Enunciado

Un padre trabaja en casa y también se encuentra al cuidado de su hijo

El niño continuamente efectúa las siguientes tareas:

Inicialmente se encuentra jugando, cuando tiene hambre recurre a un plato y toma una porción de queso de él (en comer tarda un tiempo aleatorio), después duerme durante 5 unidades de tiempo y a continuación vuelve a sus juegos durante un tiempo aleatorio. Este ciclo se repite N veces.

Cuando el niño intenta comer, si encuentra el plato vacío, avisa al padre de este suceso, esperando entonces a que el padre ponga comida en el plato. El padre si no está atendiendo al niño se encuentra trabajando (estado pasivo, no en espera activa), cuando sea requerido realizará estas tareas:

1) Levantar al niños tras su descanso (el niño es muy dormilón y no se levantaría de otra forma).

2) Reponer el plato, cuando esté vacío, con X (por ejemplo 2) porciones de queso, si el hijo solicita comida.

La soluciones propuestas deben estar libres de interbloqueo e inanición y no tener esperas activas

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

Cuando el niño deje finalmente de jugar-comer-dormir, preferiblemente, todos los procesos deberían terminar.

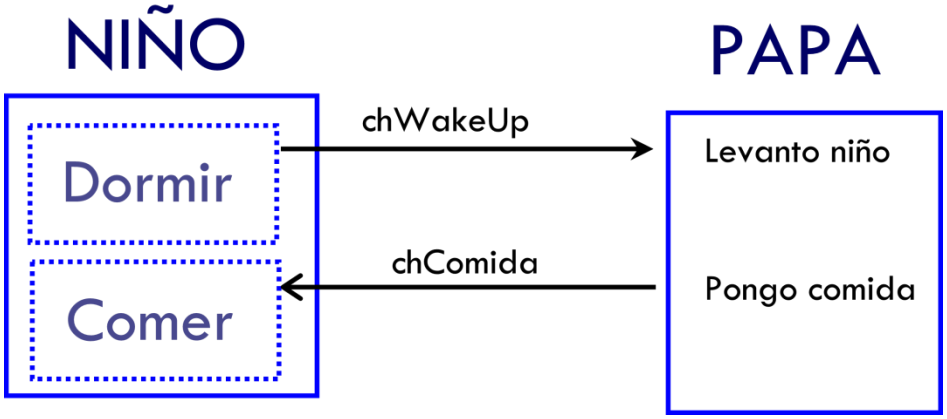
Para simplificar el código no se han añadido mensajes, puedes añadirlos para monitorizar la ejecución. Para mostrar mensajes necesitamos usar la “memoria compartida” de la pantalla, para evitar conflictos utiliza un semáforo para la exclusión mutua de pantalla.

A) Solución 1 (Simple):

En general, puesto que cuando trabajamos con paso de mensajes no disponemos de una memoria compartida, suele ser necesario un “proceso pasivo (***plato***)” que hace de intermediario entre los “procesos activos (***padre e hijo***)”. El problema es similar al conocido productor/consumidor, pero en este caso el productor (***padre***) produce bajo demanda.

En este caso dada la simplicidad del enunciado al haber un solo niño ni si quiera necesitamos explícitamente ese proceso intermedio (***plato***), puesto que el niño no comparte el plato con nadie, el propio proceso ***niño*** puede tener el plato y pedirle al padre que le ponga porciones.

Veamos una posible solución:




```
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;
```

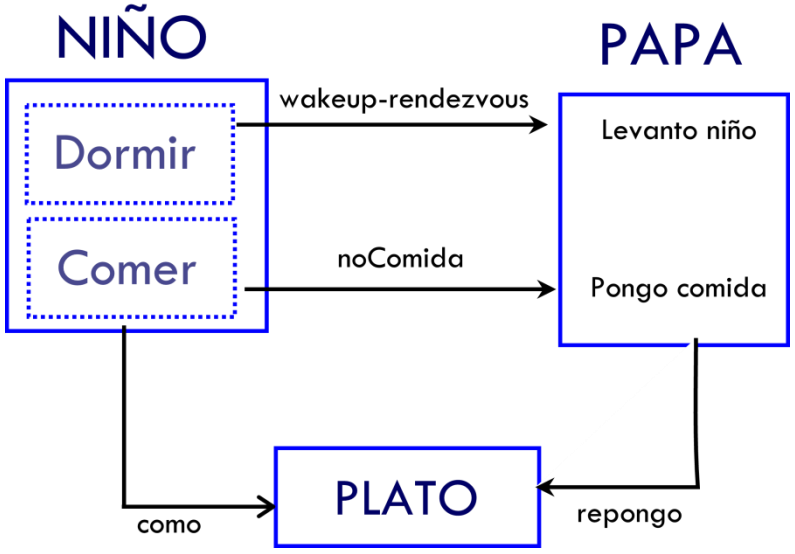
```
process papa;
 var i : integer;
 begin
  repeat
    select
       chComida ! NPORC; {Ponemos comida}
    or
       chWakeUp ? any; {Levantamos niño}
    or
       terminate;
    end;
  forever;
 end;
```

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

B) 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.
```