7 Semáforos

Introducción

Otro mecanismo importante en los sistemas multiproceso consiste en los semáforos. De manera análoga al funcionamiento de los semáforos en los cruces viales, donde impiden el flujo de autos en una dirección y lo permiten en otra dirección perpendicular, 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. Cabe aclarar que estos procesos deben compartir el "mismo" semáforo, porque de otro modo no existirá sincronización alguna. En realidad se tiene un conjunto de semáforos que funcionan de manera sincronizada (sólo uno de ellos está en "verde" en un momento dado), agrupados bajo un identificador único el cual se utiliza como referencia. En muchas situaciones se utilizan los semáforos para permitir a un proceso ejecutar un conjunto de instrucciones dado, mientras se impide a los demás ejecutar ese mismo conjunto de instrucciones.

Para que un semáforo cambie de estado, será necesario que el proceso que se está ejecutando realice dicho cambio. En lugar de cambiar de color, en cómputo los semáforos cambian su valor. Dicho valor es un entero positivo o cero y se inicializa al inicio del programa. Las operaciones sobre el semáforo hacen que se decremente o que se incremente su valor una unidad. Pero cuando un semáforo tiene el valor de cero y se lleva a cabo un decremento, el proceso que intenta decrementarlo se bloqueará inmediatamente. Este proceso no se va a desbloquear sino hasta que otro proceso realice una operación de incremento sobre el semáforo y en consecuencia su valor sea uno. Así, el proceso que se encontraba bloqueado, podrá realizar su operación de decremento, dejando el valor final del semáforo en cero. De manera análoga a la memoria compartida, se debe tener una llave compartida para la creación del semáforo.

Ejemplo práctico

Compile y ejecute el código del programa 7-1.

Programa 7-1
Impresión concurrente de dos procesos

```
int main(void)
{
   int i = 10, pid;

   /* Creación 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)
      {
         printf("PROCESO PADRE: %d\n", i--);
      }
   }
}
```

Al ejecutarlo podemos observar que aparecen todas las impresiones de un proceso y después las impresiones del otro proceso. Si queremos sincronizar las impresiones del proceso padre y el proceso hijo para que aparezcan alternadas, tendremos que hacer uso de semáforos como se muestra en el programa 7-2 (ejecútelo también).

Programa 7-2
Sincronización de impresiones usando semáforos

```
/* Programa para ilustrar el uso de semáforos */

#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

#define SEMAFORO_PADRE 1
#define SEMAFORO_HIJO  0

int main(int argc, char *argv[])
{
   int semid, pid, j = 10;
1  struct sembuf operacion;
   key_t llave;

   llave = ftok(argv[0], 'U');
```

continúa

```
2  if((semid = semget(llave, 2, IPC_CREAT | 0600)) == -1)
   {
      perror("Error al ejecutar semget");
      exit(-1);
   }
3  semctl(semid, SEMAFORO_HIJO , SETVAL, 0);
   semctl(semid, SEMAFORO_PADRE, SETVAL, 1);

   /* Se crea el proceso hijo */
 if((pid = fork()) == -1)
   {
      perror("Error al ejecutar fork");
      exit(-1);
   }
   else if (pid == 0)
   {
      /* Código correspondiente al proceso hijo */
      while(j)
      {
         /* Se realiza la operación DOWN en el semáforo del
proceso hijo */
         operacion.sem_flg = 0;
         operacion.sem_op = -1;
         operacion.sem_num = SEMAFORO_HIJO ;
4        semop(semid, &operacion, 1);

         printf("SOY EL PROCESO HIJO. IMPRESION: %d\n", j--);
         /* Se realiza la operación UP en el semáforo del
proceso padre */
         operacion.sem_op = 1;
         operacion.sem_num = SEMAFORO_PADRE;
         semop(semid, &operacion, 1);
      }
      /* Borramos el semáforo */
      semctl(semid, 0, IPC_RMID, 0);
   }
   else
   {
      /* Código correspondiente al proceso padre */
      while(j)
      {
         /* Se realiza la operación DOWN en el semáforo del
proceso padre */
         operacion.sem_flg = 0;
         operacion.sem_op = -1;
         operacion.sem_num = SEMAFORO_PADRE;

         semop(semid, &operacion, 1);

         printf("SOY EL PROCESO PADRE. IMPRESION: %d\n", j--);

         /*Se realiza la operación UP en el semáforo del
proceso hijo */
         operacion.sem_op = 1;
         operacion.sem_num = SEMAFORO_HIJO ;
         semop(semid, &operacion, 1);
      }

      /* Borramos el semáforo */
      semctl(semid, 0, IPC_RMID, 0);
   }
}
```

Línea 1 La estructura `sembuf` tiene la forma:

```
struct sembuf
{
   ushort sem_num;
   short sem_op;
   short sem_flg;
};
```

Donde `sem_num` es el número del semáforo cuyo valor se encuentra comprendido entre 0 y N-1. N es el número de semáforos agrupados bajo el mismo identificador. El valor numérico de `sem_op` indica la operación que se va a realizar sobre el semáforo, a saber -1 indica un decremento (DOWN) y 1 indica un incremento (UP). Por último `sem_flg` es una máscara de *bits*.

Pregunta 7.1 Imprima el valor de los miembros de la variable que se declaró como estructura `sembuf` antes de llamar a `ftok`. ¿Qué valores iniciales tiene? Cuando se modifican sus valores, ¿qué función hace uso de dicha variable?

Línea 2 `semget` crea un conjunto de semáforos a los cuales se va a poder acceder mediante el identificador devuelto por `semget`. El segundo parámetro indica el número de semáforos agrupados bajo el mismo identificador. El tercer parámetro indica que el semáforo se creará aun cuando otro proceso lo haya creado antes.

Línea 3 `semctl` permite acceder a la información de control para un semáforo. El primer parámetro es el identificador del semáforo, el segundo es el número de semáforo al que se quiere acceder, el tercero permite inicializar el valor del semáforo al que se especifique en el cuarto parámetro.

Pregunta 7.2 ¿Continúa funcionando el programa si se declara un conjunto de 5 semáforos en la línea `semget`? ¿Qué ocurre si se inicializan los semáforos en ceros? ¿Qué ocurre si se inicializa el semáforo hijo en uno y el semáforo padre en cero?

Línea 4 `semop` realiza operaciones atómicas sobre los semáforos. El segundo parámetro es un apuntador a un arreglo de estructuras del tipo `sembuf` mencionada anteriormente. El tercer parámetro es el número total de elementos en el arreglo de estructuras.

Pregunta 7.3 Imprima los miembros `sem_num` y `sem_op` de la variable `operacion` exactamente antes de llamar a `semop`. Explique brevemente cómo se está logrando sincronizar a los dos procesos.

Ejercicio 7.1 Analice cuidadosamente el código para la sincronización de procesos y elabore dos programas independientes que sincronicen sus impresiones del mismo modo que lo hacen el padre y el hijo en el programa actual.

Ejercicio 7.2 Modifique el programa 7-2 para que se sincronicen tres procesos independientes.