



INSTITUTO POLITÉCNICO NACIONAL  
UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA Y  
CIENCIAS SOCIALES Y ADMINISTRATIVAS



# Práctica 6. Semáforos

Yong Rodríguez Luz María

### *Introducción.*

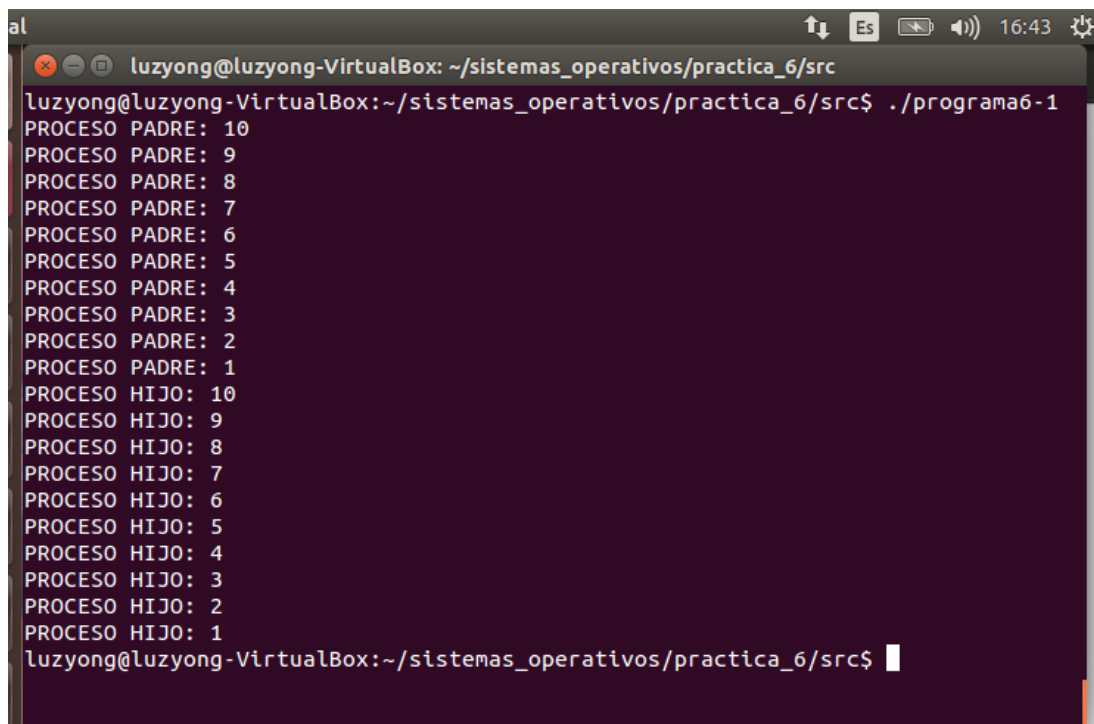
Los semáforos son un mecanismo que nos permiten alternar entre la ejecución de dos o más procesos que existen al mismo tiempo. Un semáforo en el sistema operativo permite que se ejecuten las instrucciones de un programa, mientras impide que los demás programas tengan tiempo de cómputo y ejecuten sus instrucciones al mismo tiempo. De forma análoga a un semáforo de la vida real, solo un proceso a la vez está en “verde” y se puede ejecutar, cuando termina, el proceso realiza el cambio, se pone en “rojo” y le asigna “verde” al siguiente.

En el sistema operativo, los valores análogos al verde y rojo son números enteros positivos, cero o menos uno. Si el proceso se encuentra en -1, se bloqueará hasta que otro proceso le asigne un valor diferente.

### *Ejemplos prácticos.*

Programa 6-1: Impresión concurrente de dos procesos.

En este programa se ejecutan dos procesos, primero todas las operaciones del proceso padre y después todas las del proceso hijo.



```
al
luzyong@luzyong-VirtualBox: ~/sisistemas_operativos/practica_6/src
luzyong@luzyong-VirtualBox:~/sisistemas_operativos/practica_6/src$ ./programa6-1
PROCESO PADRE: 10
PROCESO PADRE: 9
PROCESO PADRE: 8
PROCESO PADRE: 7
PROCESO PADRE: 6
PROCESO PADRE: 5
PROCESO PADRE: 4
PROCESO PADRE: 3
PROCESO PADRE: 2
PROCESO PADRE: 1
PROCESO HIJO: 10
PROCESO HIJO: 9
PROCESO HIJO: 8
PROCESO HIJO: 7
PROCESO HIJO: 6
PROCESO HIJO: 5
PROCESO HIJO: 4
PROCESO HIJO: 3
PROCESO HIJO: 2
PROCESO HIJO: 1
luzyong@luzyong-VirtualBox:~/sisistemas_operativos/practica_6/src$
```

Programa 6-2: Sincronización de impresiones usando semáforos.

En este programa se sincronizan los procesos para que de forma alternada se ejecuten, es decir, una operación del padre y después una operación del hijo.

```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_6/src
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$ ./programa6-2
SOY EL PROCESO PADRE. IMPRESION: 10
SOY EL PROCESO HIJO. IMPRESION:10
SOY EL PROCESO PADRE. IMPRESION: 9
SOY EL PROCESO HIJO. IMPRESION:9
SOY EL PROCESO PADRE. IMPRESION: 8
SOY EL PROCESO HIJO. IMPRESION:8
SOY EL PROCESO PADRE. IMPRESION: 7
SOY EL PROCESO HIJO. IMPRESION:7
SOY EL PROCESO PADRE. IMPRESION: 6
SOY EL PROCESO HIJO. IMPRESION:6
SOY EL PROCESO PADRE. IMPRESION: 5
SOY EL PROCESO HIJO. IMPRESION:5
SOY EL PROCESO PADRE. IMPRESION: 4
SOY EL PROCESO HIJO. IMPRESION:4
SOY EL PROCESO PADRE. IMPRESION: 3
SOY EL PROCESO HIJO. IMPRESION:3
SOY EL PROCESO PADRE. IMPRESION: 2
SOY EL PROCESO HIJO. IMPRESION:2
SOY EL PROCESO PADRE. IMPRESION: 1
SOY EL PROCESO HIJO. IMPRESION:1
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$
```

- Pregunta 6-1: Imprima el valor de los miembros de la variable que se declaró estructura sembuf antes de llamar a ftok. ¿Qué valores iniciales tiene?

2052, 24576 y -18570 respectivamente

```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_6/src
pregunta6-1 pregunta6-2-2.c programa6-1
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$ ./pregunta6-1
sem_num:2052 sem_op:24576 sem_flg:-18570
SOY EL PROCESO PADRE. IMPRESION: 10
SOY EL PROCESO HIJO. IMPRESION:10
SOY EL PROCESO PADRE. IMPRESION: 9
SOY EL PROCESO HIJO. IMPRESION:9
SOY EL PROCESO PADRE. IMPRESION: 8
SOY EL PROCESO HIJO. IMPRESION:8
SOY EL PROCESO PADRE. IMPRESION: 7
SOY EL PROCESO HIJO. IMPRESION:7
SOY EL PROCESO PADRE. IMPRESION: 6
SOY EL PROCESO HIJO. IMPRESION:6
SOY EL PROCESO PADRE. IMPRESION: 5
SOY EL PROCESO HIJO. IMPRESION:5
SOY EL PROCESO PADRE. IMPRESION: 4
SOY EL PROCESO HIJO. IMPRESION:4
SOY EL PROCESO PADRE. IMPRESION: 3
SOY EL PROCESO HIJO. IMPRESION:3
SOY EL PROCESO PADRE. IMPRESION: 2
SOY EL PROCESO HIJO. IMPRESION:2
SOY EL PROCESO PADRE. IMPRESION: 1
SOY EL PROCESO HIJO. IMPRESION:1
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$
```

Cuando se modifican sus valores, ¿Qué función hace uso de dicha variable?  
Semop hace uso de la variable “operación”

- Pregunta 6-2: ¿Continúa funcionando el programa si se declara un conjunto de 5 semáforos en la línea segmet?

Sí

```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_6/src
bash: ./pregunta6-2: No existe el archivo o el directorio
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$ ./pregunta6-2-1
sem_num:2052 sem_op:8192 sem_flg:-18580
SOY EL PROCESO PADRE. IMPRESION: 10
SOY EL PROCESO HIJO. IMPRESION:10
SOY EL PROCESO PADRE. IMPRESION: 9
SOY EL PROCESO HIJO. IMPRESION:9
SOY EL PROCESO PADRE. IMPRESION: 8
SOY EL PROCESO HIJO. IMPRESION:8
SOY EL PROCESO PADRE. IMPRESION: 7
SOY EL PROCESO HIJO. IMPRESION:7
SOY EL PROCESO PADRE. IMPRESION: 6
SOY EL PROCESO HIJO. IMPRESION:6
SOY EL PROCESO PADRE. IMPRESION: 5
SOY EL PROCESO HIJO. IMPRESION:5
SOY EL PROCESO PADRE. IMPRESION: 4
SOY EL PROCESO HIJO. IMPRESION:4
SOY EL PROCESO PADRE. IMPRESION: 3
SOY EL PROCESO HIJO. IMPRESION:3
SOY EL PROCESO PADRE. IMPRESION: 2
SOY EL PROCESO HIJO. IMPRESION:2
SOY EL PROCESO PADRE. IMPRESION: 1
SOY EL PROCESO HIJO. IMPRESION:1
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$
```

¿Qué ocurre si se inicializan los semáforos en cero?

No se puede ejecutar el programa.

```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_6/src
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$ ./pregunta6-2-1
sem_num:2052 sem_op:-8192 sem_flg:-18576
Error al ejecutar segmet: Invalid argument
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$
```

¿Qué pasa si se inicializa el semáforo hijo en uno y el semáforo padre en cero?

Se ejecuta primero el que tenga el valor de 0, es decir, el padre en este caso.

```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_6/src
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$ ./pregunta6-2-3
sem_num:2052 sem_op:24576 sem_flg:-18574
SOY EL PROCESO PADRE. IMPRESION: 10
SOY EL PROCESO HIJO. IMPRESION:10
SOY EL PROCESO PADRE. IMPRESION: 9
SOY EL PROCESO HIJO. IMPRESION:9
SOY EL PROCESO PADRE. IMPRESION: 8
SOY EL PROCESO HIJO. IMPRESION:8
SOY EL PROCESO PADRE. IMPRESION: 7
SOY EL PROCESO HIJO. IMPRESION:7
SOY EL PROCESO PADRE. IMPRESION: 6
SOY EL PROCESO HIJO. IMPRESION:6
SOY EL PROCESO PADRE. IMPRESION: 5
SOY EL PROCESO HIJO. IMPRESION:5
SOY EL PROCESO PADRE. IMPRESION: 4
SOY EL PROCESO HIJO. IMPRESION:4
SOY EL PROCESO PADRE. IMPRESION: 3
SOY EL PROCESO HIJO. IMPRESION:3
SOY EL PROCESO PADRE. IMPRESION: 2
SOY EL PROCESO HIJO. IMPRESION:2
SOY EL PROCESO PADRE. IMPRESION: 1
SOY EL PROCESO HIJO. IMPRESION:1
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$
```

- Pregunta 6-3: Imprima los miembros `sem_num` y `sem_op` de la variable operación exactamente antes de llamar a `semop`. Explique brevemente cómo se está logrando sincronizar a los procesos.

Al principio creí que era una operación AND entre los valores de estos miembros, sin embargo, haciendo un análisis, observé que se hace una suma de ambos valores, siendo el primer valor el proceso que se va a ejecutar enseguida, el segundo valor la operación a hacerle al proceso y la suma es el siguiente proceso para ejecutar si el valor es 0 o >0, o el bloqueo del proceso actual si el valor es -1. Algo curioso que noté es que, si la suma da un valor entero que no es ninguno de los valores de los semáforos, va a la operación del semáforo que se está ejecutando actualmente, sin embargo, se salta el imprimir y asigna valores de nuevo a las variables. Ese número coincide con el de los semáforos declarados.

```

luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_6/src
pregunta6-1      pregunta6-2-2.c  programa6-1
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$ ./pregunta6-3
entrando a propadre sem_num:2052 sem_op:8192
entrando a prohiho sem_num:2052 sem_op:8192
despues de las operaciones preprint hijo num:1 op:-1
SOY EL PROCESO HIJO. IMPRESION:10
despues de las primeras operaciones pre printpadre num:0 op:-1
SOY EL PROCESO PADRE. IMPRESION: 10
despues de las segundas operaciones postprint padre num:1 op:1
entrando a propadre sem_num:1 sem_op:1
despues de las operaciones postprint hijo num:0 op:1

entrando a prohiho sem_num:0 sem_op:1
despues de las operaciones preprint hijo num:1 op:-1
SOY EL PROCESO HIJO. IMPRESION:9
despues de las primeras operaciones pre printpadre num:0 op:-1
SOY EL PROCESO PADRE. IMPRESION: 9
despues de las segundas operaciones postprint padre num:1 op:1
entrando a propadre sem_num:1 sem_op:1
despues de las operaciones postprint hijo num:0 op:1

entrando a prohiho sem_num:0 sem_op:1
despues de las operaciones preprint hijo num:1 op:-1
SOY EL PROCESO HIJO. IMPRESION:8

```

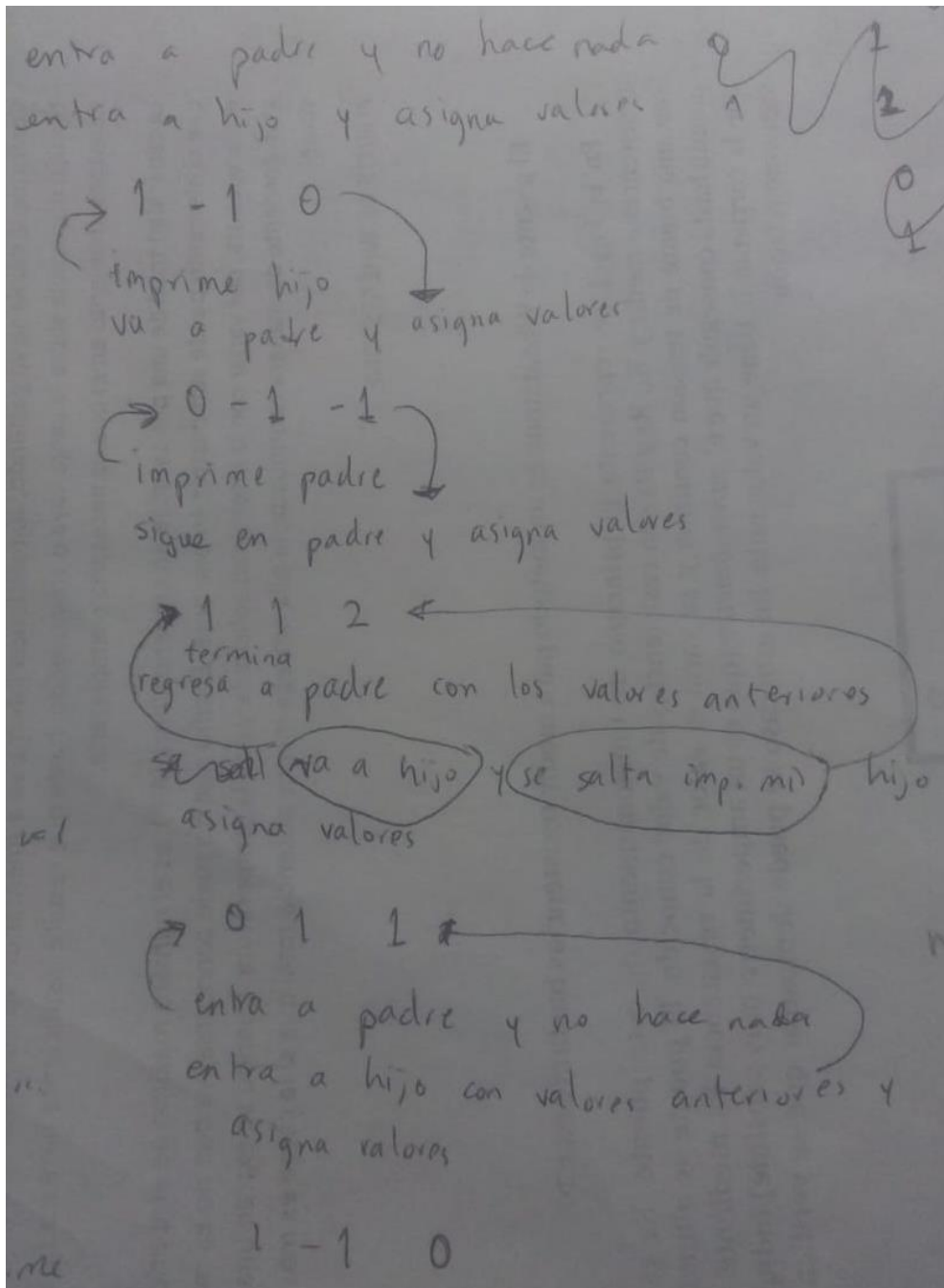
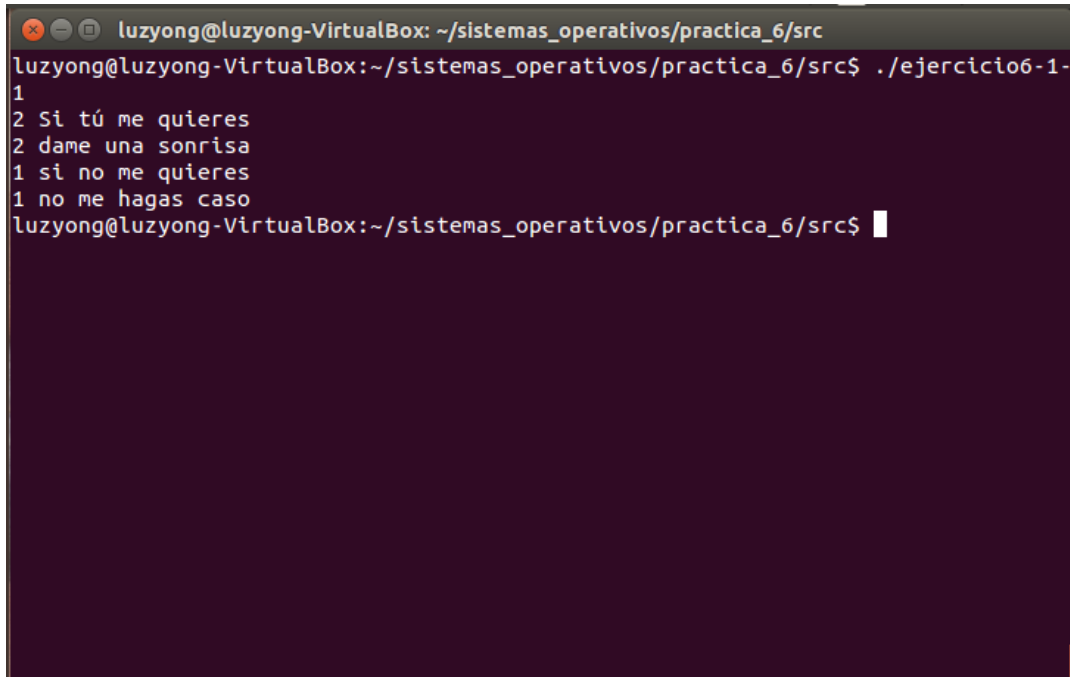


Diagrama que explica la forma en la que analicé el funcionamiento del semáforo.

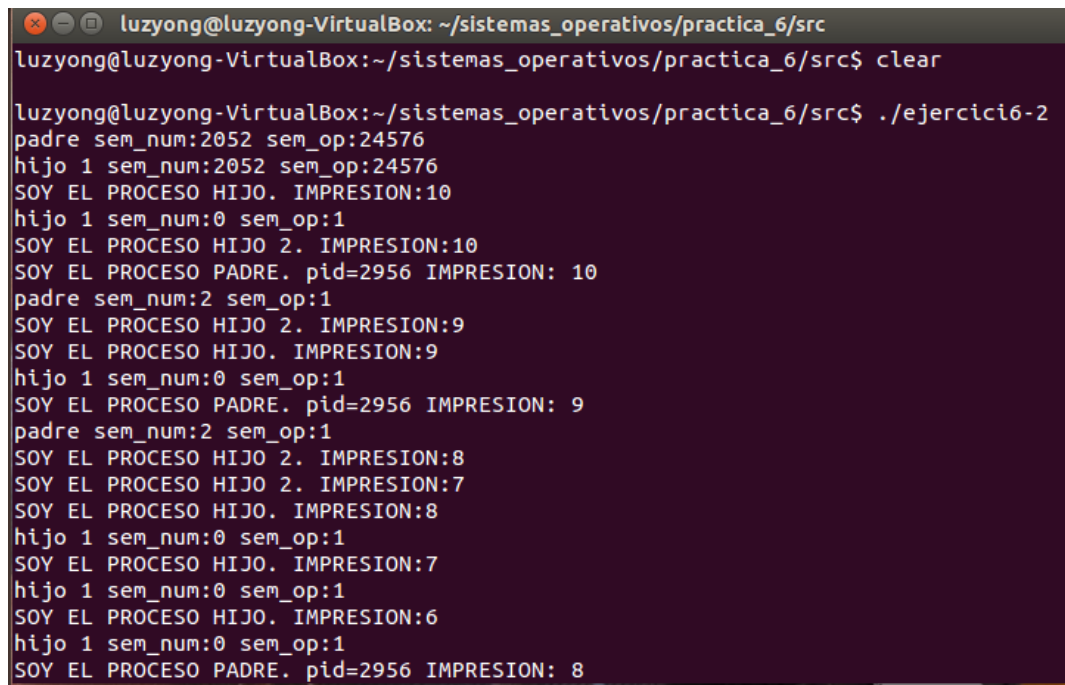
Ejercicio 6-1: Elabore dos programas independientes que sincronicen sus impresiones del mismo modo que lo hacen el padre y el hijo en el programa actual.



```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_6/src
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$ ./ejercicio6-1
1
2 Si tú me quieres
2 dame una sonrisa
1 si no me quieres
1 no me hagas caso
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$
```

Ejercicios 6-2: Modifique el programa 6-2 para que se sincronicen tres procesos independientes.

Se modificó el programa para que tres procesos independientes se sincronicen. Desde mi forma de analizar el programa, llegué a la siguiente solución, que realiza el programa padre 3 veces, después el programa hijo 1 tres veces y después el programa hijo 2 otras tres veces hasta que los tres procesos terminan su ejecución.



```
luzyong@luzyong-VirtualBox: ~/sistemas_operativos/practica_6/src
luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$ clear

luzyong@luzyong-VirtualBox:~/sistemas_operativos/practica_6/src$ ./ejercicio6-2
padre sem_num:2052 sem_op:24576
hijo 1 sem_num:2052 sem_op:24576
SOY EL PROCESO HIJO. IMPRESION:10
hijo 1 sem_num:0 sem_op:1
SOY EL PROCESO HIJO 2. IMPRESION:10
SOY EL PROCESO PADRE. pid=2956 IMPRESION: 10
padre sem_num:2 sem_op:1
SOY EL PROCESO HIJO 2. IMPRESION:9
SOY EL PROCESO HIJO. IMPRESION:9
hijo 1 sem_num:0 sem_op:1
SOY EL PROCESO PADRE. pid=2956 IMPRESION: 9
padre sem_num:2 sem_op:1
SOY EL PROCESO HIJO 2. IMPRESION:8
SOY EL PROCESO HIJO 2. IMPRESION:7
SOY EL PROCESO HIJO. IMPRESION:8
hijo 1 sem_num:0 sem_op:1
SOY EL PROCESO HIJO. IMPRESION:7
hijo 1 sem_num:0 sem_op:1
SOY EL PROCESO HIJO. IMPRESION:6
hijo 1 sem_num:0 sem_op:1
SOY EL PROCESO PADRE. pid=2956 IMPRESION: 8
```



luzyong@luzyong-VirtualBox: ~/sistemas\_operativos/practica\_6/src

```
padre sem_num:2 sem_op:1
SOY EL PROCESO PADRE. pid=2956 IMPRESION: 7
padre sem_num:2 sem_op:1
SOY EL PROCESO PADRE. pid=2956 IMPRESION: 6
SOY EL PROCESO HIJO 2. IMPRESION:6
padre sem_num:2 sem_op:1
SOY EL PROCESO HIJO 2. IMPRESION:5
SOY EL PROCESO HIJO. IMPRESION:5
hijo 1 sem_num:0 sem_op:1
SOY EL PROCESO HIJO. IMPRESION:4
hijo 1 sem_num:0 sem_op:1
SOY EL PROCESO PADRE. pid=2956 IMPRESION: 5
padre sem_num:2 sem_op:1
SOY EL PROCESO HIJO 2. IMPRESION:4
SOY EL PROCESO HIJO 2. IMPRESION:3
SOY EL PROCESO PADRE. pid=2956 IMPRESION: 4
padre sem_num:2 sem_op:1
SOY EL PROCESO HIJO 2. IMPRESION:2
SOY EL PROCESO HIJO. IMPRESION:3
hijo 1 sem_num:0 sem_op:1
SOY EL PROCESO HIJO. IMPRESION:2
hijo 1 sem_num:0 sem_op:1
SOY EL PROCESO HIJO. IMPRESION:1
SOY EL PROCESO HIJO 2. IMPRESION:1
```

entra a padre y no hace nada  
entra a hijo 1 y no hace nada  
entra a hijo 2 y asigna valores

2 -1 1

imprime hijo 2

va a hijo 1 y asigna valores

0 -1 -1 0

~~regresa a hijo 1 y asigna~~

imprime hijo 1 y va a padre  
va a padre y asigna valores

0 -1 -1

imprime padre

3 que en padre y asigna valores

2 1 3

termina, regresa a padre con valores anteriores

va a hijo 2 y se salta imprimir  
asigna valores

1 1 2

entra a hijo 1 y no hace nada

entra a hijo 2 con valores anteriores y asigna  
valores

2 -1 1

## Conclusiones.

En esta práctica aprendí sobre el funcionamiento de los semáforos. Pude experimentar con la creación y modificación de uno. Todavía quedan unas pequeñas dudas, pero en general, me quedó clara su forma de actuar dentro del sistema operativo.