

# SISTEMI OPERATIVI IIN/ SISTEMI DI ELABORAZIONE

## prova scritta preliminare del 04.09.2003

Nome: \_\_\_\_\_

Cognome: \_\_\_\_\_

In un sistema sono presenti due variabili di tipo `Pila` (`P0` e `P1`), che realizzano due pile a capacità illimitata; su queste strutture si opera con le seguenti funzioni:

- `void deposita( Pila p, Dato d )`
- `Dato preleva( Pila p )`

Sono altresì previste tre tipologie di processi:

- processi di tipo `G`, che producono singole unità di informazione e le depositano sulla pila `P0`;
- processi di tipo `E`, che prelevano singole unità di informazione dalla pila `P0`, le trasformano e le depositano sulla pila `P1`;
- processi di tipo `C`, che prelevano dalla pila `P1` `K` unità di informazione alla volta e le consumano.

In particolare, sul sistema sono attivi un processo di tipo `G` (`G1`), `M` processi di tipo `E` (`E1...EM`) ed `N` processi di tipo `C` (`C1...CN`).

Il sistema fornisce, per risolvere i problemi di sincronizzazione, un tipo di dato `Semaforo` che realizza dei semafori con contatore che possono essere controllati mediante le seguenti primitive:

- `wait( Semaforo s, unsigned int v )`: se il valore del contatore associato al semaforo `s` è inferiore al valore `v` specificato il processo chiamante viene sospeso finché il valore del contatore non risulta essere maggiore o uguale al valore specificato; quando il valore del contatore risulta essere non inferiore al valore `v` specificato il contatore viene decrementato di `v` e il processo procede nell'esecuzione;
- `signal( Semaforo s, unsigned int v )`: il contatore associato al semaforo `s` viene incrementato del valore `v`.

Data la traccia riportata di seguito, si sviluppi una soluzione per la sincronizzazione dei processi in esecuzione sul sistema.

<b>G</b>	<b>E</b>	<b>C</b>
<pre>while( true ) { ... dato = genera(); ... deposita( P0, dato ); ... }</pre>	<pre>while( true ) { ... dato = preleva( P0 ); ... nuovo = elabora( dato ); ... deposita( P1, nuovo ); ... }</pre>	<pre>while( true ) { ... for( i=0; i&lt;K; i++) {     dati[i] = preleva( P1 ); } ... consuma( dati ); ... }</pre>

## Soluzione

### Variabili condivise e semafori

```
Semaforo mutex0: inizializzato a 1;  
Semaforo mutex1: inizializzato a 1;  
Semaforo pieno0: inizializzato a 0;  
Semaforo pieno1: inizializzato a 0;
```

#### G:

```
while( true )  
{  
    wait( mutex0, 1 );  
    dato = genera();  
    deposita( P0, dato );  
    signal( pieno0, 1 );  
    signal( mutex0, 1 );  
}
```

#### E:

```
while( true )  
{  
    wait( pieno0, 1 );  
    wait( mutex0, 1 );  
    dato = preleva( P0 );  
    signal( mutex0, 1 );  
    nuovo = elabora( dato );  
    wait( mutex1, 1 );  
    deposita( P1, nuovo );  
    signal( pieno1, 1 );  
    signal( mutex1, 1 );  
}
```

#### C:

```
while( true )  
{  
    wait( pieno1, K );  
    wait( mutex1, 1 );  
    for( i=0; i < K; i++)  
    {  
        dati[ i ] = preleva( P1 );  
    }  
    signal( mutex1, 1 );  
    consuma( dati );  
}
```