Sistemi Operativi L-A
Prof. Anna Ciampolini
Prova Scritta di Venerdi` 13 Luglio 2007

Tempo a disposizione: 3 ore

Esercizio 1 [punti 12] – Progetto UNIX
Si realizzi un programma, che, utilizzando le *system call* del sistema operativo UNIX, soddisfi le seguenti specifiche:

Sintassi di invocazione: `Esame c N1 N2`

Significato degli argomenti:

- **`Esame`** è il nome del file eseguibile associato al programma.
- **`N1 e N2`** sono interi non negativi.
- **`C`** e` una stringa che rappresenta il nome di un file eseguibile (per semplicita`, si supponga che il direttorio di appartenenza del file C sia nel PATH).

Specifiche:

- Il processo iniziale (P_0) deve creare **`2`** processi figli P_1 e P_2 . Si deve quindi ottenere una gerarchia di 3 processi: P_0 (padre), P_1 (primo figlio) e P_2 (secondo figlio).

- Ognuno dei due processi figli P1 e P2, esibisce il seguente comportamento:
  non appena creato, P1 (o P2) si pone in attesa per un intervallo di N1 secondi (o N2, nel caso di P2):
  - allo scadere del timeout, il processo figlio P1 (o P2) manda un segnale al padre e poi passa ad eseguire il comando C.
  - Se il processo figlio P1 (o P2) riceve un segnale dal padre prima della scadenza del proprio timeout, esso deve terminare trasferendo al padre il numero Ns di secondi dell'intervallo N1(o N2 nel caso di P2) effettivamente trascorsi in attesa.

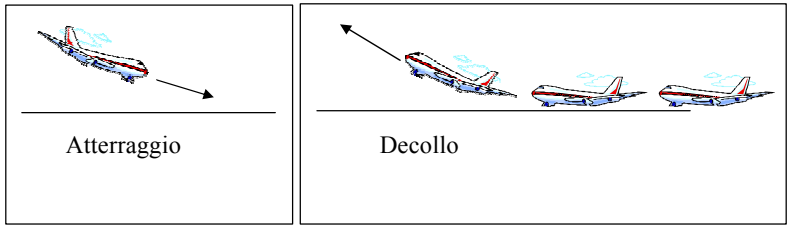
- Il **processo P_1** , dopo aver generato i figli, si pone in attesa del primo segnale proveniente da uno dei due figli; non appena riceve tale notifica:
  - Se il segnale proviene da P1, P0 deve terminare forzatamente l'esecuzione di P2 e poi attendere la terminazione di entrambi i figli, avendo cura di stampare lo stato di terminazione di ciascuno di essi.
  - Se il segnale proviene da P2, P0 deve terminare forzatamente l'esecuzione di P1 e poi attendere la terminazione di entrambi i figli, avendo cura di stampare lo stato di terminazione di ciascuno di essi.

Esercizio 2 [punti 18] – Monitor

Si consideri un piccolo Aereoporto, nel quale sia presente una sola pista, utilizzata sia per i **decolli** che per gli **atterraggi**.
Pertanto la pista puo` trovarsi in 2 stati diversi:

1. **Atterraggio**: la pista e` allocata ad un unico aereo che deve effettuare l'atterraggio.

2. **Decollo**: la pista e` allocata a **uno o piu`** aerei che sequenzialmente eseguiranno il decollo. per evitare situazioni di eccessivo traffico, in questo stato la pista non puo` accogliere piu` di MAX aerei.



Si vuole realizzare una politica di sincronizzazione degli accessi alla pista che tenga conto delle specifiche date ed inoltre del seguente vincolo di priorita`:

- nell'accesso alla pista venga data la precedenza assoluta agli aerei in fase di atterraggio.

Realizzare un'applicazione concorrente (a scelta in Java o in C/pthread) che, rappresentando ogni aereo mediante un thread, e utilizzando il concetto di monitor e le variabili condizione implementi la politica di sincronizzazione data.