# Ejercicio Fumadores de picadillo – Paso de mensajes-monitores.

Miguel Á. Galdón Romero 

## *Enunciado*

Tres amigos se reúnen regularmente para fumar en un club de fumadores, fuman tabaco liado y cada uno de ellos tiene un elemento necesario para liar y encenderse un cigarro:

1- Papel, 2-Tabaco, 3-Fuego

El camarero del club, repetidamente pone dos ingredientes sobre la mesa y aquél que tiene el tercer ingrediente para poder fumarse un cigarro los coge y se lo lia, cuando termina el camarero vuelve a poner otros dos ingredientes sobre la mesa y el proceso se repite…

Consideramos un sistema compuesto por tres procesos fumares y un proceso agente (camarero) Cada liador continuamente hace un cigarro y se lo fuma, ero para poder fumarse el cigarro se necesitan tres ingredientes: papel, tabaco y cerillas.

El agente tiene una cantidad infinita de los tres ingredientes.

El agente pone dos de los ingredientes en la mesa.

El liador que tiene el ingrediente que falta puede hacerse el cigarrillo y fumárselo, indicando al agente cuando termine que se lo ha liado.

El agente entonces pone otros dos ingredientes y el ciclo se repite.

Buscamos una solución que y sincronice este comportamiento utilizando comunicación mediante paso de mensajes con canales síncronos con Pascal-FC.

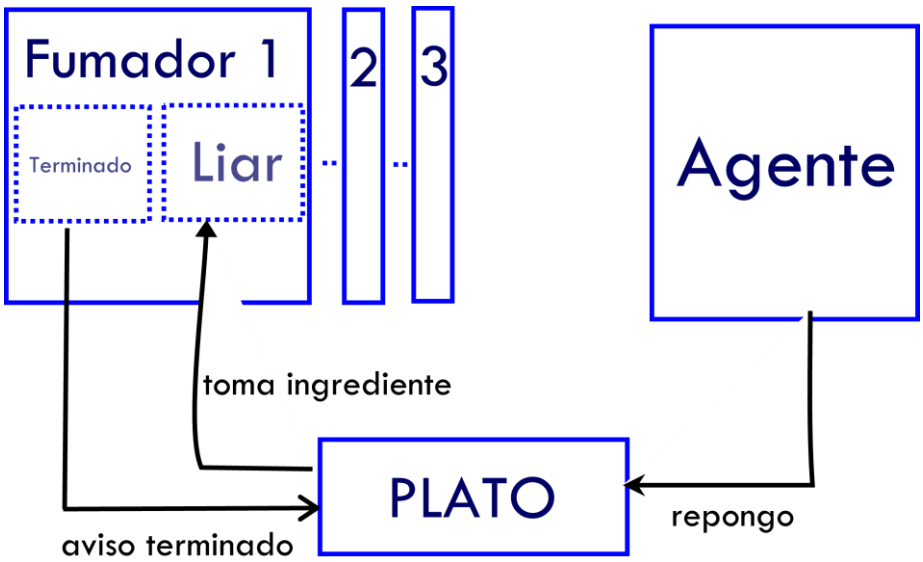
## *A) Solución 1 (Paso de mensajes):*

Para simplificar la solución asociaremos a ingrediente un número:

1- Papel, 2-Tabaco, 3-Fuego

Y a cada fumador también, por lo que cada fumador tendrá el ingrediente correspondiente con su número.

Tendremos los tres procesos fumadores, un proceso agente y un tercer proceso “mesa” que servirá de intermediario entre el agente y los fumadores, dónde aquél pondrá los ingredientes y los fumadores intentarán cogerlos para liarse un cigarro.



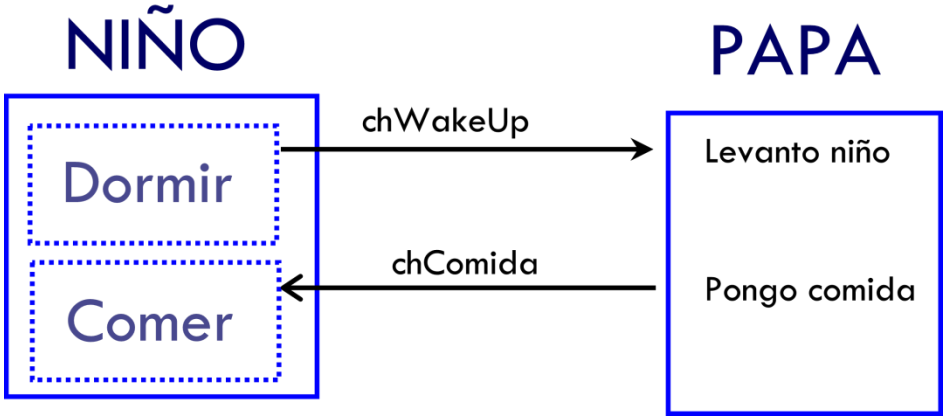

## *B) Solución 1 (monitores):*

La solución es muy similar en cuanto a los procesos y la lógica, tenemos dos tipos de procesos activos: fumadores y agente y un monitor para controlar la sincronización entre ambos y la exclusión mutua en el acceso a variables compartidas.

En este caso se ha ampliado la solución para que en lugar de 3 fumadores pueda ser cualquier número (>=3), para evitar bloqueo es necesario que al menos 3 tengan cada uno un ingrediente distinto, por eso a los tres primeros se les asignan secuencialmente y al resto aleatoriamente.

Ejercicio Fumadores de picadillo – Paso de mensajes-monitores.






```
program dadChild1;

const NPORC = 3; {número porciones}
      NVECES = 10; {número repeticiones}

type intchan =channel of integer;
     synchan =channel of synchronous;
var

 chComida: intchan;
 chWakeup: synchan;
```

```
procedure comer(var Porciones : integer; var chComida : intchan);
 begin
    if (porciones = 0) then
     chComida ? porciones;  {espero que pongan comida}
    porciones:=porciones-1; {me como 1}
    sleep(Random(4)+1); {comiendo}
 end;

 procedure dormir(var chWakeUp : synchan);
  begin
   sleep(5); {durmiendo}
   chWakeUp ! any; {Pedimos que nos levanten}
  end;

process  hijo;
 var i : integer;
     porciones:integer;
 begin
  porciones :=0; {inicialmente el plato está vacio}
  for i := 1 to NVECES do
   begin
   sleep(Random(5)+2);{jugar}
   comer(porciones,chComida);
   dormir(chWakeUp);
  end;
 end;
```

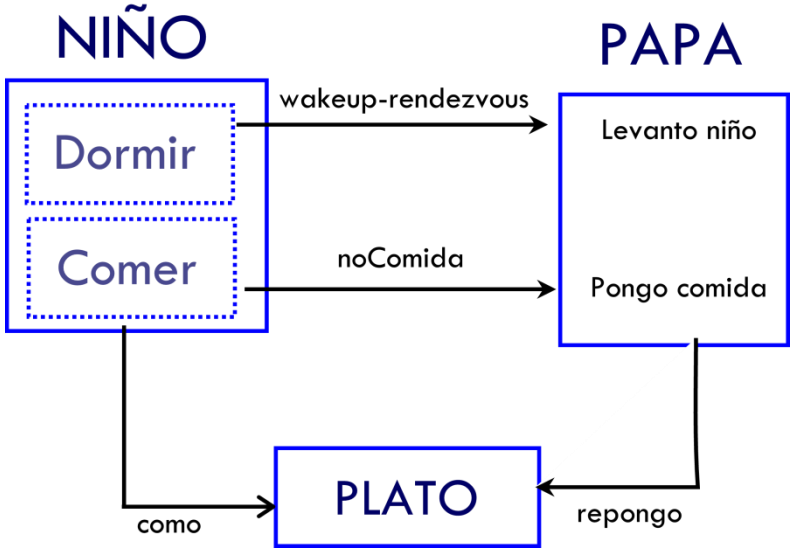
```
process papa;
 var i : integer;
 begin
  repeat
    select
       chComida ! NPORC; {Ponemos comida}
    or
       chWakeUp ? any; {Levantamos niño}
    or
       terminate;
    end;
  forever;
 end;
```

```
begin
 cobegin
  hijo;
  papa;
 coend;
end.
```

## *C) Solución 2 (Con proceso intermedio):*

La solución anterior es válida puesto que existe un solo niño, pero en general si varios procesos comparten un recurso, éste se debe gestionar a través de un proceso intermedio con el que nos comunicamos mediante mensajes.

Veamos la solución con un "proceso pasivo" intermedio ***plato***. (Por simplicidad de la solución seguiremos considerando que hay un solo niño, aunque podría generalizarse para funcionar con N niños)



```
program DadChild2;

const NPORC = 3; {número porciones}
      NVECES = 10; {número repeticiones}
type synchan = channel of synchronous;

var reponer : channel of synchronous;
    como,wakeup,noComida,levantame : synchan;
```

```
procedure jugar;
 begin
  sleep(Random(5)+2); {jugando}
 end;

procedure comer;
 var aviso : boolean;
 begin
   select
    como ! any;
   else
    noComida! any;
    como ! any;
   end; {select}
  sleep(Random(4)+1); {comiendo}
 end;

 procedure dormir;
  begin
   sleep(Random(5)+2);{durmiendo}
    levantame ! any;
  end;
```

```
process type Tninno;
 var i : integer;
 begin
  for i := 1 to NVECES do
   begin
   jugar;
   comer;
   dormir;
  end;
 end;
```

```
process papa;
 var i : integer;
 begin
  repeat
   select
    levantame ? any; {levanta niño}
   or
    noComida ? any;
    reponer ! any;  {pone comida}
   or
    terminate;
   end {select}
  forever;
 end;
```

```
process comedor;
{proceso pasivo }
 var  i : integer;
      porciones : integer;
begin
 porciones:=0;
 repeat
  select
   reponer ? any; {reponer}
   porciones:= NPORC;
  or
   when porciones > 0 =>
    como ? any; {el niño pide comida}
    porciones := porciones-1;
  or
   terminate;
 end;
 forever
end;
```

```
var ninno:  Tninno;
    i: integer;

begin
 cobegin
  ninno;
  papa;
  comedor;
coend;
end.
```