Sincronización de Bajo Nivel

Espera Activa

Introducción

Sincronización Condicional vs Exclusión Mutua

(Pasa tú) (Paso yo)

Asumimos:

- Lectura y escritura atómicas.
- Planificador justo

Sincronización Condicional

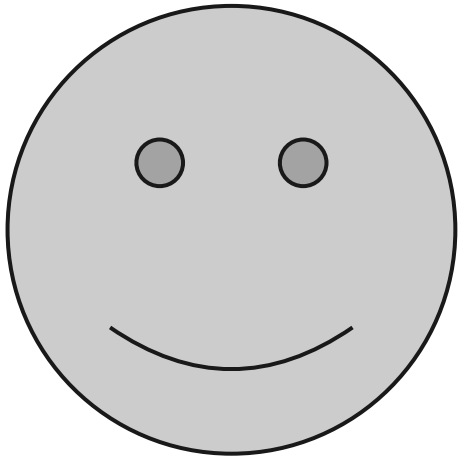
Justicia Débil: Petición Continua

```
PROGRAM PeticionContinua;
 PROCESS Servidor(VAR Peticion : BOOLEAN; VAR x : INTEGER);
 BEGIN
   WHILE NOT Peticion DO ; (* ESPERA ACTIVA *)
   WRITELN('Servidor: Peticion atendida, x = ', x);
 END;

 PROCESS Cliente(VAR Peticion : BOOLEAN; VAR x : INTEGER);
 BEGIN
   x := 1;
   WRITELN('Cliente: Solicito peticion, x = ', x);
   Peticion := TRUE;
 END;

 VAR
  x : INTEGER;
  Peticion: BOOLEAN;

BEGIN
 x := 0;
 Peticion := FALSE;
 COBEGIN
   Servidor(Peticion, x);
   Cliente(Peticion, x);
 COEND
END.
```

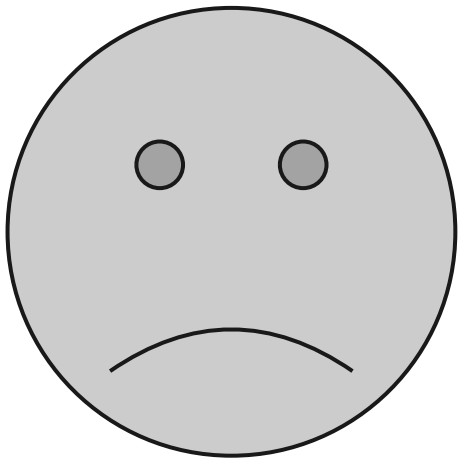


Sincronización Condicional

Justicia Fuerte: Petición Infinitamente Frecuente

```
PROCESS Cliente(VAR Peticion : BOOLEAN; VAR x : INTEGER);
  BEGIN
    REPEAT
      Peticion := TRUE;
      WRITELN('Solicito Peticion x = ', x);
      Peticion := FALSE;
      WRITELN('Rescindo Peticion x = ', x);
    FOREVER
  END;
```

El proceso servidor puede ejecutarse siempre con `Peticion := FALSE`



Exclusión Mutua

```
PROCESS P;
BEGIN
  REPEAT
    Preprotocolo;
    SeccionCritica;
    PostProtocolo;
    SeccionNoCritica;
  FOREVER
END;
```

REQUISITOS:

- Exclusión Mutua.
- Ausencia de Interbloqueo.
- Ausencia de retrasos innecesarios.
- Ausencia de inanición.

Exclusión Mutua

Primer Intento: Alternancia

```
PROGRAM PrimerIntento;
TYPE
  TAcceso = RECORD
    Turno : 1..2
    END;

PROCESS P1 (VAR Acceso : TAcceso);
BEGIN
  REPEAT
    (* Preprotocolo*)
    WHILE Acceso.Turno <> 1 DO;
    SeccionCritica1;
    Acceso.Turno := 2; (* Postprotocolo
*)
    SeccionNoCritica1;
  FOREVER
END;
```

Método trinca el turno..
Se cambian el turno, alternancia.

```
PROCESS P2 (VAR Acceso : TAcceso);
BEGIN
  REPEAT
    WHILE Acceso.Turno <> 2 DO; (* Preprotocolo*)

    SeccionCritica2;
    Acceso.Turno := 1; (* Postprotocolo *)
    SeccionNoCritica2;
  FOREVER
END;
VAR
 Acceso : TAcceso;
BEGIN
 Acceso.Turno := 1;
 COBEGIN
   P1(Acceso);
   P2(Acceso);
 COEND
END.
```

RETRASOS INNECESARIOS

¡ Un proceso necesita al otro !

Exclusión Mutua

Tercer Intento: Interbloqueo (declarar la intención de entrar)

```
PROGRAM TercerIntento;
TYPE
  TAcceso = RECORD
    Peticion1, Peticion2 : BOOLEAN
    END;

PROCESS P1 (VAR Acceso : TAcceso);
BEGIN
  REPEAT
    Acceso.Peticion1 := TRUE;
    WHILE Acceso.Peticion2 DO;
    SeccionCritica1;
    Acceso.Peticion1 := FALSE;
    SeccionNoCritica1;
  FOREVER
END;
```

Decir que quieres trabajar y levantar la mano

```
PROCESS P2 (VAR Acceso : TAcceso);
BEGIN
  REPEAT
    Acceso.Peticion2 := TRUE;
    WHILE Acceso.Peticion1 DO;
    SeccionCritica2;
    Acceso.Peticion2 := FALSE;
    SeccionNoCritica2;
  FOREVER
END;
VAR  Acceso : TAcceso;
BEGIN
 Acceso.Peticion1 := FALSE;
 Acceso.Peticion2 := FALSE;
 COBEGIN
   P1(Acceso);
   P2(Acceso);
 COEND
END.
```

Exclusión Mutua

Segundo Intento: Falta de Exclusión Mutua

```
PROGRAM SegundoIntento;
TYPE
  TAcceso = RECORD
    Peticion1, Peticion2 : BOOLEAN
    END;

PROCESS P1 (VAR Accesso : TAcceso);
BEGIN
  REPEAT
    WHILE Acceso.Peticion2 DO;
    Acceso.Peticion1 := TRUE;
    SeccionCritica1;
    Acceso.Peticion1 := FALSE;
    SeccionNoCritica1;
  FOREVER
END;
```

(Cuando un proceso entra, marca, y los demás tienen q esperar)

Ejemplo del ceda el paso…mirar a ver quien tiene la mano levantada para usar la escoba…
Levantas la mano mientras trabajas..

```
PROCESS P2 (VAR Accesso : TAcceso);
BEGIN
  REPEAT
    WHILE Acceso.Peticion1 DO;
    Acceso.Peticion2 := TRUE;
    SeccionCritica2;
    Acceso.Peticion2 := FALSE;
    SeccionNoCritica2;
  FOREVER
END;
VAR  Acceso : TAcceso;
BEGIN
 Acceso.Peticion1 := FALSE;
 Acceso.Peticion2 := FALSE;
 COBEGIN
   P1(Acceso);
   P2(Acceso);
 COEND
END.
```

Exclusión Mutua

Cuarto Intento: Espera Indefinida (Bloqueo e inanición)

```
PROCESS P1 (VAR Accesso : TAcceso);
BEGIN
  REPEAT
    Acceso.Peticion1 := TRUE;
    WHILE Acceso.Peticion2 DO
     BEGIN
       Acceso.Peticion1 := FALSE;
       Acceso.Peticion1 := TRUE;
     END;
    SeccionCritica1;
    Acceso.Peticion1 := FALSE;
    SeccionNoCritica1;
  FOREVER
END;
```

Nunca te deben interrumpir en una sección crítica…

Levanto la mano por quiero trabajar, y si el otro está trabajando la bajo…

Exclusión Mutua

Solución: El algoritmo de Dekker

```
PROCESS P1 (VAR Accesso : TAcceso);
BEGIN
  REPEAT
    Acceso.Peticion1 := TRUE;
    WHILE Acceso.Peticion2 DO
     IF Acceso.Turno <> 1 THEN
       BEGIN
         Acceso.Peticion1 := FALSE;
         WHILE Acceso.Turno <> 1 DO;
         Acceso.Peticion1 := TRUE;
     END;
    SeccionCritica1;
    Acceso.Peticion1 := FALSE;
    Acceso.Turno := 2;
    SeccionNoCritica1;
  FOREVER
END;
```

1. indico que quiero trabajar levanto mano..
2. si el otro tiene levantada la mano, y no me toca, la bajo y espero…y si me toca, paso de todo y entro…sólo bajas la mano cuando el otro la tiene levantada, o no es tu turno..