PRÁCTICA 3
UTILIZACIÓN DE LOS SEMÁFOROS

Introducción

Las llamadas al sistema de semáforos permiten a los procesos sincronizar la ejecución realizando un conjunto de operaciones atómicas sobre un conjunto de semáforos. Gracias a esta sincronización, los procesos pueden cooperar correcta y eficientemente sobre recursos no compartibles que estén utilizando conjuntamente.

En esta práctica se utilizarán las llamadas al sistema que nos permiten trabajar con semáforos.

Conceptos básicos

Dijkstra publicó el algoritmo de Dekker que describe una implementación de los semáforos según la cual eran objetos con valores enteros que tienen dos operaciones atómicas: P y V (en alguna bibliografía y en la teoría de la asignatura, a estas operaciones se les conoce como *wait* y *signal*, pero en el entorno Unix existen dos llamadas al sistema con estos mismos nombres y no tienen el mismo significado, por lo que se utilizará P y V). La operación P decrementa el valor de un semáforo si su valor es mayor que 0; la operación V incrementa su valor. Ya que las operaciones son atómicas, no se puede realizar más de una operación P y V a la vez sobre un semáforo. Las llamadas al sistema de semáforos en Unix son una generalización de las operaciones P y V de Dijkstra, en el que varias operaciones se pueden realizar simultáneamente y el valor del semáforo puede ser mayor que 1 (semáforos generales). El kernel realiza todas las operaciones atómicamente, por lo que ningún otro proceso puede ajustar los valores de los semáforos hasta que se hayan completado todas las operaciones.

Las llamadas al sistema que permiten manipular los semáforos son:

- **semget** Crea y accede a un conjunto de semáforos.
- **semctl** Permite realizar varias operaciones de control sobre el conjunto de semáforos.
- **semop** Manipula los valores de los semáforos.

La llamada al sistema *semget* crea una lista de semáforos (también se habla de conjunto de semáforos) o devuelve un identificador (ID) a una lista si ésta ya existía. Su sintaxis es la siguiente:

```
int semget (key, nsem, semflg);
    key_t key;
    int nsem, semflg;
```

La forma de obtener un ID al conjunto de semáforos es mediante una *clave* o *key* (número entero sin signo) que será igual para todos los procesos que trabajen con el mismo conjunto. En la figura 1 se puede apreciar cómo se enlazan todas las listas de semáforos existentes a una *Tabla de semáforos* (*Semaphore Table*). El ID que devuelve *semget* se utiliza en la Tabla de semáforos para localizar una determinada lista.

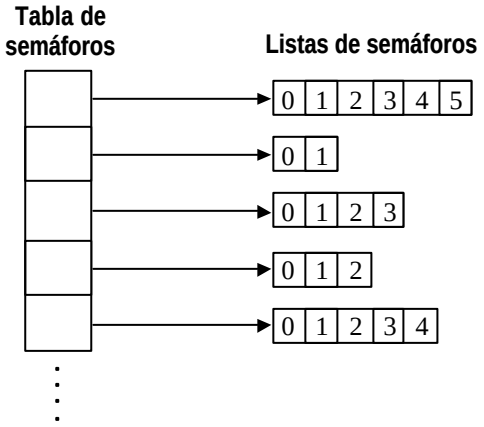

Figura 1. Representación de la tabla de semáforos

Si el bit IPC_CREAT del parámetro *semflg* está activo, el conjunto se creará a no ser que éste existiera ya (examinar el programa 14.1 al final del guión para ver cómo se emplea). También se utiliza el parámetro *semflg* para indicar los permisos de acceso al conjunto de semáforos para el propietario, grupo y otros (como ocurre con cualquier archivo en

Unix). Habrá *nsem* semáforos en el conjunto, numerados a partir de 0. El valor que devuelve (lo llamaremos *semid*) es de tipo entero y corresponde al identificador del conjunto de semáforos. En caso de error, devuelve –1.

Los procesos manipulan los semáforos con la llamada al sistema *semop*. Su sintaxis es:

```
int semop (semid, semops, nsemops);
    int semid;
    struct sembuf *semops;
    int nsemops;
```

La función *semop* se utiliza para realizar atómicamente una lista de operaciones de semáforos sobre el conjunto de semáforos asociados con el identificador del conjunto de semáforos *semid*. El parámetro *semops* es un puntero a la lista de estructuras de operación de semáforos. El contenido de cada estructura es el siguiente:

```
struct sembuf {
    unsigned short sem_num;     /* número de semáforo dentro de la lista */
    short sem_op;               /* operación de semáforos */
    short sem_flg;              /* opciones de operación */
};
```

El parámetro *nsemops* representa el tamaño de la lista. El valor de tipo entero que devuelve la función es el valor del último semáforo operado en el conjunto antes de hacer la operación.

El kernel lee la lista de operaciones de semáforos, *semops*, desde el espacio de direccionamiento del usuario y verifica que los números de los semáforos son legales y que el proceso tiene los permisos necesarios para leer o cambiar los semáforos. En caso contrario, la llamada al sistema fallará.

Podrían ocurrir situaciones peligrosas si un proceso realiza una operación sobre un semáforo, presumiblemente apoderándose o cerrando (locking) algún recurso, y después salir (exit) sin restablecer el valor del semáforo. Estas situaciones pueden ser consecuencia del resultado de un error del programador u otra circunstancia que provoque la terminación de un proceso. Para evitar tales problemas, un proceso puede establecer la opción SEM_UNDO en la llamada al sistema *semop*; así, cuando éste sale, el kernel invierte (reverse) el efecto de cada operación sobre el semáforo que el proceso haya realizado. El efecto neto es dejar el semáforo con el valor con que se inició.

La llamada al sistema *semctl* permite realizar operaciones de control sobre los semáforos. Su sintaxis es:

```
int semctl (semid, semnum, cmd, arg);
    int semid, semnum, cmd;
    union semun arg;
```

Las operaciones a realizar se especifican con *cmd* y afectan bien a un semáforo o bien a todos los semáforos que forman parte del conjunto de semáforos especificados con *semid*. En caso de que el comando vaya dirigido a un semáforo, éste estará identificado con *semnum*; y si va dirigido a todos, *semnum* especificará el número de semáforos sobre los que se va a actuar. El parámetro *arg* se declara como una unión de la forma siguiente:

```
union semun {
    int val;
    struct semid_ds *semstat;
    unsigned short *array;
} arg;
```

El kernel interpreta *arg* según el valor de *cmd*.

Algunas de las órdenes que se pueden especificar con *cmd* son:

- **GETVAL** Devuelve el valor del semáforo
- **SETVAL** Establece el valor del semáforo a *arg.val*
- **GETALL** Obtiene los valores de los semáforos y los deposita en la lista apuntada por *arg.array*
- **SETALL** Establece los valores de los semáforos a partir de la información obtenida de la lista apuntada por *arg.array*
- **IPC_RMID** Elimina del sistema el identificador de un conjunto de semáforos especificado por *semid* y destruye el conjunto de semáforos y estructuras de datos asociados con éste

En el programa 1 podemos ver un ejemplo de cómo se pueden utilizar las llamadas al sistema de semáforos. Los pasos que realiza son crear un conjunto de dos semáforos, hacer una operación P sobre uno de ellos y una operación V sobre el otro y tras visualizar sus contenidos por pantalla, se elimina el conjunto de semáforos. La clave o ID al conjunto de semáforos se define en este ejemplo con el valor 75. Esta clave será utilizada sólo por aquellos procesos que accedan a los mismos semáforos.

Ejercicio

Realizar un programa que implemente el problema del productor-consumidor. Para ello, se utilizará un buffer ficticio (los elementos realmente no se almacenarán puesto que para ello los proceso deberían compartir memoria y no se ha visto cómo hacerlo). Los procesos productor y consumidor se ejecutarán concurrentemente (se puede implementar creando uno de los procesos con una llamada a la función fork o como dos programas independientes).

Programa 1. Ejemplo de utilización de semáforos

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

#define SEMKEY 75

main ( )
{
    /* Declaración de variables */
    int semid;      /* ID de la lista de semáforos */
    struct sembuf sem_oper;          /* Para operaciones P y V sobre semáforos */
    union semun {
          int val;
          struct semid_ds *semstat;
          unsigned short *array;
    } arg;

    /* Creamos una lista con dos semáforos */
    semid = semget (SEMKEY, 2, 0777 | IPC_CREAT);

    /* Inicializamos los semáforos */
    arg.array = (unsigned short * ) malloc (sizeof (short)*2);
    arg.array [0] = arg.array [1] = 1;
    semctl (semid, 2, SETALL, arg);

    /* Operamos sobre los semáforos */
    sem_oper.sem_num = 0;          /* Actuamos sobre el semáforo 0 de la lista */
    sem_oper.sem_op = -1;          /* Decrementar en 1 el valor del semáforo */
    sem_oper.sem_flg = SEM_UNDO;       /* Para evitar interbloqueos si un proceso acaba inesperadamente */
    semop (semid, &sem_oper, 1);

    sem_oper.sem_num = 1;          /* Actuamos sobre el semáforo 1 de la lista */
    sem_oper.sem_op = 1; /* Incrementar en 1 el valor del semáforo */
    sem_oper.sem_flg = SEM_UNDO;       /* No es necesario porque ya se ha hecho anteriormente */
    semop (semid, &sem_oper, 1);

    /* Veamos los valores de los semáforos */
    semctl (semid, 2, GETALL, arg);
    printf ("Los valores de los semáforos son %d y %d", arg.array [0], arg.array [1]);

    /* Eliminar la lista de semáforos */
    semctl (semid, 2, IPC_RMID, 0);

}       /* fin de la función main */
```