sem_wait(&semC);
    sleep(3);
    printf("C  ");
    sem_post(&semE);
    sem_post(&semF);

}

void *tfD (void *arg) {

    sem_wait(&semD);
    sleep(4);
    printf("D\n");
    sem_post(&semH);

}
```

```
void *tfE (void *arg) {

    sem_wait(&semE);
    sleep(1);
    printf(" E ");
    sem_post(&semG);

}

void *tfF (void *arg) {

    sem_wait(&semF);
    sleep(2);
    printf("  F\n");
    sem_post(&semG);

}

void *tfG (void *arg) {

    sem_wait(&semG);
    sem_wait(&semG);
    sleep(1);
    printf("   G");
    sem_post(&semI);

}

void *tfH (void *arg) {

    sem_wait(&semH);
    sleep(5);
    printf("  H\n");
    sem_post(&semI);

}

void *tfI (void *arg) {

    sem_wait(&semI);
    sem_wait(&semI);
    sem_wait(&semI);
    sleep(1);
    printf("   I\n");

}

int main (int argc, char **argv) {
  pthread_t thIDA,thIDB,thIDC,thIDD, thIDE,thIDF,thIDG,thIDH,thIDI;

  sem_init (&semB, 0, 0);
  sem_init (&semC, 0, 0);
  sem_init (&semD, 0, 0);
  sem_init (&semE, 0, 0);
  sem_init (&semF, 0, 0);
  sem_init (&semG, 0, 0);
  sem_init (&semH, 0, 0);
  sem_init (&semI, 0, 0);

  pthread_create (&thIDA, NULL, tfA, (void*)NULL);
  pthread_create (&thIDB, NULL, tfB, (void*)NULL);
  pthread_create (&thIDC, NULL, tfC, (void*)NULL);
  pthread_create (&thIDD, NULL, tfD, (void*)NULL);
  pthread_create (&thIDE, NULL, tfE, (void*)NULL);
  pthread_create (&thIDF, NULL, tfF, (void*)NULL);
  pthread_create (&thIDG, NULL, tfG, (void*)NULL);
  pthread_create (&thIDH, NULL, tfH, (void*)NULL);
  pthread_create (&thIDI, NULL, tfI, (void*)NULL);

  (void*)pthread_join(thIDA, (void*)NULL);
  (void*)pthread_join(thIDB, (void*)NULL);
  (void*)pthread_join(thIDC, (void*)NULL);
  (void*)pthread_join(thIDD, (void*)NULL);
  (void*)pthread_join(thIDE, (void*)NULL);
  (void*)pthread_join(thIDF, (void*)NULL);
  (void*)pthread_join(thIDG, (void*)NULL);
  (void*)pthread_join(thIDH, (void*)NULL);
  (void*)pthread_join(thIDI, (void*)NULL);


  return 0;
}
```

Esercizio 2 (semafori):

Implementare il grado di precedenza illustrato nella figura seguente utilizzando sia i threads (partendo dalla soluzione dell'esercizio precedente) che i processi (implementando i semafori con utilizzo delle pipe) considerando i thread e i processi come ciclici.