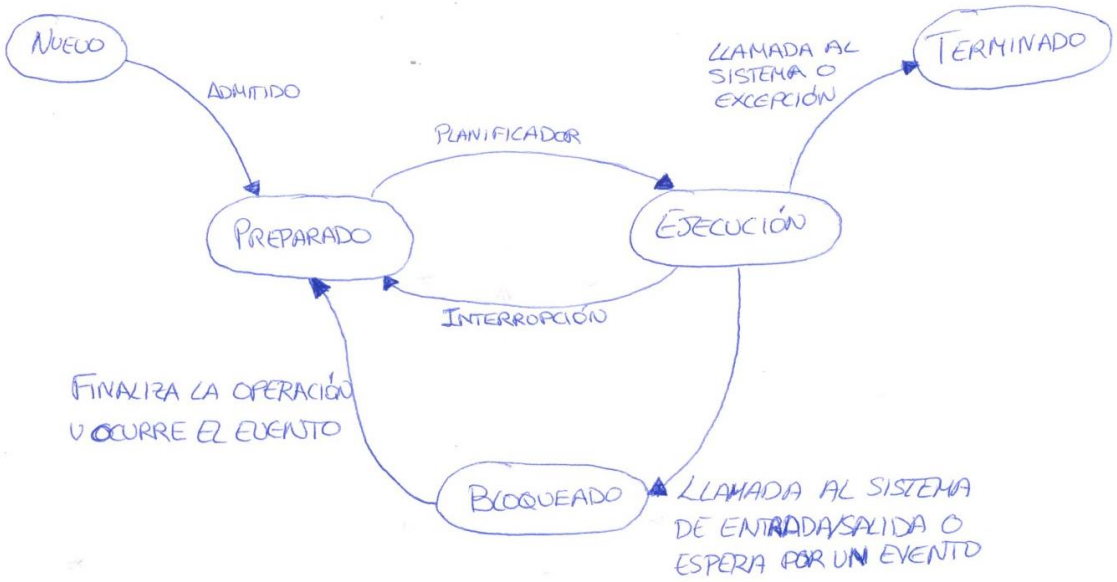
UF2. [PAC01] PROCESOS

Actividades

Parte teórica

1. Haz un esquema de todos los estados de un proceso, indicando que ocurre en cada uno de ellos.



Nuevo: El proceso se acaba de crear, pero aún no ha sido admitido en el grupo de procesos ejecutables por el sistema operativo.

Listo: El proceso está parado temporalmente y lista para ejecutarse cuando se le dé la oportunidad.

En ejecución: El proceso está actualmente ejecutándose, es decir, usando el procesador.

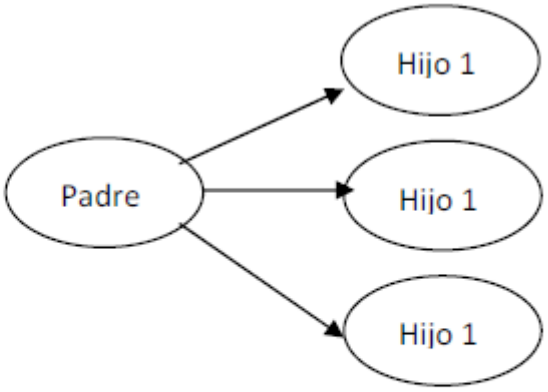
Bloqueado: El proceso no puede hacer nada hasta que no ocurra un evento externo.

Terminado: El proceso ha sido sacado del grupo de procesos ejecutables por el sistema operativo. Después de que un proceso es marcado como terminado se liberarán los recursos utilizados por ese proceso.

Parte práctica

2. Realiza un programa en C que genere una estructura de procesos con un padre y 3 hijos. Visualiza por cada hijo su PID y el del padre. Visualiza también el PID del padre de todos.

A continuación se muestra un ejemplo de ejecución:




```
Soy el hijo= 3, Mi padre es= 5086, Mi PID= 5089
Soy el hijo= 2, Mi padre es= 5086, Mi PID= 5088
Soy el hijo= 1, Mi padre es= 5086, Mi PID= 5087
Proceso PADRE = 5086
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

void main(){
        pid_t pid;
        int i;

        for(i=1; i<4; i++)
        {
                pid = fork();
                //Proceso padre
                if(pid==-1) //ha ocurrido un error
                {
                        printf("No se ha podido crear el proceso hijo %d.\n", i);
                        exit(-1);
                }
                //Proceso hijo
                else if(pid==0) //soy el hijo
                {
```

```
                        printf("Soy el hijo %d. Mi padre es el proceso: %d. Y mi PID es: %d\n",i,getppid(),getpid());
                        exit(0);
                }
        }
        //Proceso padre
        //En el caso que no hubiese puesto el exit al final del cÃ³digo del hijo, deberÃ-a poner aquÃ- un if(pid!=0) para asegurar que este cÃ³digo solo lo ejecute el padre

        printf("Soy el proceso padre. Mi PID es: %d\n", getpid());
        //Espero que finalicen los 3 hijos antes de finalizar el padre
        for(i=0;i<3;i++)
                wait(NULL);
        exit (0);
}
```

3. A partir del siguiente conjunto de instrucciones indica las que se pueden ejecutar concurrentemente y las que no. Justifica tus respuestas.

   Instrucción 1: **a:=x+y;**

   Instrucción 2: **b:=z-1;**

   Instrucción 3: **c:=a-b;**

   Instrucción 4: **w:=c+1;**

Primero hacemos los conjuntos de lectura y escritura de las distintas instrucciones.

L(I1) = {x,y} E(I1) = {a}

L(I2) = {z} E(I2) = {b}

L(I3) = {a,b} E(I3) = {c}

L(I4) = {c} E(I4) = {z}

Para que dos conjuntos se puedan ejecutar concurrentemente se deben cumplir estas 3 condiciones:

- La inserción entre las variables leídas por un conjunto de instrucciones I_i y las variables escritas por otro conjunto I_j debe ser vacío, es decir, no debe haber variables comunes:

$$L(I_i) \cap E(I_j) = \emptyset$$

- La inserción entre las variables de escritura de un conjunto de instrucciones I_i y las variables leídas por otro conjunto I_j debe ser nulo, es decir, no debe haber variables comunes:

$$E(I_i) \cap L(I_j) = \emptyset$$

- Por último, la intersección entre las variables de escritura de un conjunto de instrucciones I_i y las variables de escritura de un conjunto I_j debe ser vacío, no debe haber variables comunes:

$$E(I_i) \cap E(I_j) = \emptyset$$

Aplicando estas tres condiciones a cada pareja de instrucciones descubriremos cuales se pueden ejecutar concurrentemente y cuales no:

Conjunto de I_1 e I_2	**Conjunto de I1 e I4**	**Conjunto de I2 e I4**
$L(I_1) \cap E(I_2) = \emptyset$	$L(I_1) \cap E(I_4) = \emptyset$	$L(I_2) \cap E(I_4) = \emptyset$
$E(I_1) \cap L(I_2) = \emptyset$	$E(I_1) \cap L(I_4) = \emptyset$	$E(I_2) \cap L(I_4) = \emptyset$
$E(I_1) \cap E(I_2) = \emptyset$	$E(I_1) \cap E(I_4) = \emptyset$	$E(I_2) \cap E(I_4) = \emptyset$
Conjunto de I1 e I3	**Conjunto de I2 e I3**	**Conjunto de I3 e I4**
$L(I_1) \cap E(I_3) = \emptyset$	$L(I_2) \cap E(I_3) = \emptyset$	$L(I_3) \cap E(I_4) = \emptyset$
$E(I_1) \cap L(I_3) \neq \emptyset$	$E(I_2) \cap L(I_3) \neq \emptyset$	$E(I_3) \cap L(I_4) \neq \emptyset$
$E(I_1) \cap E(I_3) = \emptyset$	$E(I_2) \cap E(I_3) = \emptyset$	$E(I_3) \cap E(I_4) = \emptyset$

Por lo tanto, las instrucciones que se pueden ejecutar concurrentemente son: I_1 e I_2, I_1 e I_4 y también I_2 e I_4.