PRÁCTICA 5

MEMORIA COMPARTIDA

Introducción

Las versiones modernas de UNIX proporcionan un sistema de memoria virtual. El espacio de direccionamiento de cada proceso está compuesto por una lista de páginas de memoria cada una de las cuales puede ser asignada y manipulada independientemente.

La memoria compartida (Shared Memory) es una forma implícita de *comunicación entre procesos (IPC)*. Es, por tanto, un método altamente eficiente para compartir información entre procesos.

En esta práctica se verán las llamadas al sistema que permitirán compartir información entre varios procesos en el Unix System V.

Conceptos básicos

Los procesos pueden comunicarse directamente entre sí compartiendo partes de su espacio de direccionamiento virtual, por lo que podrán leer y/o escribir datos en la memoria compartida. Para conseguirlo, se crea una región o segmento fuera del espacio de direccionamiento de un proceso y cada proceso que necesite acceder a dicha región, la incluirá como parte de su espacio de direccionamiento virtual.

El sistema permite que existan varias regiones compartibles, cada una compartida por un subconjunto de procesos. Además, cada proceso puede acceder a varias regiones.

Para trabajar con memoria compartida se utilizan básicamente las siguientes llamadas al sistema:

- shmget Crea una nueva región de memoria compartida o devuelve una existente.
- shmat Une lógicamente una región al espacio de direccionamiento virtual de un proceso.
- shmdt Separa una región del espacio de direccionamiento virtual de un proceso.
- shmctl Manipula varios parámetros asociados con la memoria compartida.

Para leer y escribir en la memoria compartida, un proceso utiliza las mismas instrucciones que en el caso de memoria no compartida, pero debe haber incluido la región en su espacio de direccionamiento.

Para la gestión de la memoria compartida, el núcleo (kernel) utiliza tres tablas:

- *Tabla de memoria compartida*: Contiene una entrada por cada región distinta que se esté compartiendo en el sistema.
- *Tabla de regiones*: Contiene una entrada por cada región.
- *Tabla del proceso (o tabla de regiones por proceso)*: Contiene las entradas correspondientes a las regiones que forman el espacio de direccionamiento del proceso. Habrá una tabla del proceso por cada proceso que se encuentre en el sistema.

La sintaxis de shmget es la siguiente:

```
int shmget (key, size, shmflg)
    key_t key;
    int size, shmflg;
```

donde *size* es el número de bytes de la región. El kernel busca en la tabla de memoria compartida por la clave (*key*) indicada. La clave será un valor entero único para todos los procesos que compartan la misma región de memoria. Si existe una entrada y los permisos lo permiten, devuelve un identificador a la región (*shmid*). Si no lo encuentra y se ha indicado la opción IPC_CREAT en shmflg, el núcleo creará una nueva región. Otras opciones que se pueden indicar en el parámetro shmflg son los permisos de la región (se establecen de la misma forma que se hacía en el caso de la creación de una lista de semáforos).

En caso de que la función genere un error, devolverá -1. La región de memoria quedará reservada y sólo se asignará al espacio de direccionamiento de los procesos (tablas de páginas) cuando éstos se unan a la región mediante la ejecución de la llamada al sistema *shmat*. La sintaxis de esta llamada es la siguiente:

```
char *shmat (shmid, addr, shmflg)
    int shmid;
    char *addr;
    int shmflg;
```

donde *shmid* (valor que devuelve la función shmget) identifica a la región de memoria compartida, *addr* es la dirección virtual donde el usuario quiere unir la memoria compartida y *shmflg* especifica los permisos de la región. El valor que devuelve es la dirección virtual donde el núcleo ha unido la región. En caso de error, devuelve –1.

Cuando se ejecuta *shmat*, el núcleo verifica que el proceso tiene los permisos necesarios para acceder a la región. Si la dirección especificada por el proceso es 0, el núcleo elige una dirección virtual conveniente.

Para separar una región de memoria de su espacio de direccionamiento virtual, los procesos utilizan la llamada al sistema *shmdt*. Su sintaxis es la siguiente:

```
int shmdt (addr)
    char *addr;
```

donde *addr* es la dirección que devuelve la llamada *shmat*.

La llamada al sistema *shmctl* permite a un proceso buscar el estado y establecer los parámetros de la región de memoria compartida. Su sintaxis es:

```
shmctl (shmid, cmd, shmstatbuf)
    int shmid, cmd;
    struct shmid_ds *shmstatbuf;
```

donde *shmid* identifica a la entrada correspondiente en la tabla de memoria compartida, *cmd* indica el tipo de operación a realizar y *shmstatbuf* indica la dirección de una estructura de datos a nivel de usuario que contiene la información de estado de la entrada de la tabla de memoria compartida (sólo para las operaciones de lectura o definición del estado). La sintaxis de esta estructura es:

```
struct shmid_ds {
    struct ipc_perm shm_perm;      /* permisos */
    int shm_segsz;          /* Tamaño del segmento */
    ushort shm_cpid;        /* PID del creador */
    …
};
```

Algunas operaciones que se pueden indicar en el parámetro *cmd* de la llamada *shmctl* son:

- IPC_STAT Sitúa el valor actual de cada elemento de la estructura de datos asociada con *shmid* en la estructura apuntada por *shmstatbuf.*
- IPC_SET Establece el valor de los elementos de la estructura de datos asociada con *shmid*, obtenidos a partir de la estructura de datos apuntada por *shmstatbuf.*
- IPC_RMID Elimina del sistema el identificador de memoria compartida especificado en *shmid*.

Los programas 1 y 2 muestran un ejemplo de compartición de una región de memoria. En este ejemplo, se puede apreciar una sincronización entre los procesos utilizando una posición de memoria compartida. Para ello, el primer proceso crea la región de memoria compartida e introduce un valor distinto de 0 en la posición de memoria a compartir. El segundo proceso se sincroniza con el primero esperando a que el primero actualice el valor para poder continuar su ejecución.

Programa 1

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdlib.h>

#define SHMKEY 75

main ()
{
    int shmid;
    char *addr, *pint;

    /* Crear la región de memoria y obtener la dirección */
    shmid = shmget (SHMKEY, 1, 0777 | IPC_CREAT);

    /* Enlazar región al espacio de direccionamiento del proceso */
    addr = shmat (shmid, 0, 0);
    pint = (char *) addr;    /* Reservar addr */

    /* Código de la sincronización */
    sleep (10);    /* Para dar tiempo a que se ejecute el segundo proceso */
    (*pint)++;
```

```
    /* Separar la región del espacio de direccionamiento del proceso */
    shmdt (addr);
    /* Eliminar la región de memoria compartida */
    shmctl (shmid, IPC_RMID, 0); /* Esta operación también la puede realizar otro proceso */
    return 0;
} /* fin de la función main */
```

Programa 2

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define SHMKEY 75
main ()
{
    int shmid;
    char *addr, *pint;
    /* Obtener dirección de la región de memoria */
    shmid = shmget (SHMKEY, 1, 0777);
    /* Enlazar región al espacio de direccionamiento del proceso */
    addr = shmat (shmid, 0, 0);
    pint = (char *) addr;    /* Reservar addr */
    /* Espera a que el contenido de la primera dirección tenga un valor distinto de 0 */
    while (*pint == 0)
        sleep(1);           /* Esperar 1 segundo */
    printf ("\n Continuo la ejecución \n");
    /* Separar la región del espacio de direccionamiento del proceso */
    shmdt (addr);
    return 0;
} /* fin de la función main */
```

Ejercicio

Modificar los programas implementados en la Práctica 3 (Utilización de los semáforos) para que los procesos productor y consumidor utilicen un búfer real de memoria (según el planteamiento del problema, el búfer debe ser circular). Para ello, se debe crear una región de memoria compartida que alojará el búfer en el que el productor escribirá los elementos que produzca (por ejemplo, letras en orden alfabético) y del que el consumidor extraerá dichas letras mostrándolas en pantalla. La sincronización entre los procesos debe seguir realizándose a través de semáforos.

Información complementaria

Cualquier duda con alguna de las funciones utilizadas en esta práctica se puede consultar en la ayuda en línea (*man*).