Exercícios

Uma ponte com uma única faixa mas com tráfego nos dois sentidos

Quando um carro chega a uma entrada:

-- só pode entrar se uma das condições for satisfeita:

- A ponte está desocupada

ou

- O tráfego corrente desloca-se no mesmo sentido que o do carro que acabou de chegar

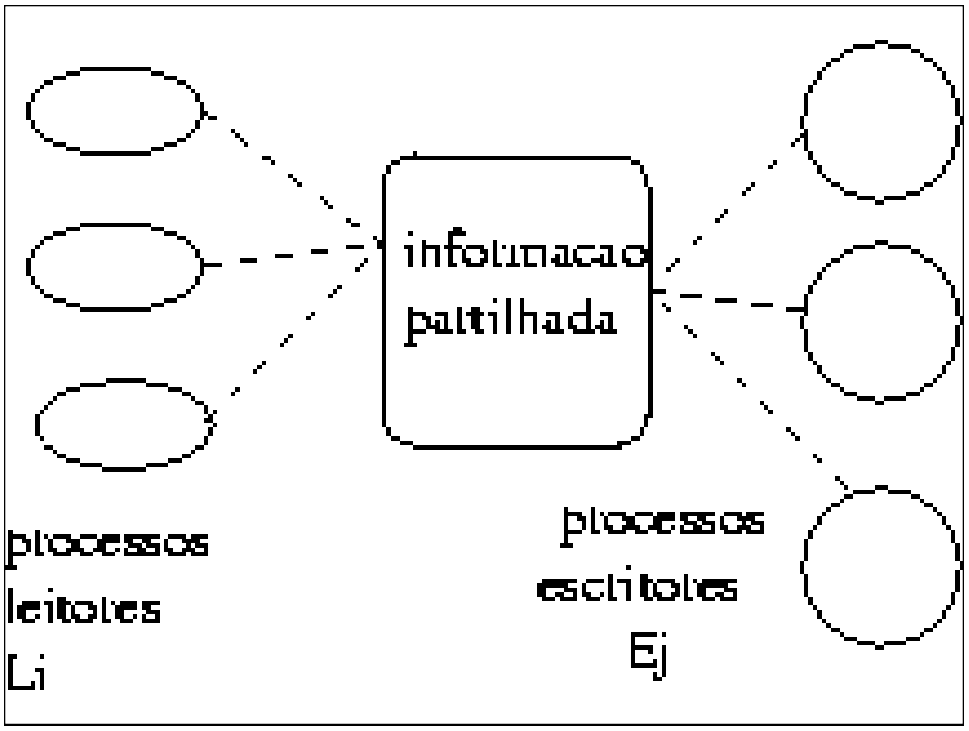
```
E:  P(exE);                 D:  P(exD);
    contE++;                    contD++;
    if (contE=1)                if (contD=1)
       then P(block);              then P(block);
    V(exE);                     V(exD);
    na ponte();                 na ponte();
    P(exE);                     P(exD);
    contE--;                    contD--;
    if (contE=0)                if (contD=0)
      then V(block);              then V(block);
    V(exE);                     V(exD);
```

Solução baseada num Monitor

```
ponte: monitor;
{ var contE, contD: int;
      blockE, blockD: condition;
 Procedure entry InE; {…}
 Procedure entry outE; {…}
 Procedure entry InD; {…}
 Procedure entry OutD; {…}
 { contE := contD := 0;} /*inicialização*/
}
```

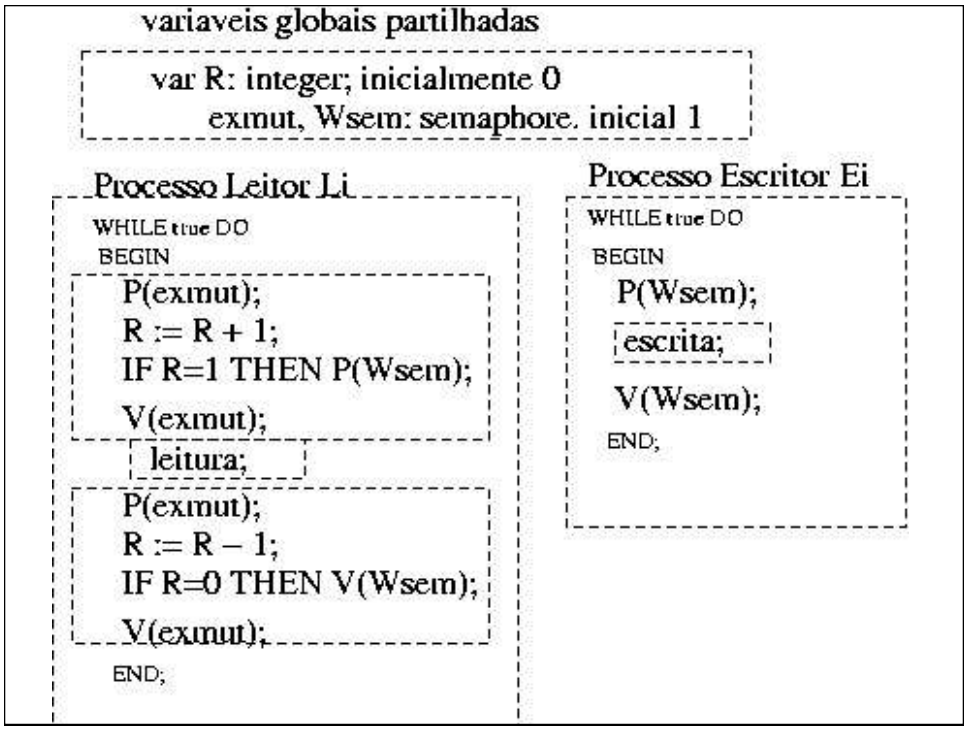
```
Carro entrando                Carro entrando
pelo lado esquerdo            pelo lado direito

 InE;                          InD;
 na ponte();                   na ponte();
 OutE;                         OutD;
```

```
InE:                          InD:
 { if (contD>0)                { if (contE>0)
    then WAIT(blockE);            then WAIT(blockD);
  contE++;                      contD++;
  SIGNAL(blockE);               SIGNAL(blockD);
 }                             }
OutE:                         OutD:
{ contE--;                    { contD--;
  if (contE=0)                  if (contD=0)
  then SIGNAL(blockD);          then SIGNAL(blockE);
}                              }
```



Leitor:
  Pede-para-Ler(); (1)
    Leitura;
  Termina-de-Ler(); (2)
Escritor:
  Pede-para-Escrever(); (3)
    Escrita;
  Termina-de-Escrever(); (4)

- Como programar as acções 1, 2, 3 e 4?



- Solução baseada num Monitor

```
Leitores_Escritores: monitor;
{ var R: int; W: boolean;
        OK_to_R, OK_to_W: condition;
 Procedure entry Start_R; {…}
 Procedure entry End_R; {…}
 Procedure entry Start_W; {…}
 Procedure entry End_W; {…}
 { R :=0; W := false;} /*inicialização*/
}
```

Um Leitor	Um Escritor
Start_R;	Start_W;
lendo();	escrevendo();
End_R;	End_W;

```
Start_R:
 { if (W or non_empty(OK_to_W))
                    then WAIT(OK_to_R);
   R := R + 1;
  SIGNAL(OK_to_R);
 }
End_R:
{ R := R – 1;
  if (R = 0) then SIGNAL(OK_to_W);
}
```

```
Start_W:
 { if ((R > 0) or W)
                then WAIT(OK_to_W);
   W := true;
 }
End_W:
{ W := false;
  if (non_empty(OK_to_R))
      then SIGNAL(OK_to_R);
      else  SIGNAL(OK_to_W);
}
```

Synchronisation Barrier

- N processes synchronise in a collective ´´rendez-vous´´:

Each process:

```
   ….
  arrival()   /* waits for all the others*/
  ….          /* proceeds, after all have arrived*/
```

N-Barrier: based on semaphores

```
var count: integer;   /* inicialmente 0*/
      exmut: semaphore; /* inicial/ 1 */
      block: semaphore; /*inicial/ 0*/
arrival()
{  P(exmut); count++;
 if (count<N) then {V(exmut); P(block); P(exmut)};
 count--; if (count > 0) then V(block);
 V(exmut);
}
-- verify if it is a correct implementation
```

a) Barreira com “Time-out”

int arrival(int N, int Tempo);

N – número de processos

Tempo – segundos

Retorna logo que:

a)--- N processos tenham invocado ‘barreira’

OU

b) --- ‘Tempo’ expirar

E no retorno deve ser indicado (0) se ocorreu a) ou (1) se ocorreu b)

No programa de cada processo, inicializações:

…

signal(SIGALRM, Tratar-alarme);

…

No programa, declaração do handler:

Tratar-alarme()

{ … }

```
int P-com-TIME-OUT(Block, Tempo):
{    alarm(Tempo);
     ret = P(Block);
     if (ret > 0) { alarm(0);
                … …
          return (0);
      }
     else {
        if ((ret ==-1) & (errno ==EINTR))
             { return (1);}
       else {… erro… return ( -1);}
     }
}
```

- Alterar a função arrival() de modo a utilizar a função P-com-TIME-OUT()

b) Implemente cancelar_barreira() que força o retorno imediato de todos os processos que tenham invocado 'barreira' até ao momento.

? Uma solução:

- Antes que um processo se bloqueie, registar o seu PID numa tabela.
- Para cancelar(): enviar um sinal, por exemplo KILL(SIGUSR1, PID) a cada um dos processos na tabela

Cada processo define um “handler”
no seu programa, inicializações:
…
signal(SIGUSR1, Tratar);
…
No programa, declaração do handler:

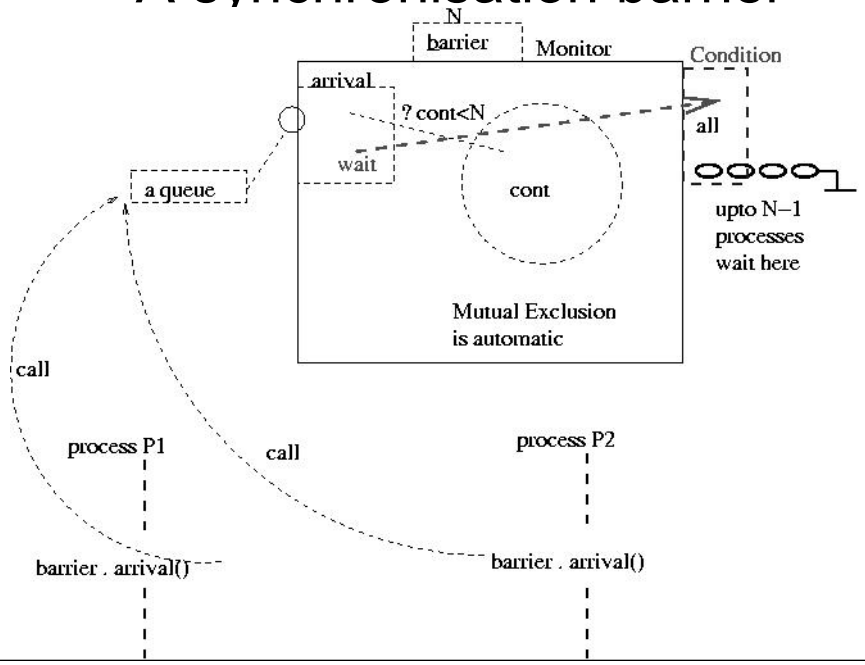
Tratar()
{ …o processo deve decrementar o count…
e retornar da função arrival() }

-- Outra solução:
Para cancelar() fazer
P(exmut);
if (count > 0) then V(block);
V(exmut);

c) Implemente, com base em filas de mensagens (nao pode usar semáforos)

-- ? Soluções ?

A synchronisation barrier



```
barrier: monitor;
 var cont: integer;
       all: condition;
 procedure entry arrival;
 { cont := cont + 1;
   if cont < N then WAIT(all);
   cont := cont – 1;
   SIGNAL(all);
 }
{ cont := 0;
}
```

Monitor de uma Barreira com TIME-OUT?