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 codigo lanza los siguientes errores:

**Errores**

Procedimiento de purga de errores, se compilara el código, se realizara 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 ejecucion 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 en cero. 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 errno1, 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 este 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

**Scritpts en shell**

Utilizaremos los scripts shell, que se presentaron junto al codigo, y agregaremos dos nuevos scrpits para la espera, que alguien bloquee el semaforo, y la eliminacion del semaforo.

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 creacion del conjunto de semaforos, *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 semaforo, 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 scritp la B sera liberada.

En este caso ejecutamos *toma-sem.sh*, por lo que A estara bloquendo el semaforo hasta recibir el ENTER, ahora en B ejecutaremos *trabajo.sh*, e ingresaremos el texto a grabar, lo que no concluira hasta liberar a A, que es el proceso que bloque el semaforo, por lo que para que el texto sea grabado iremos a A y pulsaremos ENTER, una vez realizara esta accion B concluira su ejecucion*.*

Ahora probaremos ejecutar dos tomas en A y C, y un trabajo en B, el orden de ejecucion sera A,B y C, es decir, se bloqueara un semaforo, se intentara escribir el archivo y antes de hacer el grabado ejecutaremos una nueva toma*, ahora trataremos de hacer la grabacion sin liberar el semaforo, como esperamos el archivo no es grabado, por lo que liberaremos a A, y observamos que el archivo aun 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 semaforo, es decir un solo recurso disponible, no importara el orden de bloqueo, hasta que todos los liberen no se tendrá el recurso disponible.*

Cambiaremos la ejecucion 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 situacion al enviar a grabar, no se realiza el mismo, sin importar que el bloqueo para el grabado se realizara antes que el bloque solo. Por lo que siempre que exista una toma y solo un semaforo, el orden de ejecucion no cambiara el resultado.

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

**Conclusión**

**Códigos**

**Script shell**

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**

1https://man7.org/linux/man-pages/man3/errno.3.html