PCAD a.a. 2018/19 - Scritto 18 luglio 2019

L'esame è composto da domande a risposta multipla ed esercizi a risposta libera.
Se richiesto dal testo o se avete dubbi sulla formulazione di una domanda aggiungete una breve spiegazione per giustificare la risposta. Nella stessa domanda ci può essere più di un'affermazione vera. Occorre raggiungere almeno 15 punti per poter far media con il voto della discussione del progetto.

D1 (1 punto) Un processo nei sistemi operativi:
1. è un'unità di allocazione sia di risorse che di esecuzione
2. può condividere lo stack con altri processi
3. può essere usato per eseguire una parte di un calcolo parallelo
4. può essere eseguito in modalità user e modalità kernel

D2 (1 punto) Un context switch si può verificare
1. quando un processo ritorna da una chiamata di sistema
2. quando un processo ritorna da una chiamata di funzione
3. quando un processo si sospende su un'operazione di I/O
4. Quando un processo esegue una routine di gestione di interrupt

D3 (1 punto) Un variabile **condition**
1. si può usare in qualsiasi parte del codice
2. ha un'operazione di **signal** che è equivalente alla **up** dei semafori
3. è un flag di tipo Bool dichiarato **volatile**
4. garantisce **write atomicity** nei programmi dove viene usata

D4 (1 punto) Nel debugger di Eclipse si usano i trigger condizionali
1. per controllare condizioni su uno specifico thread
2. per forzare la generazione di context switch
3. per attivare routine di interrupt
4. per stampare messaggi di profiling utili allo sviluppatore

D5 (1 punto) Un Executor
1. è un'implementazione di un monitor fornita da Java
2. viene usato per garantire la mutua esclusione tra thread con variabili condivise
3. viene usato per evitare deadlock
4. viene usato come costruttore di thread pool

D6 (1 punto) Una BlockingQueue in Java
1. ha metodi thread safe
2. è una struttura dati sincronizzata
3. può essere usata per implementare un thread pool
4. non può essere usata come campo di una struttura dati sincronizzata

D7 (2 punti) Lo schema di codice in C

if (fork() != 0) { … } else { execve(command, parameters, NULL); }

1. viene usato per creare un thread Unix/Linux
2. si può usare per assegnare un task ad un thread in Unix/Linux
3. si può usare per eseguire in parallelo due parti di codice
4. si può usare come blocco base per usare in maniera efficiente un'architetture multicore

D8 (2 punti) Il programma concorrente in pseudo codice

Shared Memory: semaphor s1=0; s2=0;
Thread T1: { down(s1); down(s2); print("a"); }
Thread T2: { down(s2); down(s1); print("b"); }

1. Quando eseguito stampa sempre le lettere a e b
2. " " stampa solo una delle due lettere tra a e b
3. " " stampa sempre la sequenza ab
4. " " non stampa mai la sequenza ab

D9 (2 punti) Nella libreria RMI una callback è definita come

1. una chiamata di metodo di un client effettuata tramite skeleton/stub
2. una chiamata di metodo di un client eseguita ad ogni connessione con il server
3. un thread associato ad un long running task nel server
4. una chiamata asincrona nel server

D10 (4 punti) Considerare il seguente codice ispirato alla sintassi di Java:

```
class mydata{
   private a=0;
   public synchronized m1(){  if (a = = 0) { wait(); } }
   public synchronized m2(){  a=1; }
   public synchronized m3(){ if (a = = 1) notify();  }
}
```

Main: mydata A = new mydata();
Thread 1 (creato da Main con reference ad A): A.m1();
Thread 2 (creato da Main con reference ad A): A.m2(); A.m3();

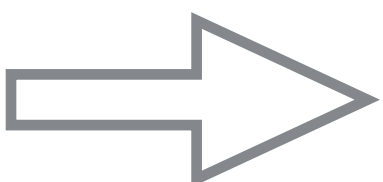
1. Il programma può eseguire le chiamate di m1, m2 e m3 in tutti gli ordini possibili? (motivare la risposta)

2. Il programma può andare in deadlock? (motivare la risposta)

3. Il programma può andare in deadlock se il corpo di Thread 1 viene eseguito 2 volte? (motivare la risposta)

4. Il programma può andare in deadlock se il corpo di Thread 2 viene eseguito 2 volte? (motivare la risposta)

Esercizio 1 (8 punti)
Scrivere una possibile soluzione (cioè un programma concorrente nel linguaggio che preferite) al problema dei lettori e scrittori con priorità ai lettori rispetto agli scrittori (priorità nel caso un altro lettore sia già in sezione critica).



Esercizio 2 (8 punti)
Spiegare brevemente quali sono le difficoltà principali della programmazione distribuita descrivendo almeno un esempio, tra quelli visti a lezione, di programma distribuito basato su codice multithreaded.