CAP. 6 : SINCRONIZZAZIONE TRA PROCESSI

Qualche concetto di base

Processi cooperanti ***condividono uno spazio logico di indirizzi***

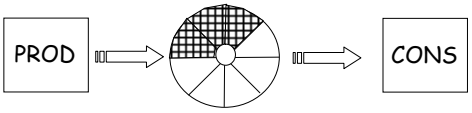
⇩

Bisogna ***in qualche modo*** garantire consistenza di dati...

... e questo richiede meccanismi che assicurino ***l'esecuzione in ordine appropriato*** dei processi cooperanti (***IPC***)

Di nuovo su PROD-CONS

La prima soluzione vista del problema produttore-consumatore permetteva al piu' ***n-1 elementi nel buffer*** alla volta



Proviamo allora a trovare una soluzione che ne permetta ***n*** : variabile ***counter*** che conta il numero di elementi nel buffer

Una nuova soluzione

Variabili condivise:

```
type item = … ;
var buffer array [0..n-1] of item;
in, out : 0..n-1 ;
counter : 0..n ;
in, out, counter := 0;
```

Processo produttore:

```
repeat
    …
  produci un elemento in nextp
    …
  while counter = n do no-op;
  buffer [in] := nextp;
  in := in + 1 mod n;
  counter := counter +1;
until false;
```


Processo consumatore:

```
repeat
  while counter = 0 do no-op;
  nextc := buffer [out];
  out := out + 1 mod n ;
  counter := counter – 1;
    …
  consuma l'elemento presente in nextc
    …
until false ;
```

La ***condivisione*** vera e propria ***si sposta sulla variabile counter*** : i due processi possono aggiornare la variabile ***counter*** "allo stesso tempo" senza garanzia di consistenza...

Esempio di sequenza "dannosa" di istruzioni

Condizione per la consistenza dei dati

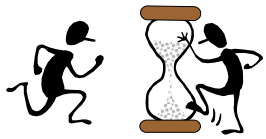
Le istruzioni

```
counter := counter + 1;
counter := counter - 1;
```

devono essere eseguite ***in maniera atomica***

E comunque …

Qualunque soluzione per la sincronizzazione tra processi deve ***prescindere*** dalla ***velocita' relativa*** di esecuzione ***dei processi***



Cos'e' una *Sezione Critica*

Ogni zona di codice in cui ***si accede ad una variabile condivisa*** viene detta ***sezione critica***

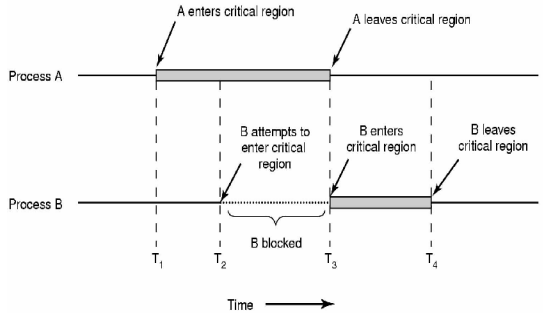
⇩

Bisogna ***far qualcosa*** prima e dopo la sezione critica per ***garantire*** il ***corretto funzionamento***

repeat
entry section
sezione critica
exit section
sezione "normale"
until *false* ;

Tutto cio' che non e' la sezione critica

A e B accedono alle loro rispettive *Sezioni Critiche*



Condizioni per il *corretto funzionamento*

1. **Mutua esclusione**:
   Se un processo sta eseguendo la sua sezione critica, allora ***nessun altro processo*** puo' stare ad ***eseguire la sua sezione critica***

Condizioni per il *corretto funzionamento*

2. **Progresso**:
Se nessun processo sta eseguendo la sua sezione critica e c'e' qualche processo che vuole entrare nella sua, allora la ***decisione su quale processo far entrare*** nella sezione critica deve essere ***presa in un tempo finito*** e ***solo dai processi che non stanno nella loro sezione "normale"***

Condizioni per il *corretto funzionamento*

3. **Attesa limitata**: Dopo che un processo ha fatto richiesta di ingresso nella sua sezione critica e prima che la sua richiesta venga soddisfatta deve essere ***limitato il numero di volte*** che ad ***altri processi*** e' consentito ***entrare nelle loro sezioni critiche***

Caso di due processi : *Alternanza*

Variabili condivise :
var *turn* : (0..1);

inizialmente *turn* = 0
turn = *i* ⇒ *Pi* puo' entrare nella sua sezione critica

Processo *Pi* :
repeat
while *turn* ≠ *i* **do** *no-op* ;
sezione critica
turn := *j* ;
sezione "normale"
until *false* ;

Valutazione di *Alternanza*

Sono garantite :
- ***mutua esclusione*** (1)
- ***attesa limitata*** (3)

Non e' garantito progresso (2)

Un processo che sta nella sua sezione "normale" puo' condizionare un altro all'ingresso nella sezione critica

Esempio : se P1 vuole entare due volte di seguito non puo', condizionato dal fatto che P2 sta ancora nella sua sezione "normale"

Piu' conoscenza reciproca : *Segnalazioni*

Variabili condivise :
var *flag* : **array** [0..1] **of** *boolean* ;

inizialmente *flag* [0] = *flag* [1] = *false*
flag [*i*] = *true* ⇒ *Pi* e' pronto ad entrare nella sua sezione critica

Processo *Pi* :
repeat
flag[*i*] := *true* ;
while *flag*[*j*] **do** *no-op* ;
sezione critica
flag [*i*] := *false* ;
sezione "normale"
until *false* ;

Valutazione di *Segnalazioni*

Sono garantite :
- ***mutua esclusione (1)***
- ***attesa limitata (3)***

Comunque non e' garantito progresso (2)!!!

La decisione su chi deve entrare nella propria sezione critica potrebbe non essere presa mai

- ***Esempio di sequenza "dannosa"***
- ***Cosa succede se si invertono le due istruzioni della entry section?!***

Soluzione ottimale : *Passo*

repeat
flag [*i*] := *true* ;
turn := *j* ;
while (*flag* [*j*] and *turn* = *j*) **do** *no-op* ;

sezione critica

flag [*i*] := *false* ;

sezione "normale"

until *false* ;