Caselle riservate

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

Sistemi Operativi

Compito d'esame

06 Febbraio 2015

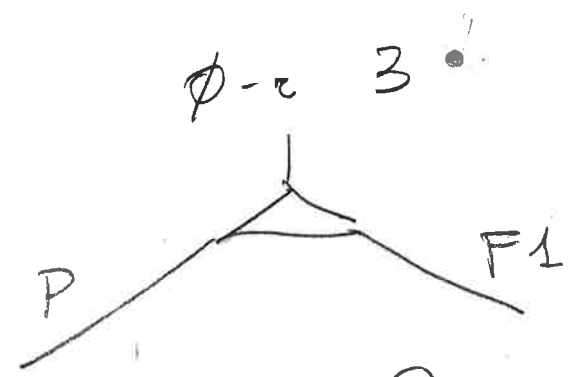
Matricola _____ Cognome _____ Nome _____

Non si possono consultare testi, appunti o calcolatrici. Riportare i passaggi principali. L'ordine sarà oggetto di valutazione.
Durata della prova: 100 **minuti.**

1. Si riporti l'albero di generazione dei processi a seguito dell'esecuzione del seguente tratto di codice C. Si supponga che il programma venga eseguito con un unico parametro, il valore intero 3, sulla riga di comando. Si indichi inoltre che cosa esso produce su video e per quale motivo.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
int main (int argc, char *argv[]) {
  int i;
  char str[50];
  i = atoi (argv[1]);
  printf ("0 - run with i=%d\n", i); fflush (stdout);
  if (i<=0) exit (0);
  if (fork () > 0) {
    sprintf (str, "echo 1 - run with i=%d", i);
    system (str);
    sprintf (str, "%d", i-1);
    execlp (argv[0], argv[0], str, NULL);
  } else {
    sprintf (str, "echo 2 - run with i=%d", i);
    system (str);
    sprintf (str, "%d", i-2);
    execlp (argv[0], argv[0], str, NULL);
  }
  exit (0);
}
```

(1)

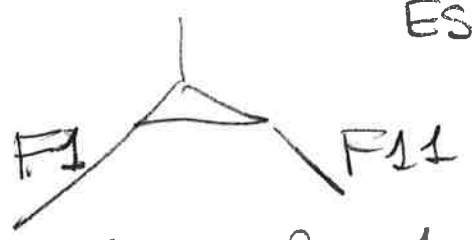


OGNI STAMPA È DEL TIPO
0
1 – run with i = #
2

1 - r 3 •
R
0 - r 2 •

2 - r 3 •
R
0 - r 1 •

ALBERO ESECUZIONE

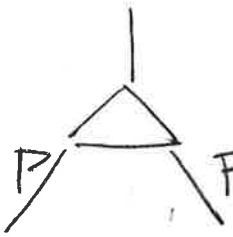


1 - r 2 •
R
0 - r 1 •

2 - r 2 •
R
0 - r Ø •
STOP

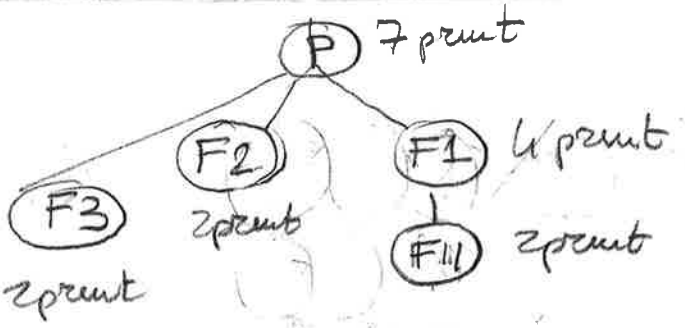
1 - r 1 •
R
0 - r Ø •
STOP

2 - r 1 •
R
0 - r - 1 •
STOP



1 - r 1 •
R
0 - r Ø •
STOP

2 - r 1 •
R
0 - r - 1 •
STOP



ALBERO GENERAZIONE PROCESSI

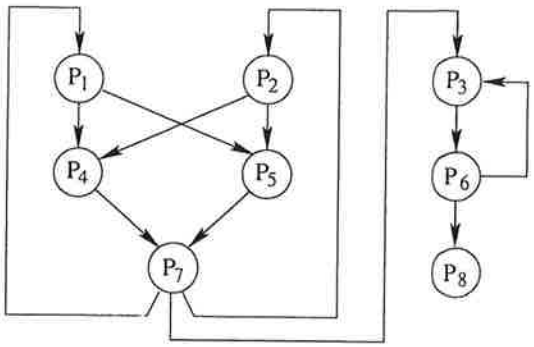
3 con 0 - run with	3	2 1	1 0 0 - 1	0 - 1
4 con 1 - run with		3	2 1	1
4 con 2 - run with		3	2 1	1

17 RIGHE di OUTPUT

2. Si indichi sinteticamente il significato dei seguenti termini e/o concetti.

Funzioni e system call	VEDERE LUCI DI UNITA' 02 SEZIONE 02 PAGINE 18-24
Process Control Block	11 04 11 02 11 5-7
Processo init	11 04 11 01 11 9
Processo zombie	11 04 11 01 11 47
Funzione rientrante e race condition	11 04 11 05 11 35-37 04 05 38-43
Half-duplex pipe	11 04 11 07 11 10
Context switching	11 04 11 02 11 8-9
Thread kernel e thread utente	11 05 11 01 11 15-23

3. 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)`). Utilizzare le primitive `init`, `signal`, `wait` e `destroy`. Indicare gli eventuali archi superflui e riportare il corpo dei processi ($P_1, \ldots, P_8$) e l'inizializzazione dei semafori.



Sem S1, S2, S3A, S3B, S4, S5, S6, S7, S8;
init(S4,1); init(S2,1); init(S3A,0);
init(S3B,1); init(S4,0); ... ; init(S8,0);

P1
```
while(1) {
    wait(S1);
    P1();
    signal(S4);
    signal(S5);
}
```

P2
```
while(1) {
    wait(S2);
    P2();
    signal(S4);
    signal(S5);
}
```

P3
```
while(1) {
    wait(S3A);
    wait(S3B);
    P3();
    signal(S6);
}
```

P4
```
while(1) {
    wait(S4);
    wait(S4);
    P4();
    signal(S7);
}
```

P5
```
while(1) {
    wait(S5);
    wait(S5);
    P5();
    signal(S7);
}
```

P6
```
while(1) {
    wait(S6);
    P6();
    signal(S3B);
    signal(S8);
}
```

P7
```
while(1) {
    wait(S7);
    wait(S7);
    P7();
    signal(S1);
    signal(S2);
    signal(S3A);
}
```

P8
```
while(1){
    wait(S8);
    P8();
}
```

```
destroy(S1);
}
destroy(S8);
```

4. Un file, creato dal comando "`ls -laR`", è stato successivamente semplificato in modo da ottenere il seguente file:

```
total 24
drwxrwxr-x 6 quer quer 4096 Jan 31 16:38 .
drwxr-xr-x 4 quer quer 4096 Feb 10 13:30 ..
drwxrwxr-x 3 quer quer 4096 Feb 10 17:08 current
-rwxrwxr-x 1 quer quer 9037 Jan 30 23:24 pgrm
-rw-rw-rw- 1 quer quer 8881 Jan 30 23:49 pgrm.c
...
total 1116
drwxrwxr-x 3 quer quer 4096 Feb 11 17:08 .
...
```

In cui la parola chiave `total` serve esclusivamente a iniziare l'elenco di un nuovo sotto-direttorio, i campi di ciascun elenco sono separati da uno spazio singolo e il quinto campo indica lo spazio occupato dalla directory entry.

Scrivere uno script BASH in grado di ricevere il nome di un file con tale formato sulla riga di comando e di visualizzare una qualche informazione (il nome, la dimensione, tutta la riga, etc., a scelta) sui file che occupano più spazio di tutti gli altri file inclusi nel direttorio a cui essi stessi appartengono. Ad esempio, nel primo direttorio tale file è `pgrm`.

SOLUZIONE A

```
#!/bin/bash

while read line
do
  echo $line| grep "total" &>  /dev/null
    if [ $? -eq 0 ]
    then
      echo $max_name
      max=0
      max_name=""
    else
      echo $line | grep "^d" &>  /dev/null
      if [ $? -ne 0 ]
      then
        size=`echo $line | cut -d " " -f 5`
        if [ $size -gt $max ]
        then
          max=$size
          max_name=`echo $line | cut -d " " -f 9`
        fi
      fi
    fi
done < $1

echo $max_name
```

SOLUZIONE B

```
#!/bin/bash

while read line
do
  isstart=`echo $line| grep "total"| wc -c`
  if [ $isstart -ne 0 ]
  then
    echo $max_name
    max=0
    max_name=""
  else
      isdir=`echo $line | grep "^d" | wc -c`
      if [ $isdir -eq 0 ]
      then
        size=`echo $line | cut -d " " -f 5`
        if [ $size -gt $max ]
        then
          max=$size
          max_name=`echo $line | cut -d " " -f 9`
        fi
      fi
  fi
done < $1

echo $max_name
```

5. Scrivere un script AWK in grado di generare l'elenco delle citazioni bibliografiche riportate in un testo secondo le seguenti specifiche.

   Un testo contiene un numero illimitato di citazioni bibliografiche con il seguente formato:

   `(id) citazione (ID)`

   in cui una stringa qualsiasi (ma con tutti caratteri alfabetici minuscoli e racchiusa tra parentesi tonde) rappresenta l'inizio della citazione, e la stessa stringa (ma con tutti caratteri alfabetici maiuscoli) rappresenta la sua terminazione. Ciascuna citazione è in generale costituita da più parole, è inclusa interamente in un'unica riga, e può comparire più volte all'interno del testo (tutte le citazioni racchiuse dallo stesso identificatore sono identiche). Le parentesi tonde non sono utilizzare per altri scopi all'interno del file.

   Il file di ingresso così formattato va ricopiato in un file di uscita, in cui ogni citazione viene sostituita all'interno del testo con un numero intero crescente (a partire da 1) e va quindi inserita in fondo al testo stesso con formato:

   `(numero crescente) citazione`

   A numero uguale corrisponde ovviamente citazione uguale.

   Il nome del file di ingresso e del file di uscita vanno ricevuti dallo script sulla riga di comando o in alternativa assegnati quali variabili sulla riga di comando stessa.

   Il seguente esempio riporta a sinistra un possibile file di ingresso e a destra il corrispondente file di uscita.

```
...
... (soa) A. Silbershatz, ... (SOA) ...
...
(sob) W. R. Stevens, ... (SOB)
... (soa) ... (SOA) ...
...
... (sob) ... (SOB) ...
```

```
...
... (1) ....
...
(2)
... (1) ...
...
... (2) ...
BIBLIOGRAFIA
(1) A. Silbershatz, ...
(2) W. R. Stevens, ...
```

```
{
  line=""
  for(i=1;i<=NF;i++) {
    if ($i~/\([a-z]*\)/) {
      id=$i
      #print "ID= "id
      if (cit_num[id]!=0) {
        # sostituisco
        line=line" (" cit_num[id] ")"
        # elimino dal testo
        j=i+1
        while(!($j~/\([A-Z]*\)/)) {
          j++
        }
        i=j;
      } else {
        # aggiorno e sostiuisco
        num_cit++
        cit_num[id]=num_cit;
        # elimino dal testo e salvo in un vettore la citazione
        j=i+1;
        cit=""
        while(!($j~/\([A-Z]*\)/)) {
          cit=cit " " $j
          j++
        }
        i=j;
        # aggiungo cit nel vettore delle citazioni
        cit_id[id]=cit
        line=line" (" cit_num[id] ")"
      }
    } else
      line=line " " $i
  }
  print line
}

END {
  print "\nBIBLIOGRAFIA"
  for (id in cit_id) {
    print cit_num[id] cit_id[id]
  }
}
```

6. **Per i candidati iscritti al corso nell'anno accademico 2014–2015.**
Si consideri il seguente insieme di processi:

Processo	Tempo arrivo	Burst Time	Priorità
P_0	0	2	5
P_1	1	8	1
P_2	2	1	4
P_3	3	6	2
P_4	4	4	3

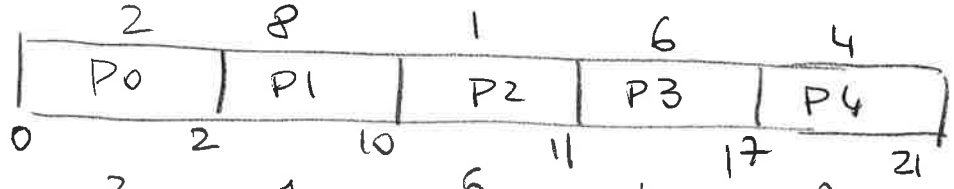
Rappresentare mediante diagramma di Gantt l'esecuzione di tali processi utilizzando gli algoritmi di scheduling "First Come First Served" senza prelazione e "Shortest Job First" senza e con prelazione.

Calcolare il tempo di completamento per ciascun processo, nonchè il tempo di attesa medio. Confrontare i tre algoritmi in base a quest'ultimo criterio, motivando il risultato.

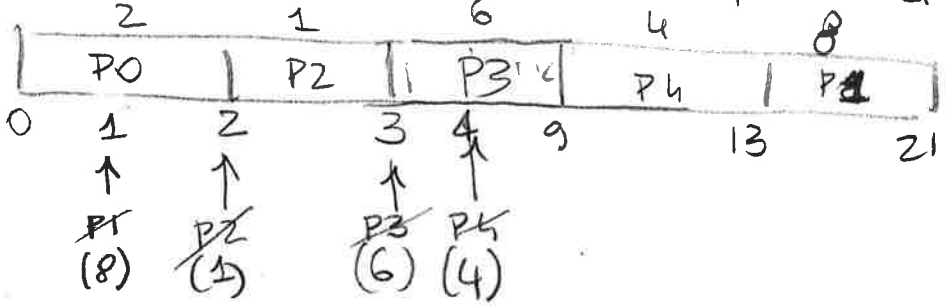
Per i candidati iscritti al corso negli anni accademici 2012–2013 o 2013–2014.
Chiarire il significato dei seguenti termini: *directory entry*, *hard-link*, *soft-link*. Riportare i comandi per creare hard- e soft-link. Riportare un esempio di gestione e di conteggio degli hard-link nel caso della creazione di un direttorio quale sotto-direttorio di un direttorio dato. Rappresentare la filosofia generale e la struttura del file-system mediante ausili grafici opportuni.

P0	0	2
P1	1	8
P2	2	1
P3	3	6
P4	4	4

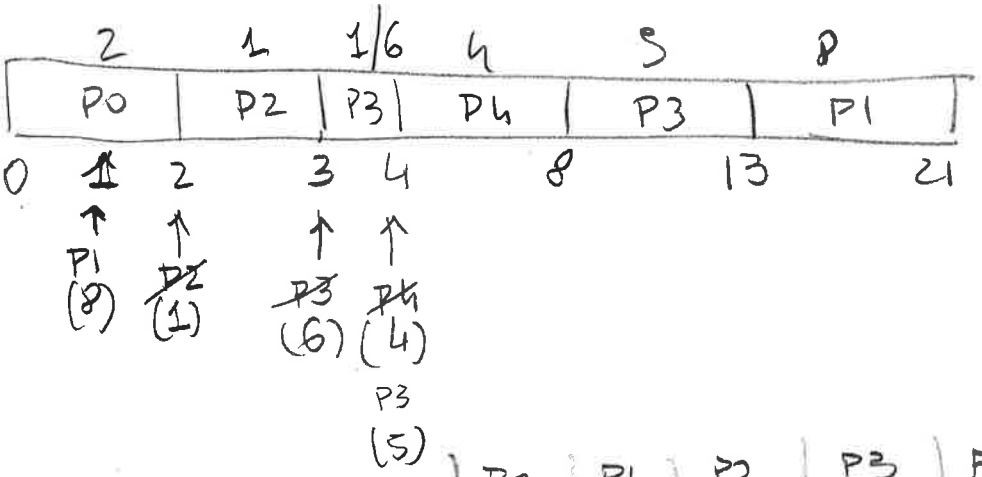
FCFS



SJF$_{nop}$



SJF$_{cp}$



tempo completamento
≙ ($t_{fine} - t_{arrivo}$)

tempo attesa
≙ $\sum t_{nella\ coda\ ready}$
= $\sum t_{attesa}$

	P0	P1	P2	P3	P4
FCFS	2-0 2	10-1 9	11-2 8	17-3 14	21-4 17
SJF$_{nop}$	2-0 2	21-1 20	3-2 1	9-3 6	13-4 9
SJF$_p$	2-0 2	21-1 20	3-2 1	13-3 10	8-4 4

	P0	P1	P2	P3	P4
FCFS	0-0 0	2-1 1	10-2 8	11-3 8	17-4 13
SJF$_{nop}$	0-0 0	13-1 12	2-2 0	3-3 0	9-4 5
SJF$_p$	0-0 0	13-1 12	2-2 0	3-3+ 8-4 4	4-4 0

tattesa medio

FCFS = 30/5 = 6
SJF$_{nop}$ = 17/5 = 3.4
SJF$_p$ = 16/5 = 3.2