TP A

Sistemas Operativos

Semáforos UNIX IPC System V

**Compilación del código**

En la versión de Ubuntu, no se encuentra el compilador del lenguaje C, por lo que habrá que instalar el mismo, por medio del comando *sudo apt install gcc .*

Con el comando gcc ARCHIVO -o COMPILADO, ejecutaremos la compilación siendo ARCHIVO el código en C y COMPILADO el nombre de salida del código compilado

*gcc usem.c -o usem*

Al compilar el código lanza los siguientes errores:

**Errores**

Procedimiento de purga de errores, se compilará el código, se realizará la modificación y se re compilara, para observar si se produce un nuevo error o se soluciona el error.

*usem.c: In function ‘crear\_sem’:*

*usem.c:80:17: error: storage size of ‘semopciones’ isn’t known*

*union semun semopciones;*

*^~~~~~~~~~~*

Este error nos informa que no se conoce el tamaño de semopciones, observamos que utiliza semun, y revisando el paper y el código, que no se encuentra definida la estructura semun. Por lo que agregamos la misma, previo a la definición de los procedimientos

/\* arg para la llamada del sistema semctl\*/

union semun {

int val;/\* valor para SETVAL\*/

struct semid\_ds \*buf;/\* buffer para IPC\_STAT e IPC\_SET\*/

ushort\*array; /\* array para GETALL y SETALL\*/

struct seminfo \*\_\_buf; /\* buffer para IPC\_INFO\*/

void \*\_\_pad;

};

usem.c:81:19: error: ‘SEMMSL’ undeclared (first use in this function); did you mean ‘SETVAL’?

if(cantidad > SEMMSL)

^~~~~~

SETVAL

Este error nos informa que SEMMSL, no está definido y nos sugiere que tal vez nos equivocamos de constante y nos referimos a SETVAL, pero como se trata de constantes distintas agregaremos todas las constantes que se mencionan en el paper

#define SEMMNI 128 /\* máximo número de identificadores de conjuntos\*/

#define SEMMSL 32 /\* máximo número de semáforos por identificador\*/

#define SEMOPM 32 /\* máximo número de operaciones por llamada semop\*/

#define SEMVMX 32767 /\* máximo valor por semáforo\*/

usem.c:81:19: note: each undeclared identifier is reported only once for each function it appears in

usem.c: In function ‘zero\_espera\_sem’:

usem.c:120:51: error: ‘errno’ undeclared (first use in this function)

fprintf(stderr, "Valor de ERRNO : %d \n", errno);

^~~~~

El error nos muestra que errno no esta declarada, investigando, encontré que es necesaria la librería errno.h, por lo que la incluimos

#include <errno.h> /\*Libreria agregadad\*/

Una vez agregada la librería, la compilación es completada exitosamente

**Testeo del código**

Probamos la ejecución de usem con las distintas variantes, por medio de la ejecución por consola.

*./usem* nos ejecutara el main mostrando las distintas opciones que se le pueden pasar

*./usem c* Comando principal, que crea los semáforos

*./usem t* Comando que ejecuta una espera, dejando la terminal en espera del ENTER, como este comando no está pensado para ser llamado por fuera de un script shell, no notifica como es que se podrá salir de la espera.

*./usem e* Comando que espera que el semáforo este ocupado.

Este comando lanza un error

Proceso a la ESPERA de valor CERO en semáforo IPC...

La Espera NO pudo establecerse

Valor de ERRNO : 22

Buscamos en la documentación de errno[[1]](#footnote-1), y observamos que se trata de un error EINVAL, es decir, cantidad de argumentos inválidos, revisando y comparando con los otros llamados note que en este procedimiento le falta la apertura de semáforos, por lo que agregamos la instrucción a esta opción.

case 'e': *abrir\_sem(&semset\_id, clave);*

zero\_espera\_sem(semset\_id, INI);

break;

Recompilamos el código y volvemos a probar el procedimiento, esta vez espera que el semáforo llegue a cero

*./usem b* Elimina el conjunto de semáforos

*./usem a* Bloquea el semáforo, y espera que se ingrese un texto para ser almacenado en un archivo, este comando, necesitara más parámetros el nombre del archivo y el texto

*./usem a archivo.dat TEXTO* donde TEXTO es una cadena de caracteres sin espacios, que se almacenara en el archivo

**Scripts en shell**

Utilizaremos los scripts shell, que se presentaron junto al código, y agregaremos dos nuevos scripts para la espera, que alguien bloquee el semáforo, y la eliminación del semáforo.

A partir de este punto utilizaremos de 2 a 3 terminales para las pruebas, las llamaremos A, B y C.

Debemos ejecutar primero el script de creación del conjunto de semáforos, *crear-sem.sh*, lo haremos en A, luego podemos ejecutar los otros scripts.

Comenzaremos ejecutando *zero.sh*, en B, lo que nos bloquear esta terminal, hasta que se bloquee a un semáforo, volvemos a A y ejecutamos *toma-sem.sh* o *trabajo.sh*, el primero ejecutara la toma y esperar al ENTER para continuar y el segundo esperara un texto para guardar en un archivo, al ejecutar esos script la B será liberada.

En este caso ejecutamos *toma-sem.sh*, por lo que A estará bloqueando el semáforo hasta recibir el ENTER, ahora en B ejecutaremos *trabajo.sh*, e ingresaremos el texto a grabar, lo que no concluirá hasta liberar a A, que es el proceso que bloque el semáforo, por lo que para que el texto sea grabado iremos a A y pulsaremos ENTER, una vez realizara esta acción B concluirá su ejecución*.*

Ahora probaremos ejecutar dos tomas en A y C, y un trabajo en B, el orden de ejecución es A,B y C, es decir, se bloqueara un semáforo, se intentara escribir el archivo y antes de hacer el grabado ejecutaremos una nueva toma*,* ahora trataremos de hacer la grabación sin liberar el semáforo, como esperamos el archivo no es grabado, por lo que liberaremos a A, y observamos que el archivo aún no se actualiza y B no es liberada, por lo que debemos liberar a C, una vez realizado esto el archivo es grabado. Esto ocurre debido a que como solo hay un semáforo, es decir un solo recurso disponible, no importará el orden de bloqueo, hasta que todos lo liberen no se tendrá el recurso disponible*.*

Cambiaremos la ejecución a que A y C realicen el grabado y B el bloqueo, es decir se bloqueara para grabar, luego se bloqueara y se intentara grabar, es esta situación al enviar a grabar, no se realiza el mismo, sin importar que el bloqueo para el grabado se realizara antes que la *toma*. Por lo que siempre que exista una toma y solo un semáforo, el orden de ejecución no cambiara el resultado.

Por último eliminaremos este conjunto de semáforos con *borrar.sh.*

**Conclusión**

La utilización de semáforos, permitirá dar al sistema, seguridad en cuanto a la concurrencia e integridad sobre sus recursos.

Como observamos en las pruebas, con un el semáforo si se marca como ocupado, y existen nuevas tomas, los recursos no se liberarán hasta que todos los procesos de toma, no importa en qué momento llegue, hayan concluido, con esto podemos notar el problema que genera esta simulación, ya que a pesar de que el proceso de trabajo llegue luego de la primera toma, no concluirá hasta que todos los procesos de toma que arribaron luego de que el comenzó a trabajar no hayan concluido. Lo que provocará una espera que será infinita si el proceso de liberación nunca se ejecuta.

**Códigos**

Código C

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* - usem.c - Utilitario básico para semáforos IPC System V -

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <ctype.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#include <errno.h> /\*Libreria agregadada\*/

#define SEM\_MAX\_RECURSO 1 /\* Valor inicial de todos los semáforos \*/

#define INI 0 /\* Índice del primer semáforo \*/

/\*Constantes agregadas\*/

#define SEMMNI 128 /\* máximo número de identificadores de conjuntos\*/

#define SEMMSL 32 /\* máximo número de semáforos por identificador\*/

#define SEMOPM 32 /\* máximo número de operaciones por llamada semop\*/

#define SEMVMX 32767 /\* máximo valor por semáforo\*/

void abrir\_sem(int \*sid, key\_t clave);

void crear\_sem(int \*sid, key\_t clave, int idx);

void bloquear\_sem(int clave, int idx);

void zero\_espera\_sem(int clave, int idx);

void desbloquear\_sem(int clave, int idx);

void remover\_sem(int sid);

void uso(void);

/\*Estructura semun agregada\*/

/\* arg para la llamada del sistema semctl\*/

union semun {

    int val;/\* valor para SETVAL\*/

    struct semid\_ds \*buf;/\* buffer para IPC\_STAT e IPC\_SET\*/

    ushort\*array; /\* array para GETALL y SETALL\*/

    struct seminfo \*\_\_buf; /\* buffer para IPC\_INFO\*/

    void \*\_\_pad;

};

int main(int argc, char \*argv[])

{

    key\_t clave;

    int semset\_id;

    FILE \*fptr;

    if(argc == 1)

        uso();

    /\* Crea una clave única mediante ftok() \*/

    clave = ftok(".", 'u');

    switch(tolower(argv[1][0]))

        {

        case 'c':

            if(argc != 2)

                uso();

            crear\_sem(&semset\_id, clave, 1);

            break;

        case 't': abrir\_sem(&semset\_id, clave);

            bloquear\_sem(semset\_id, INI);

            getchar();

            desbloquear\_sem(semset\_id, INI);

            break;

        case 'e': abrir\_sem(&semset\_id, clave);

             zero\_espera\_sem(semset\_id, INI);

            break;

        case 'b': abrir\_sem(&semset\_id, clave);

            remover\_sem(semset\_id);

            break;

        case 'a': abrir\_sem(&semset\_id, clave);

            bloquear\_sem (semset\_id, INI);

            if ((fptr = fopen(argv[2], "a")) == NULL)

                exit (-1);

            else

            {

                fprintf(fptr, "%s\n", argv[3]);

                fclose(fptr);

            }

            desbloquear\_sem (semset\_id, INI);

            break;

        default: uso();

        }

    return(0);

}

void abrir\_sem(int \*sid, key\_t clave)

{

    /\* Abre el conjunto de semáforos\*/

    if((\*sid = semget(clave, 0, 0666)) == -1)

    {

        printf("El conjunto de semáforos NO existe!\n");

        exit(1);

    }

}

void crear\_sem(int \*sid, key\_t clave, int cantidad)

{

    int cntr;

    union semun semopciones;

    if(cantidad > SEMMSL)

    {

        printf("ERROR : cant. máx. de sem. en el conjunto es %d\n", SEMMSL);

        exit(1);

    }

    printf("Creando nuevo conjunto IPC con %d semáforo(s)...\n",cantidad);

    if((\*sid = semget(clave, cantidad, IPC\_CREAT|IPC\_EXCL|0666)) == -1)

    {

        fprintf(stderr, "Ya existe un conjunto con esta clave!\n");

        exit(1);

    }

    printf("Nuevo conjunto IPC de sem. creado con éxito\n");

    semopciones.val = SEM\_MAX\_RECURSO;

    /\* Inicializa todos los semáforos del conjunto \*/

    for(cntr=0; cntr<cantidad; cntr++)

        semctl(\*sid, cntr, SETVAL, semopciones);

}

void bloquear\_sem(int sid, int idx)

{

    struct sembuf sembloqueo={ 0, -1, SEM\_UNDO};

    sembloqueo.sem\_num = idx;

    if((semop(sid, &sembloqueo, 1)) == -1)

    {

        fprintf(stderr, "El Bloqueo falló\n");

        exit(1);

    }

}

void zero\_espera\_sem(int sid, int idx)

{

    struct sembuf esperazero={ 0, 0, SEM\_UNDO};

    esperazero.sem\_num = idx;

    printf("Proceso a la ESPERA de valor CERO en semáforo IPC...\n");

    if((semop(sid, &esperazero, 1)) == -1)

    {

        fprintf(stderr, "La Espera NO pudo establecerse \n");

        fprintf(stderr, "Valor de ERRNO : %d \n", errno);

        exit(1);

    }

    printf("ESPERA concluída. Terminación del proceso.\n");

}

void desbloquear\_sem(int sid, int idx)

{

    struct sembuf semdesbloqueo={ 0, 1, SEM\_UNDO};

    semdesbloqueo.sem\_num = idx;

    /\* Intento de desbloquear el semáforo \*/

    if((semop(sid, &semdesbloqueo, 1)) == -1)

    {

        fprintf(stderr, "El Desbloqueo falló\n");

        exit(1);

    }

}

void remover\_sem(int sid)

{

    semctl(sid, 0, IPC\_RMID, 0);

    printf("Conjunto de semáforos eliminado\n");

}

void uso(void)

{

    fprintf(stderr, " - usem - Utilitario básico para semáforos IPC\n");

    fprintf(stderr, "    USO : usem    (c)rear    \n");

    fprintf(stderr, "    (t)oma recurso compartido    \n");

    fprintf(stderr, "    (e)spera IPC de valor cero    \n");

    fprintf(stderr, "    (b)orrar    \n");

    fprintf(stderr, "    (a)gregar <PATH-DB> <LINE>    \n");

    exit(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* fín de usem.c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-Script Shell

crea-sem.sh

#!/bin/sh

clear

echo "-----------------------------------------------------"

echo " Crea un Conjunto de un único Semáforo IPC System V "

echo "-----------------------------------------------------"

./usem c

echo "------------------< FIN de crea-sem.sh >-------------"

toma-sem.sh

#!/bin/sh

clear

echo "-----------------------------------------------------"

echo " Pulse <ENTER> cuando desee desbloquear el SEMÁFORO "

echo "-----------------------------------------------------"

./usem t

echo "------------------< FIN de toma-sem.sh >-------------"

trabajo.sh

#!/bin/sh

clear

archivo="p-usem.dat"

Narchivo="./$archivo"

echo "-----------------------------------------------------"

echo " 1) Ingrese una línea de texto para probar el código "

echo " 2) Compruebe que el texto que Ud. ingresó figure en "

echo "el archivo : $archivo"

echo "-----------------------------------------------------"

read Param

./usem a $Narchivo $Param

echo "-------------------< Trabajo Realizado >-------------"

scripts agregados

borrar.sh

#!/bin/sh

clear

echo "-----------------------------------------------------"

echo " Elimina un Conjunto de un único Semáforo IPC System V "

echo "-----------------------------------------------------"

./usem b

echo "------------------< FIN de borrar.sh >-------------"

zero.sh

#!/bin/sh

clear

echo "-----------------------------------------------------"

echo " Esperando que se tome el semáforo "

echo "-----------------------------------------------------"

./usem e

echo "------------------< FIN de zero.sh >-------------"

**Referencias**

1

1. <https://man7.org/linux/man-pages/man3/errno.3.html> [↑](#footnote-ref-1)