— COMPITO CON SOLUZIONE —

Sistemi Operativi (modulo II / B)

3 Giugno 2014

Esercizio 1

Si considerino i seguenti processi:

```
process P1 {
    P(S1);
    P(S2);
    P(S3);
    < usa risorse >
    V(S1);
    V(S2);
    V(S3);
}
```

```
process P2 {
    P(S1);
    P(S3);
    P(S2);
    < usa risorse >
    V(S1);
    V(S3);
    V(S2);
}
```

```
process P3 {
    P(S2);
    P(S1);
    P(S3);
    < usa risorse >
    V(S2);
    V(S1);
    V(S3);
}
```

I semafori S1, S2 e S3 sono inizializzati a 1 (verde). È possibile che i processi vadano in stallo? Discutere mostrando eventuali sequenze di esecuzione che portano a stalli. Modificare il codice in modo da prevenire ogni situazione di stallo.

SOLUZIONE: Poiché P1 e P2 fanno entrambi una P su S1 solo uno dei due procederà mentre l'altro processo rimarrà in attesa sul semaforo S1. Tra P1 e P2 non si può quindi formare una situazione di attesa circolare. Supponiamo che P1 vada in esecuzione assieme a P3. P1 esegue P(S1) mentre P3 esegue P(S2). Entrambi i semafori diventano rossi. A questo punto quando P1 esegue P(S2) e P3 esegue P(S1) si forma una situazione di attesa circolare. P1 e P3 sono in stallo. Se P2 va in esecuzione anch'esso andrà in stallo sul semaforo S1. Una situazione analoga si ha se partono P2 e P3.
La soluzione generale per prevenire lo stallo tra ispirazione dall'allocazione gerarchica: I processi fanno le P sui semafori nello stesso ordine: P(S1), P(S2), P(S3). In questo caso specifico è sufficiente invertire P(S2) e P(S1) nel processo P3 in modo da mettere tutti i processi in attesa su S1 ed evitare ogni attesa circolare. Di fatto S1 diventa un mutex di sezione critica.

Esercizio 2

Progettare un Monitor `Sc` che permetta a quattro thread di scrivere in modo sincronizzato la parola "CIAO". Ognuno dei thread scrive una lettera attendendo che le lettere precedenti siano state scritte.

```
thread C {
  Sc.attendi("");
  print "C"
  Sc.notifica("C")
}
```

```
thread I {
  Sc.attendi("C");
  print "I"
  Sc.notifica("I")
}
```

```
thread C {
  Sc.attendi("CI");
  print "A"
  Sc.notifica("A")
}
```

```
thread C {
  Sc.attendi("CIA");
  print "O"
  Sc.notifica("O")
}
```

Spiegare il funzionamento della soluzione proposta.

SOLUZIONE: Utilizziamo una condizione implicita come in java. Inoltre usiamo una variabile di tipo String per memorizzare la stringa già stampata. il metodo attendi mette i thread in attesa finché non è stata stampata la stringa passata per parametro. Il metodo notifica aggiunge la lettera notificata alla stringa stampata e notifica tutti i thread in modo che possano ricontrollare la condizione di bloccaggio (usando un'unica condizione è necessario svegliare tutti i thread).

```
monitor Sc {
    String stampata=""; // tiene traccia della stringa stampata

    attendi(String s) {
        // attende che sia stata stampata s
        while(stampata != s)
            wait();

    }
```

```
    notifica(String s) {
        // aggiunge s alla stringa stampata
        stampata += s;
        // notifica tutti i thread
        notifyAll();
    }
```