Caselle riservate

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

Sistemi Operativi

Compito d'esame

30 Gennaio 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. Si riporti il grafo di controllo del flusso e l'albero di generazione dei processi a seguito dell'esecuzione del programma riportato sulla sinistra. Si supponga che tale programma venga eseguito con due parametri sulla riga di comando: il nome dell'eseguibile corrispondente al programma sulla destra (`pgrmB`) e il valore intero 2. Si indichi inoltre quali messaggi vengono generati su video e per quale motivo.

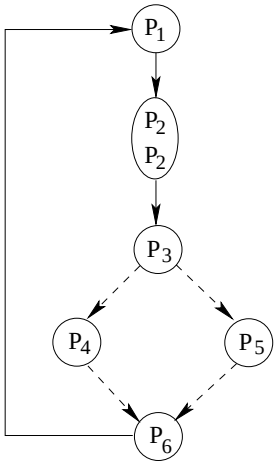
pgrmA

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main (int argc, char *argv[]) {
  char str[20];
  int i, n = atoi (argv[2]);
  for (i=0; i<n; i++) {
    if (fork()>0) {
      printf ("exec: %s %s\n", argv[1], argv[2]);
      execlp (argv[1], argv[1], argv[2], (char *) 0);
    } else {
      sprintf (str, "%s %d", argv[1], i);
      printf ("system A: %s\n", str);
      system (str);
    }
  }
  return (1);
}
```

pgrmB

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main (int argc, char *argv[]) {
  char str[20];
  int n = atoi (argv[1]);
  if (n<=0)
    return (1);
  sprintf (str, "%s %d", argv[0], n-1);
  printf ("system B: %s\n", str);
  system (str);
  return (1);
}
```

2. Dato il seguente grafo di precedenza, realizzarlo utilizzando il **minimo** numero possibile di semafori. I processi rappresentati devono essere processi ciclici (con corpo del tipo `while(1)`). Si noti che il processo P_2 (instanziato una sola volta) deve essere eseguito 2 volte per ogni iterazione del ciclo esterno prima di iniziare P_3 (ma non deve contenere codice ripetuto o costrutti iterativi oltre il principale). Inoltre gli archi tratteggiati da P_3 a P_4 e P_5 indicano che P_4 e P_5 devono essere eseguiti in mutua esclusione, ovvero deve essere eseguito P_4 oppure P_5 ma non entrambi. Utilizzare le primitive `init`, `signal`, `wait` e `destroy`. Riportare il corpo dei processi ($P_1, \ldots, P_6$) e l'inizializzazione dei semafori.



Si riportino inoltre in pseudo-codice una possibile implementazione reale delle funzioni `signal` e `wait`.

3. Si illustri il problema dei *Readers e Writers* e se ne riporti la soluzione mediante primitive semaforiche nel caso di precedenza ai Readers.

   Si utilizzi quindi tale schema per descrivere la sincronizzazione necessaria per organizzare il flusso automobilistico su un ponte (o un tunnel) percorribile solo in senso alternato.

   Si supponga infine che per ragioni di sicurezza, il ponte possa essere attraversato da un massimo di 10 veicoli, ovvero che possano transitare sul ponte un massimo di 10 veicoli alla vota e se si eccede tale numero il flusso debba essere interrotto, per poi essere ripreso nello stesso senso o in senso opposto. Si estenda lo schema precedente per comprendere tale caso.

4. Il comando `df <file>` mostra lo spazio su disco disponibile sul file system contenente `<file>` (misurato in blocchi da 1KB). Ad esempio, nell'output di `df /data/backup` si specificano le caratteristiche dell'unitá fisica contenente tale file:

```
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sda7       41170492 5881472  33174616  16% /data
```

il secondo, terzo e quarto campo mostrano lo spazio totale, usato e disponibile sul file system contenente `/data/backup`. I campi sono suddivisi da un numero qualunque di spazi. Si supponga che nessun altro carattere di separazione diverso dallo spazio possa apparire in tale output e che lo spazio non compaia in nessun altro punto.

Si scriva uno script BASH che riceva in ingresso il percorso di un file sorgente e un percorso destinazione. Lo script deve:

- Controllare il corretto passaggio dei parametri.
- Effettuare la copia in background del file sorgente nel percorso destinazione.
- Analizzare ad intervalli regolari di un secondo lo spazio occupato sul file system destinazione, visualizzando a video la percentuale di avanzamento dell'operazione di copia (si supponga che questa sia l'unica operazione in corso su tale file system), fino al termine della stessa.

Suggerimento: il comando `sleep <n>` può essere utilizzato per mettere in pausa lo script per `<n>` secondi.

5. Un file contiene un numero di righe indefinito, in ciascuna delle quali sono specificate due date con formato:

   `GG:MM:AAAA GG:MM:AAAA`

   Si osservi che su ciascuna riga sono contenute date appartenenti allo stesso anno solare. Si scriva uno script AWK che ricevuto il nome del file sulla riga di comando sia in grado di:

   - verificare che su ciascuna riga la prima data preceda la seconda data, visualizzando un messaggio di errore in caso contrario.
   - visualizzare il numero di giorni che separano le due date per ciascuna riga del file.

   Si utilizzi il comando `cal`, invocato all'interno dello script AWK, per calcolare il numero di giorni presenti nei vari mesi. Ad esempio, si ricorda che il comando `cal 01 2018` fornisce il seguente risultato:

```
    January 2018
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

6. Si consideri il seguente insieme di processi:

Processo	Tempo arrivo	Burst Time
P_0	0	22
P_1	0	26
P_2	5	19
P_3	12	17
P_4	23	18

Rappresentare mediante diagramma di Gantt l'esecuzione di tali processi utilizzando gli algoritmi di scheduling RR (Round Robin) e SRTF (Shortest Remaining Time First). Calcolare il tempo di completamento nonchè il tempo di attesa medio per ciascun processo. Si consideri un quantum temporale di 10 unità di tempo.

Si chiarisca infine che cosa si intende per scheduling a code multi-livello.