# Reporte de la práctica 7

**Hernández Tapia Luis Enrique**

**Sistemas Operativos**
**Grupo 2CM7**
**Profesor: Montes Casiano Hermes Francisco**
13 de abril de 2018

## 1. Introducción

Un semáforo en los sistemas operativos permite que se ejecuten instrucciones de un programa, mientras impide que el administrador de procesos le asigne el procesador a otros programas y, en consecuencia, que se ejecuten sus instrucciones. Para esto deben compartid el mismo semáforo.

## 2. Desarrollo

Se nos pide escribir el siguiente programa 7-1 y guardarlo como programa7-1.c, como ejemplo demostrativo que no están sincronizados ele proceso hijo y padre.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main(){
    int i = 10, pid;
    //Creacion del proceso hijo
    if ((pid = fork()) == -1){
        perror("fork");
        exit(-1);
    }
    else if (pid == 0){
        while(i)
        {
                printf("PROCESO HIJO: %d\n", i--);
        }
    }else
    {
        while(i)
        {
```

```
21                  printf("PROCESO PADRE: %d\n", i--);
22          }
23      }
24      return 0;
25  }
```

// Enseguida el programa 7-2 y guardarlo como programa7-2.c, nos muestra el trabajo de un semáforo y como alterna el proceso padre e hijo.

```
1   /*Programa para ilustrar el uso de semaforos*/
2   #include <stdio.h>
3   #include <stdlib.h>
4   #include <unistd.h>
5   #include <sys/types.h>
6   #include <sys/ipc.h>
7   #include <sys/sem.h>
8
9
10  #define SEMAFORO_PADRE 1
11  #define SEMAFORO_HIJO 0
12
13  int main(int argc, char const *argv[])
14  {
15          int semid, pid, j = 10;
16          struct sembuf operacion;
17          key_t llave;
18          llave = ftok(argv[0], 'U');
19
20          if ((semid = semget(llave,2,IPC_CREAT | 0600)) == -1)
21          {
22                  perror("Error al ejecutar segmet");
23                  exit(-1);
24          }
25          semctl (semid, SEMAFORO_HIJO, SETVAL, 0);
26          semctl (semid, SEMAFORO_PADRE, SETVAL, 1);
27
28          /*Se cre el proceso hijo*/
29          if ((pid = fork()) == -1)
30          {
31                  perror("Error al ejecutar fork");
32                  exit(-1);
33          }
34          else if (pid == 0)
35          {
36                  /*Codigo correspondiente al proceso hijo*/
37                  while(j)
```

```
38                      {
39                              /*Se realiza la operacion DOWN en el
                                   semaforo del proceso hijo*/
40                              operacion.sem_flg = 0;
41                              operacion.sem_op = 1;
42                              operacion.sem_num = SEMAFORO_HIJO;
43                              semop(semid, &operacion,1);
44                              printf("SOY EL PROCESO HIJO. IMPRESION:
                                   %d\n", j--);
45                              /*Se realiza la operacion UP en el
                                   semaforo del proceso padre*/
46                              operacion.sem_op = 1;
47                              operacion.sem_num = SEMAFORO_PADRE;
48                              semop(semid,&operacion,1);
49                      }
50                      /*Borramos el semaforo*/
51                      semctl(semid,0,IPC_RMID,0);
52              }else
53              {
54                      /*Codigo correspondiente al proceso padre*/
55                                      while(j)
56                      {
57                              /*Se realiza la operacion DOWN en el
                                   semaforo del proceso padre*/
58                              operacion.sem_flg = 0;
59                              operacion.sem_op = 1;
60                              operacion.sem_num = SEMAFORO_PADRE;
61                              semop(semid, &operacion,1);
62                              printf("SOY EL PROCESO PADRE. IMPRESION:
                                    %d\n", j--);
63                              /*Se realiza la operacion UP en el
                                   semaforo del proceso hijo*/
64                              operacion.sem_op = 1;
65                              operacion.sem_num = SEMAFORO_HIJO;
66                              semop(semid,&operacion,1);
67                      }
68                      /*Borramos el semaforo*/
69                      semctl(semid,0,IPC_RMID,0);
70
71              }
72              return 0;
73      }
```

## 3. Pruebas

**Con el programa 7-1 se pide responder las siguientes preguntas.**

**Pregunta 7.1** - Imprima los valores de los miembros de la variable que se declaró como estructura *sembuf* antes de llamar a ftok, ¿qué función hace uso de dicha variable?.
**R:**
Tienen valores iniciales con unsigned short, después son utilizados en el DOWN así como el UP, hacen uso el padre y el hijo al momento de sincronizarse.

**Pregunta 7.2** - ¿Qué ocurre si inicializamos los valores de los semáforos en 0?
**R:** se permite a mas o menos procesos utilizar el recurso en forma simultanea.

**Pregunta 7.3** - Imprima los miembros *sem-num* y *sem-op* de la variable operacion exactamente antes de llamar a *semop*. Explique la sincronización.
**R:** El proceso HIJO hace DOWN en operacion.sem-num, lo que bloque al PADRE de forma temporal, cuando termina hace UP, entonces el proceso PADRE puede hacer DOWN.

## 4. Conclusiones

Como vemos, el proceso de los semáforos requiere colaboración de los procesos. Un proceso debe decrementar el contador antes de acceder al fichero e incrementarlo cuando termine. Si los procesos no siguen este ”protocolo”(y pueden no hacerlo), el semáforo no sirve de nada.