Tema 4. Continguts

laSalle ENG
Universitat Ramon Llull

1. El problema de l'exclusió mútua i el dead-lock
2. Solucions hardware
3. Solucions software
4. Comunicació entre processos.
    - 4.1 Pipes
    - 4.2 Pas de missatges
    - 4.3 Sockets
5. Eines d'Exclusió Mútua i Sincronització
    - 5.1 Semàfors
    - **5.2 Monitors**

5.1. Semàfors

laSalle ENG
Universitat Ramon Llull

A) Exclusió mútua

```
var
   buffer:array[0..MAX-1] de producte
   compt: enter
fivar
compt:=0
```

```
procés Productor()
var
  esc:enter
  C1:producte
fivar
   esc:=0
   mentre CERT fer
     C1:=produir()
     mentre compt=MAX fimentre
     buffer[esc]:=C1
     esc:=(esc+1) mod MAX
     compt:=compt+1
   fimentre
fiprocés
```

```
procés Consumidor()
var
  lec:enter
  C2:producte
fivar
   lec:=0
   mentre CERT fer
     mentre compt=0 fimentre
     C2:=buffer[lec]
     lec:=(lec+1) mod MAX
     compt:=compt-1
   fimentre
fiprocés
```

5.1. Semàfors

laSalle ENG
Universitat Ramon Llull

```
var
   buffer:array[0..MAX-1] de producte
   compt: enter
   mutex:semàfor
fivar
compt:=0
Init(mutex,1)
```

```
procés Productor()
var
  esc:enter
  C1:producte
fivar
   esc:=0
   mentre CERT fer
     C1:=produir()
     mentre compt=MAX fimentre
     buffer[esc]:=C1
     esc:=(esc+1) mod MAX
     wait(mutex)
     compt:=compt+1
     signal(mutex)
   fimentre
Fiprocés
```

```
procés Consumidor()
var
  lec:enter
  C2:producte
fivar
   lec:=0
   mentre CERT fer
     mentre compt=0 fimentre
     C2:=buffer[lec]
     lec:=(lec+1) mod MAX
     wait(mutex)
     compt:=compt-1
     signal(mutex)
   fimentre
Fiprocés
```

5.1. Semàfors

laSalle ENG
Universitat Ramon Llull

```
const
 MAX = 50
ficonst
var
 buff: array[0..MAX-1] de element
 lec: enter                                   /* posició lectura */
 esc:enter                                    /* posició escriptura */
 mutex1, mutex2, buit, ple: semàfor
fivar
init (mutex1, 1);   /* exclusió mútua */
init (mutex2, 1);   /* exclusió mútua */
init (buit, MAX);  /* número de posicions lliures */
init (ple, 0);     /* número d'elements en el buffer */
lec = 0; esc = 0;
```

```
procés productor()
mentre CERT fer
  c1 := produir();
  wait(buit);
  wait(mutex1);
  buff[esc] := c1;
  esc := (esc+1) mod MAX;
  signal(mutex1);
  signal(ple);
fimentre
fiprocés
```

```
procés consumidor()
mentre CERT fer
  wait(ple);
  wait(mutex2);
  c2 := buff[lec];
  lec := (lec+1) mod MAX;
  signal(mutex2);
  signal(buit);
fimentre
fiprocés
```

5.2. Monitors

laSalle ENG
Universitat Ramon Llull

Monitor = TAD + exclusió mútua + variables de condició.

```
<nom_monitor> = monitor
    /* declaració de variables */
    /* codi inicialització */
    proc P1(...)
         ⋮
    proc P2(...){
         ⋮
fimonitor
```

- Només hi haurà un procés actiu dins del monitor. Com que el monitor és exclusiu, no es tallarà l'execució a mig P1() o a mig P2().
- Les variables són privades al monitor i només són accessibles des de dins els procediments del monitor (per modificar-les).

5.2. Monitors

Hi ha variables de condició: bloquejar / desbloquejar processos.
Per exemple:

```
var x, y: condicio fivar
```

Funcionen de manera similar als semàfor binaris:

x.wait()→el procés que l'executa es bloqueja fins rebre un x.signal()
x.signal()→desperta un únic procés bloquejat per la variable x.wait()

Si no hi ha res bloquejat es PERD el signal (no es memoritza).

```
Buffer = monitor
         a: array[0..MAX-1] de element
         ple, buit: condicio
         lec, esc, compt: enter

    lec:=0
    esc:=0
    compt:=0

    proc PosaElement(e:element)
         si compt=MAX llavors ple.wait()    fisi
         a[esc]=e
         esc:=(esc+1) mod MAX
         compt:=compt+1
         si compt=1 llavors buit.signal() fisi
    fiproc

    funcio TreuElement() retorna element
    var
        e:element
    fivar
        si compt=0 llavors buit.wait() fisi
        e:=a[lec]
        lec:=(lec+1) mod MAX
         compt:=compt-1
         si compt=MAX-1 llavors ple.signal() fisi
        retorna e
    fifuncio

fimonitor
```

```
B:Buffer
proces Productor()
var
    C1:element
fivar
    mentre CERT fer
        C1:=Produeix()
        B.PosaElement(C1)
    fimentre
fiproces

proces Consumidor()
var
    C2:element
fivar
    mentre CERT fer
        C2:=B.TreuElement(C1)
    fimentre
fiproces
```

5.2. Monitors: advanced browns

```
proc P1(...){
      ⋮
  x.wait();
      ⋮  R1
}
proc P2(...){
      ⋮
  x.signal();
      ⋮  R2
}
```

Si cada procés és exclusiu, primer executem P1 → fa el wait i s'espera, després s'executa P2 → i quan fa el signal, es continua executant R2 i a més R1 perquè el signal ha despertat el wait.

5.2. Monitors: advanced browns