Per sincronizzare i due processi **Produttore e Consumatore**

Attraverso buffer singolo

Variabile condivisa: **Condi**

|  |  |  |
| --- | --- | --- |
| Variabile condivisa: **Condi**  Variabile semaphore: syncro=1 | | |
| Produttore  {  Var BufferP;  While(true){  Generadati(BufferP);  P(syncro);  **Condi**=BufferP;  V(syncro);  }  } |  | Consumatore  {  Var BufferC;  while(true){  P(syncro);  BufferC=**Condi**;  V(syncro);  Consumadati(BufferC);  }  } |

Cosa otteniamo?

* Mutex?
* Sincronizzazione?

Risposta:

* Mutex? SI
* Sincronizzazione? NO

**Per sincronizzare i due processi**

I due semafori devono essere inizializzati nel seguente modo

* C: 0
* P: 1

|  |  |  |
| --- | --- | --- |
| Variabile condivisa: **Condi** | | |
| Produttore  {  Var BufferP;  While(true){  Generadati(BufferP)  P(P);  **Condi**=BufferP;  V(C);  }  } |  | Consumatore  {  Var BufferC;  while(true){  P(C);  BufferC=**Condi**;  V(P);  Consumadati(BufferC);  }  } |
| Variabile condivisa: **Condi**  Semaforo binario P=1  Semaforo binario C=0 | | |
| Produttore  {  Var BufferP;  While(true){  Generadati(BufferP);  P(C);  **Condi**=BufferP;  V(P);  }  } |  | Consumatore  {  Var BufferC;  while(true){  P(P);  BufferC=**Condi**;  V(C);  Consumadati(BufferC);  }  } |

**Produttore-Consumatore a tre stadi**

|  |  |  |
| --- | --- | --- |
| Variabile condivisa: **Condi**  Semaforo binario PP=1  Semaforo binario P=0  Semaforo binario C=0 | | |
| PProduttore  {  Var BufferPP;  While(true){  Generadati(BufferPP);  P(PP);  **Condi**=BufferPP;  V(P);  }  } | Produttore  {  Var BufferP;  While(true){  P(P);  BufferP=**Condi**;  Elaboradati(BufferP);  **Condi**=BufferP;  V(C);  }  } | Consumatore  {  Var BufferC;  while(true){  P(C);  BufferC=**Condi**;  V(PP);  Consumadati(BufferC);  }  } |

**Produttori molteplici-Consumatore**

|  |  |  |  |
| --- | --- | --- | --- |
| Variabile condivisa: **Condi**  Semaforo binario P=1  Semaforo binario P=0  Semaforo binario C=0 | | | |
| Produttore1  {  Var BufferPP;  While(true){  Generadati(BufferPP);  P(P);  **Condi**=BufferPP;  V(C);  }  } | ….. | ProduttoreN  {  Var BufferPP;  While(true){  Generadati(BufferPP);  P(P);  **Condi**=BufferPP;  V(C);  }  } | Consumatore  {  Var BufferC;  while(true){  P(C);  BufferC=**Condi**;  V(P);  Consumadati(BufferC);  }  } |

**Produttore-Consumatore a buffer multiplo**

|  |  |  |
| --- | --- | --- |
| Variabile condivisa: int **buffer[MAX];**  Semaforo piene=0;  Semaforo vuote=MAX;  Semaforo binario mutex=1; | | |
| int inserisci=0; // uso locale  Produttore {    while(true) {      Generadati();      P(vuote); // richiede una cella vuota      P(mutex);      buffer[inserisci] = d;      inserisci = (inserisci+1) % MAX;      V(mutex);      V(piene); // rilascia una cella piena    }  } |  | int preleva=0; //uso locale  Consumatore() {    while(true) {      P(piene); // richiede una cella piena      P(mutex);      d = buffer[preleva];      preleva = (preleva+1) % MAX;      V(mutex);      V(vuote); // rilascia una cella vuota      Consumadati();    }  } |

**Proposta di esercizio**

**Variante del produttore-consumatore in cui ci sono N produttori che generano N dati che verranno usati tutti assieme dal consumatore**

**Esercizio lettori/scrittore:** possiamo avere un solo scrittore oppure più lettori, ma non lettore/i e scrittore contemporaneamente.

|  |  |  |
| --- | --- | --- |
| int num\_lettori=0  Semaforo binario w=0 //semaforo di scrittura  Semaforo binario mutex=0 | | |
| Scrittore  {  While(true){  P(w);  **scrivi()**;  V(w);  }  } | ….. | ProduttoreN  {  Var BufferP;  While(true){  P(mutex);       num\_lettori++;       if (num\_lettori == 1)  // primo lettore         P(w);        // acquisisce la mutex in scrittura       V(mutex);  **leggi();**       P(mutex);              // protegge num\_lettori       num\_lettori--;       if (num\_lettori == 0)  // ultimo lettore         V(w);        // rilascia la mutua esclusione in scrittura       V(mutex);  }  } |