Lenguaje C e IPC. Región Crítica

Conceptos básicos

Los procesos que se ejecutan de forma concurrente en un sistema se pueden clasificar como procesos independientes o colaborantes. Un proceso independiente es aquel que se ejecuta sin requerir la ayuda o colaboración de otros procesos. Los procesos son colaborantes cuando están diseñados para trabajar conjuntamente en alguna tarea, para lo que deben ser capaces de comunicarse e interactuar entre ellos. Tanto si los procesos son independientes como colaborantes, pueden producirse una serie de interacciones entre ellos. Estas interacciones pueden ser de dos tipos:

1. Interacciones motivadas porque los procesos comparten o compiten por el acceso a recursos físicos o lógicos. Por ejemplo, varios procesos quieren acceder al disco al mismo tiempo.
2. Interacción motivada porque los procesos se comunican y sincronizan entre sí para alcanzar un objetivo común. Por ejemplo, un compilador se puede construir mediante dos procesos: el compilador propiamente dicho, que se encarga de generar código ensamblador, y el proceso ensamblador, que obtiene código en lenguaje máquina a partir del ensamblador. En este ejemplo puede apreciarse la **necesidad de comunicar y sincronizar** a los dos procesos.

Estos dos tipos de interacciones obligan al sistema operativo a incluir mecanismo y servicios que permitan la comunicación y la sincronización entre procesos.

IPC – Comunicación entre Procesos

La comunicación entre procesos permite que 2 o más procesos colaboren para alcanzar un objetivo en común.

Para que los procesos se comuniquen entre sí, se utilizan los recursos que provee el SO (Sistema Operativo). Desde la programación de código en lenguaje C, se accede a esos recursos a través de llamadas al sistema.

Las llamadas al sistema, o en inglés *system call*, es el mecanismo usado por una aplicación para solicitar un servicio al sistema operativo. Las llamadas al sistema comúnmente usan una instrucción que causa que el procesador transfiera el control al núcleo del SO. Cuando

una llamada al sistema es invocada, la ejecución del programa que invoca es interrumpida y sus datos son guardados para poder continuar ejecutándose luego. El procesador entonces comienza a ejecutar las instrucciones de código de bajo nivel de privilegio, para realizar la tarea requerida. Cuando esta finaliza, se retorna al proceso original, y continúa su ejecución. El retorno al proceso depende del tiempo de ejecución de la llamada al sistema y del algoritmo de planificación de CPU.

Mecanismos de comunicación entre procesos (IPC)

- Archivos.
- Memoria compartida.
- Cola de mensajes.

Recursos de comunicación entre procesos (IPC)

- Semáforos.
- Cola de mensajes.
- Memoria compartida.

Biblioteca

Generalmente, los sistemas operativos proveen bibliotecas que relacionan los programas de usuario y el resto del sistema operativo, usualmente una biblioteca en tiempo de ejecución de C como la glibc de GNU. Esta biblioteca maneja, entre otras cosas, los detalles de bajo nivel para transferir información al *kernel*, esto reduce la dependencia entre el sistema operativo y la aplicación y el software, e incrementa su portabilidad.

Librerías

Generalmente cuando se desarrolla código en lenguaje C, se utilizan librerías y se las incluye en el código mediante la directiva #include. La librería que se utiliza para la comunicación entre procesos es <ipc.h>, y la misma contiene la llamada al sistema ftok():

```
#include<sys/ipc.h>

key_tftok(char*,int)
```

La función ftok utiliza la identidad del fichero que indica pathname (que debe referirse a un fichero existente y accesible) y los 8 bits menos significativos de proj_id (que debe ser distinto de cero) para generar una clave IPC de System V de tipo key_t , adecuada para el uso con las llamadas al sistema de los recursos del IPC como ser la cola de mensajes (msgget()), el semáforo (semget()), o la memoria compartida (shmget()), el valor resultante es el mismo para todos los nombres de ruta que hacen referencia al mismo fichero, cuando se utiliza el mismo valor de proj_id. El valor devuelto debería ser diferente cuando los ficheros (que existen simultáneamente) o los identificadores de proyecto son distintos.

Sección critica

Sección crítica: porción de código con variables compartidas.

Éste es uno de los problemas que con mayor frecuencia aparece cuando se ejecutan procesos concurrentes tanto si son colaborativos como independientes.

Si se considera un sistema compuesto por *n* procesos {P1, P2, ..., PN} en el que cada uno tiene un fragmento de código, que se denomina sección crítica. Dentro de la sección crítica, los procesos pueden estar accediendo y modificando variables comunes, registros de una base de datos, un archivo, en general cualquier recurso compartido.

La característica más importante de este sistema es que cuando un proceso se encuentra ejecutando código de la sección crítica, **ningún otro proceso puede ejecutar en su sección.**

Para resolver el problema de la sección crítica es necesario utilizar algún mecanismo desincronización que permita a los procesos cooperar entre ellos sin problemas. Este mecanismo debe proteger el código de la sección crítica y su funcionamiento básico es el siguiente:

- Cada proceso debe solicitar permiso para entrar en la sección crítica mediante algún fragmento de código, que se denomina de forma genérica entrada en la sección crítica.
- Cuando un proceso sale de la sección crítica debe indicarlo mediante otro fragmento de código, que se denomina salida de la sección crítica. Este fragmento permitirá que otros procesos entren a ejecutar el código de la sección crítica.

La estructura general, por tanto, de cualquier mecanismo que pretenda resolver el problema de la sección crítica es la siguiente:

Entrada en la sección crítica→Código de la sección crítica→Salida de la sección crítica

Cualquier solución que se utilice para resolver este problema debe cumplir los tres requisitos siguientes:

Exclusión mutua

Si un proceso está ejecutando código de la sección crítica, ningún otro proceso lo podrá hacer.

Progreso

Si ningún proceso está ejecutando dentro de la sección crítica, la decisión de qué proceso entra en la sección se hará sobre los procesos que desean entrar. Los procesos que no quieren entrar no pueden formar parte de esta decisión. Además, esta decisión debe realizarse en tiempo finito.

Espera acotada

Debe haber un límite en el número de veces que se permite que los demás procesos entren a ejecutar código de la sección crítica después de que un proceso haya efectuado una solicitud de entrada y antes de que se conceda la suya.

Llamadas al sistema

Librería comunicación entre procesos: “sys/ipc.h”

Llamadas al sistema:

- ftok

Librería cola de mensajes: “sys/msg.h”

Llamadas al sistema:

- msgctl
- msgget
- msgrcv
- msgsnd

Librería memoria compartida: “sys/shm.h”

Llamadas al sistema:

- shmat
- shmctl
- shmdt
- shmget

Librería semáforos: “sys/sem.h”

Llamadas al sistema:

- semctl
- semget
- semop

Código ejemplo

```
#include <sys/ipc.h>

key_t creo_clave()
{
// Igual que en cualquier recurso compartido (memoria compartida, semaforos
// o colas) se obtien una clave a partir de un fichero existente cualquiera
// y de un entero cualquiera. Todos los procesos que quieran compartir este
    // semáforo, deben usar el mismo fichero y el mismo entero.
    key_t clave;
    clave = ftok ("/bin/ls", 33);
    if (clave == (key_t)-1)
    {
        printf("No puedo conseguir clave semáforo, mem compartida, etc.\n");
        exit(0);
    }
    return clave;
}
```

La función de ejemplo retorna una clave generada con la llamada al sistema ftok, pasándole como parámetro un comando conocido “/bin/ls” y un valor entero “33”. La clave la genera

haciendo un HASH (es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija) entre ambos parámetros.

La clave permitirá acceder desde varios procesos al mismo recurso compartido (semáforos, cola de mensajes, memoria compartida).