Sistemi Operativi T
Prova di laboratorio
8 Giugno 2021

0. Teoria [punti 10]

A. Memoria virtuale.

B. Sincronizzazione tra processi Unix con i segnali.

C. Si consideri il seguente programma di sistema Unix:

```
#include <stdio.h>

main()
{     int p1=1,p2=0, p3=-1, k=1;
      p1 = fork();
      k+=p3;
      if (p1) p2 = fork();
      if (p2)
      {   p3=fork();
          if (!p3)
              printf("IOS %d\n",k);
          k++;
      }
      if (k>0) printf("Android %d\n",k);
}
```

In condizioni di funzionamento ideale (senza errori nelle fork, printf, ecc.), quanti processi complessivamente vengono creati? Quali sono le relazioni gerarchiche tra i processi creati? Quale output produce ogni processo?

Sistemi Operativi T
Prova di laboratorio
8 Giugno 2021

1. Esercizio di Programmazione Concorrente in Java [punti 10]

Si consideri un grande centro vaccinale per la somministrazione del vaccino anti-Covid19, nella giornata in cui si svolge un "**open day vaccinale**" per la somministrazione del vaccino J&J.
La **dotazione totale** di dosi J&J per l'Open Day è pari a **TOT_JJ**.
Nel giorno considerato, oltre all'attività straordinaria dovuta all'open day, il centro svolge anche l'attività ordinaria per la somministrazione dei vaccini prenotati; di conseguenza, gli utenti del centro vaccinale possono essere di due tipi:

- **Utenti Prenotati**, ai quali verrà somministrato il vaccino Pfizer;
- **Utenti OpenDay**, ai quali verrà inoculato il vaccino J&J (**se possibile**).

Le vaccinazioni vengono eseguite in una grande sala suddivisa in **2 Aree vaccinali**:

- **Area Prenotati**: in questa area sono disponibili N_P operatori sanitari dedicati alla vaccinazione delle persone prenotate.
- **Area OpenDay**: in questa area sono disponibili N_OD operatori sanitari dedicati alla vaccinazione delle persone che partecipano all'open day.

Ogni operatore sanitario può assistere **un utente alla volta**.

Un utente può entrare nella sala:

- **Utente Prenotato:** è necessario che vi sia un operatore libero nell'Area Prenotati;
- **Utente OpenDay**: è necessario che che ci sia almeno una dose di vaccino J&J e che ci sia un operatore libero nell'area OpenDay; nel caso in cui non sia disponibile la dose di vaccino, l'utente **non attende e va a casa**.

Una volta entrato nella sala, ogni **utente verrà assistito** per tutto il tempo di permanenza nella sala **dallo stesso operatore sanitario**.

La sala è accessibile attraverso un **corridoio** che viene utilizzato **sia per l'entrata che per l'uscita degli utent**i.
Per motivi di sicurezza il corridoio è utilizzato **a senso unico alternato**: la presenza nel corridoio di uno o più utenti in una direzione D impedisce il transito ad ogni utente nella direzione opposta a D.
Inoltre, per garantire un adeguato distanziamento, il **numero degli utenti** che possono contemporaneamente percorrere il corridoio è limitato dal valore massimo **MAXC**.

Realizzare **un'applicazione concorrente in Java basata sul monitor** nella quale **ogni utente** sia rappresentato da un **thread distinto** ed il **centro vaccinale** sia una **risorsa** condivisa dagli utenti.
La politica di gestione del centro vaccinale dovrà tenere in considerazione tutti i vincoli dati ed inoltre:

- dovrà **privilegiare gli utenti in uscita** dalla sala rispetto a quelli in entrata.
- nell'ambito di una stessa direzione (entrata o uscita dalla sala), **gli utenti prenotati dovranno avere la precedenza sugli utenti OpenDay**.

Sistemi Operativi T
Prova di laboratorio
8 Giugno 2021

2. Esercizio di programmazione di sistema con system call Linux [punti 10]

Si realizzi un programma C che, utilizzando le system call di GNU/Linux preveda la seguente interfaccia:

```
esame Fin S N
```

dove:

- `Fin` è il nome assoluto di un file di testo esistente nel file system;
- `S` è una stringa non contenente caratteri numerici
- `N` è un numero intero positivo.

Dopo avere effettuato gli opportuni controlli sui parametri in ingresso, il processo iniziale P0 creerà un figlio P1, il quale creerà a sua volta un nipote P2. Il processo P2 è, pertanto, figlio di P1 e nipote di P0.

Successivamente P0 dovrà leggere il contenuto di `Fin` e comunicare a P1 tutti i caratteri letti **ad eccezione dei caratteri numerici** (cioè quelli compresi tra '0' e '9').

Il processo P1 tramite il comando **`grep`** calcolerà il numero di linee che contengono almeno una occorrenza della stringa S nella sequenza di caratteri inviatagli da P0, comunicando il risultato del conteggio al processo P2. (suggerimento: `grep` con opzione -c)

P2 deve leggere il valore V ricevuto da P1, e confrontarlo con il valore N dato come argomento:

- se **V è maggiore di N**: P2 stamperà *"Trovate V righe contenenti la stringa S"*; successivamente P2 terminerà.
- se V è minore o uguale a N: P2 stamperà *"Le righe contenenti la stringa S sono troppo poche",* attenderà 5 secondi e cancellerà il file Fin; successivamente P2 terminerà.

P0 attenderà la terminazione di P1e, dopo averne analizzato lo stato di terminazione (stampando le relative informazioni), terminerà la sua esecuzione.