TEC | Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Principios de Sistemas Operativos - IC6600

Aplicación Readers – Writers

Equipo de Proyecto:

Andres Masis Rojas 2020127158

Juan Sebastián Gamboa Botero 2020030303

Leonardo David Fariña Ozamis 2020045272

Nikholas Ocampo Fuentes 2020061243

Profesor:

Erika Marín Schumann

I Semestre del 2023

Fecha: 22 de Mayo, 2023

Estrategia de Solución

El tipo de semáforos que utilizó y ¿por qué?

Se utilizan dos tipos de semáforos, los cuales son de tipo, binario y contadores. Los semáforos binarios se utilizan porque:

- Tienen dos estados posibles: 0 (bloqueado) o 1 (desbloqueado).
- Se utilizan para controlar el acceso a recursos compartidos y para lograr la exclusión mutua.

Con esto en mente, los binarios se utilizan para el manejo de accesos y escrituras en memoria compartida, donde almacenamos datos de los diferentes programas. Más que todo para el programa Pspy y su reporte.

El otro tipo de semáforos que se utilizan son los contadores esto porque:

- Los semáforos contadores tienen un valor entero y pueden tener cualquier valor mayor o igual a cero.
- Se utilizan para controlar el acceso a un número limitado de recursos, como un conjunto de hilos (que es nuestro caso).
- Los hilos o procesos pueden tomar prestado un recurso si el contador es mayor que cero y se bloquearán si el contador es cero.

Con esto en mente, tenemos dos contadores el primero para el acceso a memoria y el segundo para el acceso a los readers egoístas, los contadores se inicializan con los valores 1 para memoria y 3 para los readers egoístas, este valor es la cantidad de procesos que pueden acceder a la memoria compartida al mismo tiempo.

Un punto importante a destacar es que se utilizan las funciones y la estructura de datos de la librería **sys/sem.h** para la creación de los semáforos

Se usan funciones como:

- **semget**(): Crea un conjunto de semáforos o recupera su identificador existente.
- **semctl**(): Controla operaciones en semáforos, como configurar valores iniciales o realizar operaciones específicas.

- **semop**(): Realiza operaciones en semáforos, como bloquear, desbloquear o esperar en un semáforo.

En conclusión, se utilizan semáforos binarios para acceder a la memoria y saber si se va a escribir o leer y los semáforos de tipo contador para llevar un control de la cantidad de procesos que están accediendo a la memoria.

Una explicación de ¿cómo logró la sincronización?

La sincronización se llevó a cabo a través de los semáforos anteriormente explicados. En resumen había un control al acceso de memoria y la cantidad de reader egoístas consecutivos que se pueden ejecutar. Y también se utilizó un semáforo binario para el acceso y modificación de otros recursos en memoria compartida.

Algunos detalles a fondo sobre código relevante

Se definieron 2 llaves, una para acceder a memoria compartida y otra para acceder a los semáforos.

```
1. #define SHM_KEY 1234
2. #define SEM_KEY 9999
```

Se utilizó el siguiente struct para el control de los semáforos.

```
1. union semun {
2.     int val;
3.     struct semid_ds *buf;
4.     unsigned short *array;
5.     struct seminfo *__buf;
6. };
```

Se utilizó el siguiente struct para el control de las acciones a mostrar en el programa espía(Pspy).

```
1. typedef struct SpyNode {
2.     struct SpyNode* next;
3.     long pid;
4.     int type, action;
```

```
} SpyNode;
```

Se utilizó el siguiente struct para el control de los mensajes a guardar en la memoria compartida.

```
typedef struct MSJ{
    long pid;
    char fecha[11]; // "YYYY-MM-DD"
    char hora[9]; // "HH:MM:SS"
    int linea;
    int is;
} MSJ;
```

Se utilizó el siguiente struct para poder pasar parámetros a las funciones de los hilos.

```
typedef struct Settings{
        int sleeping;
        int actor;
        int sem_id;
        MSJ *shared_memory;
} Settings;
```

Las siguientes bibliotecas son las utilizadas en la mayoría de programas.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
// Manejo de cadenas
#include <string.h>
// Tipos de datos
#include <ctype.h>
#include <limits.h>
// Interfaz
#include <ncurses.h>  //ADD AT THE END OF COMPILING THE FILE -lncurses
// Hilos
#include <pthread.h>
// Errores
#include <errno.h>
// Tiempo
#include <time.h>
// Memoria compartida
#include <sys/ipc.h>
#include <sys/shm.h>
// Semáforos
#include <sys/sem.h>
```

Una explicación de Memoria compartida

La memoria compartida se crea, se vincula, se inicializa y se desvincula en el programa Pinit.
La memoria compartida se vincula, se edita y se desvincula en los programas Pwriter, Preader y PreaderEgo.

La memoria compartida se vincula y se desvincula en el programa Pspy.

La memoria compartida se vincula, se desvincula y se borra en el programa Pend.

Análisis de resultados

Después de haber desarrollado el simulador de la sincronización de los reader y writers con los diferentes algoritmos y con semáforos en C utilizando **sys/sem.h**, se realizaron pruebas exhaustivas para analizar su rendimiento.

A continuación se presentan los análisis de resultados según lo indicado en la descripción detallada del proyecto:

1. Programa Inicializador : 100%
2. Programa Write: 100%
3. Mensaje : 100%
4. Programa Reader : 100%
5. Programa Reader Egoísta: 100%
6. Programa Finalizador: 100%
7. Programa Espía : 100%
8. Sincronización de Procesos y Memoria compartida : 100%
9. Bitácora: 100%
10. Documentación : 100 %

¿Qué algoritmo genera más fragmentación?

El programa "Reader Egoísta" puede generar más fragmentación en comparación con otros programas debido a su comportamiento de lectura egoísta(que roba el mensaje) y su gestión de acceso a la memoria compartida.

Ya que esto genera fragmentación interna. Esto sucede cuando un reader egoísta selecciona una entrada en la memoria compartida y la lee, roba el mensaje asociado y lo guarda en su propio espacio de memoria. Esto puede generar fragmentación interna, ya que el espacio de memoria asignado a la entrada original ahora está parcialmente vacío. A medida que más procesos egoístas realizan lecturas y roban mensajes, la memoria compartida puede llenarse de espacios vacíos dispersos, lo que resulta en una fragmentación interna considerable.

Lecciones aprendidas

Este proyecto aportó beneficios a la formación como ingenieros de todos los miembros del equipo. Hubo aprendizaje técnico donde se profundizó lo estudiado en el aula. Se complementa la teoría dada por la profesora con fuentes en Internet y además se puso en práctica todo este conocimiento, lo que ayudó a reforzar aún más. También se trabajan habilidades blandas tanto intrapersonales como interpersonales.

Se mejoró el trabajo en equipo y la organización de cada miembro. Con respecto a las metodologías de trabajo, se probó aplicar la delegación. Esto significa que a cada miembro se le asignaba un avance y este debió compartirlo con el resto del grupo apenas lo terminara. Es importante que esto se aplique solo para tareas más sencillas como elementos de la documentación. Para tareas más complejas se dio un trabajo entre todos los miembros donde todos aportan ideas y se ayudaban mutuamente. Por ejemplo, en la creación de semáforos con nuevas librerías para todo el equipo y el manejo de los diferentes algoritmos como writers, readers, espías y forma de enviar mensajes.

Adicionalmente a esto, también se aprenden lecciones técnicas como el manejo adecuado de recursos, ya que la implementación debe asegurar un manejo

adecuado de los recursos del sistema, como los procesos y los sockets. Si estos no se gestionan adecuadamente, pueden surgir problemas de rendimiento.

Tanto como en la parte de planificación y diseño se hizo una comprensión clara de los requisitos, los algoritmos de planificación de CPU, los semáforos y la comunicación de sockets antes de comenzar la implementación, junto con pruebas exhaustivas ya son necesarias para detectar y corregir errores en el sistema antes de desplegarlo en producción.

Para este proyecto se dio un uso intensivo de medios de comunicación digitales como WhatsApp o Discord. Debido a los horarios y actividades personales de cada miembro, fue muy difícil que todos estuvieran disponibles en el mismo momento y en el mismo lugar. La ventaja es que se fortaleció el uso de estas herramientas que cada vez son más comunes en el ámbito profesional, como el sistema de control de versiones GitHub y el uso de las ramas de trabajo individual y grupal. Adicionalmente, siempre se procuró una participación de todos los miembros y la colaboración a pesar de ser medios virtuales.

Afortunadamente no se tuvieron problemas a lo interno del grupo. Solo se fortalecieron habilidades como la cooperación, trabajo en equipo y organización de tareas. Estas son características muy importantes para satisfacer el mercado laboral.

Por otro lado, este proyecto fue especialmente beneficioso para reforzar los conceptos vistos en clase donde todos los miembros pudimos aplicar estos conceptos y técnicas aprendidas.

En términos generales la comunicación y trabajo en equipo siempre fue presente ya que nos apoyamos de buenas herramientas que impulsan esta práctica como Kanban, ya que es importante visualizar el trabajo en todo momento. Kanban permite a los equipos ver el estado actual de los proyectos en tiempo real, lo que ayuda a identificar cuellos de botella y problemas de flujo de trabajo. Esto puede ayudar a los equipos a priorizar tareas y a asegurarse de que se están enfocando en las áreas más críticas del proyecto.

Casos de pruebas

Los casos de prueba deben ser diseñados de manera exhaustiva y sistemática, cubriendo diferentes escenarios y posibles situaciones en las que el sistema podría fallar. De esta manera, se asegura que el sistema funcione correctamente en todo momento y se minimiza el riesgo de errores o fallos inesperados. En el siguiente apartado, se presentarán los casos de prueba diseñados para cada uno de los componentes del sistema.

Programa Inicializador

Se prueba el programa con una cantidad de memoria de 5:

```
./Pinit

Uso: ./Pinit <num_vectors>

./Pinit 5

Inicialización completa. El programa inicializador terminará.
```

Programa Writer

Se prueba el writer con 3 writer, con tiempo 3 de sleeping y 3 de writing

```
./Pwriter

Uso: ./Pwriter <num_writers> <sleeping> <writing>

./Pwriter 3 3 3

Writer looking for space

 Escribiendo en la linea 0 fecha 2023-05-22 hora 23:06:52

 Escribiendo en la linea 1 fecha 2023-05-22 hora 23:06:55
```

Escribiendo en la linea 2 fecha 2023-05-22 hora 23:06:58

Escribiendo en la linea 3 fecha 2023-05-22 hora 23:07:01

Escribiendo en la linea 4 fecha 2023-05-22 hora 23:07:04

: El writer 140347791968000 no encuentra espacio

: El writer 140347808753408 no encuentra espacio

: El writer 140347808753408 no encuentra espacio

: El writer 140347800360704 no encuentra espacio

: El writer 140347791968000 no encuentra espacio

: El writer 140347808753408 no encuentra espacio

: El writer 140347800360704 no encuentra espacio

: El writer 140347791968000 no encuentra espacio

: El writer 140347808753408 no encuentra espacio

: El writer 140347800360704 no encuentra espacio

: El writer 140347791968000 no encuentra espacio

: El writer 140347808753408 no encuentra espacio

: El writer 140347800360704 no encuentra espacio

: El writer 140347791968000 no encuentra espacio

: El writer 140347808753408 no encuentra espacio

: El writer 140347800360704 no encuentra espacio

: El writer 140347791968000 no encuentra espacio

Escribiendo en la linea 1 fecha 2023-05-22 hora 23:07:32

Escribiendo en la linea 2 fecha 2023-05-22 hora 23:07:35

Escribiendo en la linea 3 fecha 2023-05-22 hora 23:07:38

```
 Escribiendo en la linea 0 fecha 2023-05-22 hora 23:07:50

 Escribiendo en la linea 1 fecha 2023-05-22 hora 23:07:53

 Escribiendo en la linea 4 fecha 2023-05-22 hora 23:07:56

 Escribiendo en la linea 1 fecha 2023-05-22 hora 23:08:09

Error al obtener el ID de la memoria compartida SPY: No such file or
directory

 Escribiendo en la linea 2 fecha 2023-05-22 hora 23:08:10

Error al obtener el ID de la memoria compartida SPY: No such file or
directory

 Escribiendo en la linea 4 fecha 2023-05-22 hora 23:08:10

Terminated
```

Programa Reader

Se prueba el reader con 3 readers, con tiempo 3 de sleeping y 3 de reading

```
./PreaderEgo 3 3 3

Reader looking for a line

 Robando la linea 3 fecha 2023-05-22 hora 23:07:23

 Robando la linea 2 fecha 2023-05-22 hora 23:07:26

 Robando la linea 1 fecha 2023-05-22 hora 23:07:29

 Robando la linea 4 fecha 2023-05-22 hora 23:07:41

 Robando la linea 1 fecha 2023-05-22 hora 23:07:44

 Robando la linea 0 fecha 2023-05-22 hora 23:07:47

 Robando la linea 2 fecha 2023-05-22 hora 23:08:00
```

```
 Robando la linea 4 fecha 2023-05-22 hora 23:08:03

 Robando la linea 1 fecha 2023-05-22 hora 23:08:06

Error al obtener el ID de la memoria compartida SPY: No such file
or directory

Error al obtener el ID de la memoria compartida SPY: No such file
or directory

 Robando la linea 4 fecha 2023-05-22 hora 23:08:10

 Robando la linea 1 fecha 2023-05-22 hora 23:08:10

Terminated
```

Programa Espía

Se prueba el programa espía que solo se ejecuta y va accediendo a la memoria

```
./Pspy


             ### Spy ###

       # Memory

Leyendo el mensaje PID: 140347808753408 Linea:0 Fecha: 2023-05-22 Hora:
23:06:52

Leyendo el mensaje PID: 140347808753408 Linea:1 Fecha: 2023-05-22 Hora:
23:07:32

Leyendo el mensaje PID: 140347800360704 Linea:2 Fecha: 2023-05-22 Hora:
23:07:35

Leyendo el mensaje PID: 140347791968000 Linea:3 Fecha: 2023-05-22 Hora:
23:07:38

Leyendo el mensaje PID: NULL Linea:4 Fecha: NULL Hora: NULL
```

```
       # States
PID: 140347808753408, Type: Writer, Action: Bloqueado

PID: 140347800360704, Type: Writer, Action: Bloqueado

PID: 140347791968000, Type: Writer, Action: Durmiendo

PID: 140509152388864, Type: Reader, Action: Bloqueado

PID: 140509169174272, Type: Reader, Action: Bloqueado

PID: 140509160781568, Type: Reader, Action: Memoria

PID: 139753192638208, Type: Reader Egoista, Action: Memoria

PID: 139753209423616, Type: Reader Egoista, Action: Bloqueado

PID: 139753201030912, Type: Reader Egoista, Action: Bloqueado


./Pspy


             ### Spy ###

       # Memory

Leyendo el mensaje PID: 140347808753408 Linea:0 Fecha: 2023-05-22 Hora:
23:07:50

Leyendo el mensaje PID: NULL Linea:1 Fecha: NULL Hora: NULL

Leyendo el mensaje PID: 140347800360704 Linea:2 Fecha: 2023-05-22 Hora:
23:07:35

Leyendo el mensaje PID: 140347791968000 Linea:3 Fecha: 2023-05-22 Hora:
23:07:38

Leyendo el mensaje PID: NULL Linea:4 Fecha: NULL Hora: NULL
```

```
       # States
PID: 140347808753408, Type: Writer, Action: Memoria
PID: 140347800360704, Type: Writer, Action: Bloqueado
PID: 140347791968000, Type: Writer, Action: Bloqueado
PID: 140509152388864, Type: Reader, Action: Bloqueado
PID: 140509169174272, Type: Reader, Action: Bloqueado
PID: 140509160781568, Type: Reader, Action: Bloqueado
PID: 139753192638208, Type: Reader Egoista, Action: Bloqueado
PID: 139753209423616, Type: Reader Egoista, Action: Bloqueado
PID: 139753201030912, Type: Reader Egoista, Action: Durmiendo
```

Programa Pend

Se prueba programa finalizador que libera la memoria y se muestra en pantalla la bitácora

```
./Pend

      ### --- EXECUTION START --- ###
PID: 2841, Action: Writer, Time: 2023-05-22 23:06:52,
       Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:06:52
LINEA: 0]


PID: 2841, Action: Writer, Time: 2023-05-22 23:06:55,
       Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:06:55
LINEA: 1]
```

PID: 2841, Action: Writer, Time: 2023-05-22 23:06:58,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:06:58 LINEA: 2]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:01,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:01 LINEA: 3]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:04,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:04 LINEA: 4]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:07,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:06:52 LINEA: 0]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:07,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:06:52 LINEA: 0]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:07,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:06:52 LINEA: 0]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:13,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:06:55

LINEA: 1]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:14,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:06:55
LINEA: 1]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:14,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:06:55
LINEA: 1]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:20,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:06:58
LINEA: 2]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:20,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:06:58
LINEA: 2]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:20,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:06:58
LINEA: 2]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:07:23,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:01
LINEA: 3]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:07:26,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:06:58 LINEA: 2]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:07:29,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:06:55 LINEA: 1]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:32,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:32 LINEA: 1]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:35,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:04 LINEA: 4]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:35,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:04 LINEA: 4]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:35,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:35 LINEA: 2]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:38,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:07:38 LINEA: 3]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:41,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:07:38 LINEA: 3]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:07:41,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:04 LINEA: 4]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:07:44,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:32 LINEA: 1]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:07:47,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:06:52 LINEA: 0]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:50,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:50 LINEA: 0]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:53,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:50 LINEA: 0]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:53,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:50 LINEA: 0]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:53,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:53 LINEA: 1]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:56,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:07:56 LINEA: 4]

PID: 2871, Action: Reader, Time: 2023-05-22 23:08:00,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:07:56 LINEA: 4]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:08:00,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:35 LINEA: 2]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:08:03,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:07:56 LINEA: 4]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:08:06,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:53 LINEA: 1]

PID: 2841, Action: Writer, Time: 2023-05-22 23:08:09,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:08:09 LINEA: 1]

PID: 2871, Action: Reader, Time: 2023-05-22 23:08:10,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:08:09 LINEA: 1]

PID: 2841, Action: Writer, Time: 2023-05-22 23:08:10,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:08:10 LINEA: 2]

PID: 2841, Action: Writer, Time: 2023-05-22 23:08:10,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:08:10 LINEA: 4]

PID: 2871, Action: Reader, Time: 2023-05-22 23:08:10,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:08:09 LINEA: 1]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:08:10,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:08:10 LINEA: 4]

PID: 2871, Action: Reader, Time: 2023-05-22 23:08:10,

```
       Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:50
LINEA: 0]



PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:08:10,

       Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:08:09
LINEA: 1]



Finalización completa. Todos los recursos han sido liberados.
```

Acceso a la Bitácora

Se revisa la bitácora para validar que la información se haya guardado de manera correcta

```
cat bitacora.txt


      ### --- EXECUTION START --- ###


PID: 2841, Action: Writer, Time: 2023-05-22 23:06:52,

       Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:06:52
LINEA: 0]



PID: 2841, Action: Writer, Time: 2023-05-22 23:06:55,

       Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:06:55
LINEA: 1]



PID: 2841, Action: Writer, Time: 2023-05-22 23:06:58,

       Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:06:58
LINEA: 2]
```

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:01,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:01 LINEA: 3]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:04,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:04 LINEA: 4]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:07,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:06:52 LINEA: 0]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:07,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:06:52 LINEA: 0]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:07,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:06:52 LINEA: 0]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:13,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:06:55 LINEA: 1]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:14,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:06:55

LINEA: 1]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:14,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:06:55
LINEA: 1]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:20,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:06:58
LINEA: 2]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:20,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:06:58
LINEA: 2]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:20,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:06:58
LINEA: 2]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:07:23,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:01
LINEA: 3]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:07:26,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:06:58
LINEA: 2]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:07:29,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:06:55 LINEA: 1]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:32,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:32 LINEA: 1]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:35,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:04 LINEA: 4]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:35,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:04 LINEA: 4]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:35,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:35 LINEA: 2]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:38,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:07:38 LINEA: 3]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:41,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:07:38 LINEA: 3]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:07:41,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:04 LINEA: 4]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:07:44,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:32 LINEA: 1]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:07:47,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:06:52 LINEA: 0]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:50,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:50 LINEA: 0]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:53,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:50 LINEA: 0]

PID: 2871, Action: Reader, Time: 2023-05-22 23:07:53,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:50 LINEA: 0]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:53,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:53 LINEA: 1]

PID: 2841, Action: Writer, Time: 2023-05-22 23:07:56,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:07:56 LINEA: 4]

PID: 2871, Action: Reader, Time: 2023-05-22 23:08:00,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:07:56 LINEA: 4]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:08:00,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:35 LINEA: 2]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:08:03,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:07:56 LINEA: 4]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:08:06,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:07:53 LINEA: 1]

PID: 2841, Action: Writer, Time: 2023-05-22 23:08:09,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:08:09 LINEA: 1]

PID: 2871, Action: Reader, Time: 2023-05-22 23:08:10,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:08:09 LINEA: 1]

PID: 2841, Action: Writer, Time: 2023-05-22 23:08:10,

Message:[PID: 140347800360704 FECHA: 2023-05-22 HORA: 23:08:10 LINEA: 2]

PID: 2841, Action: Writer, Time: 2023-05-22 23:08:10,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:08:10 LINEA: 4]

PID: 2871, Action: Reader, Time: 2023-05-22 23:08:10,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:08:09 LINEA: 1]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:08:10,

Message:[PID: 140347791968000 FECHA: 2023-05-22 HORA: 23:08:10 LINEA: 4]

PID: 2871, Action: Reader, Time: 2023-05-22 23:08:10,

Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:07:50 LINEA: 0]

PID: 2885, Action: ReaderEgo, Time: 2023-05-22 23:08:10,

```
        Message:[PID: 140347808753408 FECHA: 2023-05-22 HORA: 23:08:09
LINEA: 1]



        ### --- EXECUTION ENDED --- ###
```

Comparación

En los sistemas operativos Unix-like, como Linux, existen varios mecanismos para compartir memoria entre procesos. Dos métodos comunes son **mmap** y **shmget**. Ambos permiten a los procesos acceder a regiones de memoria compartida, pero difieren en su enfoque y características. En esta comparación, explicaremos las diferencias y similitudes entre **mmap** y **shmget**, y examinaremos sus detalles y casos de uso

La principal diferencia entre **mmap** y **shmget** es que mmap crea un mapeo de memoria de un archivo, mientras que shmget crea un segmento de memoria compartida. Un mapeo de memoria es una forma de acceder a un archivo como si fuera memoria, mientras que un segmento de memoria compartida es una región de memoria que se comparte entre varios procesos (*Linux shared memory: shmget() vs mmap()*, s. f.)..

1. **mmap**:

La función **mmap** crea un mapeo de memoria de un archivo. Esto significa que el archivo es accesible como si fuera memoria. El archivo se puede acceder utilizando los mismos punteros y desplazamientos que se usarían si estuviera en memoria.

- mmap (mapeo de memoria) es un sistema de llamadas al sistema que permite a los procesos compartir regiones de memoria mediante el mapeo de archivos o dispositivos en el espacio de direcciones de un proceso.

- Con mmap, un archivo o dispositivo se mapea directamente en el espacio de direcciones virtuales de un proceso, lo que permite que los procesos compartan datos leyendo y escribiendo en estas regiones de memoria compartida.
- La memoria compartida creada a través de mmap es persistente, lo que significa que se mantiene incluso después de que los procesos que la crearon hayan finalizado. También se puede utilizar para la comunicación interproceso (IPC, por sus siglas en inglés) en tiempo real.
- mmap se utiliza comúnmente para compartir memoria entre procesos relacionados, como procesos hijos y su padre, en entornos de programación de alto nivel como C o C++.

Las banderas que se pueden usar con mmap incluyen:

- **MAP_SHARED**: El mapeo se comparte entre todos los procesos que tienen un mapeo para el mismo archivo.
- **MAP_PRIVATE**: El mapeo es privado para el proceso actual.
- **MAP_ANONYMOUS**: El mapeo no está respaldado por un archivo.

2. **shmget**:

- shmget (obtención de memoria compartida) es una función de la biblioteca de llamadas al sistema que permite a los procesos acceder a una región de memoria compartida identificada por una clave.
- shmget se utiliza en conjunción con otras funciones, como shmat y shmdt, para adjuntar y desadjuntar la memoria compartida al espacio de direcciones de un proceso.
- La memoria compartida creada a través de shmget se mantiene en el sistema operativo y puede ser accedida por múltiples procesos a través de la clave asociada.
- A diferencia de mmap, la memoria compartida creada con shmget no está vinculada a un archivo o dispositivo. En su lugar, se crea una

región de memoria compartida independiente que se puede utilizar para la comunicación y el intercambio de datos entre procesos.

Las banderas que se pueden usar con shmat incluyen:

- **SHM_RDONLY**: El segmento de memoria compartida se mapea como solo lectura.
- **SHM_RDWR**: El segmento de memoria compartida se mapea como lectura y escritura.

Una tabla de las principales diferencias entre **mmap** y **shmget**:

Característica	mmap	shmget
Crea	Mapeo de memoria	Segmento de memoria compartida
Accede a	Un archivo	Una región de memoria
Compartido entre	Múltiples procesos	Dos procesos
Más flexible	Sí	No
Más complejo	No	Sí

Analizando lo siguiente, **mmap** se utiliza para mapear archivos o dispositivos en el espacio de direcciones de un proceso, mientras que **shmget** se utiliza para crear y acceder a regiones de memoria compartida independientes en el sistema operativo. Ambos métodos permiten el intercambio de datos entre procesos, pero tienen diferencias en términos de persistencia y uso.

Manual de usuario

Este manual está planeado para alguien que no sabe cómo fue que se programó la solución y ya tiene una versión completa y estable del producto.

Esta aplicación readers – writers es un proyecto que permite implementar diferentes algoritmos de sincronización existen diferentes tipos de semáforos. Se toma en

cuenta que dado que en este problema los procesos a utilizar son programas separados, se debe buscar el tipo de semáforo que me permite la comunicación entre programas. Este problema incluye una memoria compartida entre todos los actores, y para ello se utiliza la librería de C llamada ncurses.

Es importante mencionar que, antes de iniciar con la ejecución de este proyecto, es necesario realizar la instalación de esta librería en su sistema operativo. A continuación, se presentarán los pasos necesarios para llevar a cabo la instalación de la librería y la ejecución del simulador.

Paso 1: Instalación de la librería ncurses

Para instalar la librería de ncurses en su sistema operativo, utilice el siguiente comando en su terminal:

```
sudo apt-get install libncursesw5-dev
```

Paso 2: Compilación del programa

Para compilar todos los archivos que vamos a ejecutar, utilice el siguiente comando en su terminal:

```
chmod +x compilar.sh
```

Para compilar el clear, utilice el siguiente comando en su terminal:

```
chmod +x clear.sh
```

Paso 3: Ejecución del programa

Lo primero que se tiene que hacer es ejecutar el archivo compilar.sh para que compile todos los demás archivos necesarios:

`./compilar.sh`

Es importante que antes de hacer este pinit, se libere la memoria y los semáforos por si tiene en ella algún elemento guardado de alguna ejecución anterior puede derivar en errores y bloqueo del programa si no se hace .Para ello, utilice el siguiente comando en su terminal:

```
./Pend
```

Una vez compilado el Pinit, se puede ejecutar la información de los algoritmos de sincronización de procesos. Para ello, utilice el siguiente comando en su terminal:

```
./Pinit <Cantidad de líneas>
```

Ejemplo

```
./Pinit 5
```

Paso 4: Ejecución del writer and readers

Para ejecutar este algoritmo que va escribir, se debe abrir una nueva terminal y ejecutar el siguiente comando en su terminal:

```
./Pwriter <cant Escritores> <tiempo de espera> <tiempo
escribiendo>
```

La cantidad de escritores son la cantidad de elementos que van a escribir información, en cuanto al tiempo de espera, es el tiempo que van a tener que esperar para poder acceder a la memoria si es que está disponible y por último el tiempo escribiendo, será el tiempo que van a tener la memoria y van escribir información

```
Ejemplo: ./Pwriter 10 2 2
```

Para el algoritmo que va a leer la información recibe los mismos parámetros lo único que cambia es el archivo que se está ejecutando, para hacer esto es de la siguiente manera:

```
./Preader <cant lectores> <tiempo de espera> <tiempo leyendo>
```

```
Ejemplo: ./Preader 10 2 2
```

Paso 6: Reader Egoísta

Para el algoritmo que va a leer de forma egoísta la información recibe los mismos parámetros que reader normal lo único que cambia es el archivo que se está ejecutando, para hacer esto es de la siguiente manera:

```
./PreaderEgo <cant de ego> <tiempo de espera> <tiempo leyendo>
```

```
Ejemplo: ./PreaderEgo 10 2 2
```

Paso 7: Espía

Para ejecutar el espía solo se tiene que llamar el archivo en una terminal nueva, esto sin ningún parámetros, se puede hacer de la siguiente manera:

`Ejemplo: ./Pspy`

Paso 8: Liberar memoria

Es importante que se libere la memoria y los semáforos al terminar la ejecución del programa. También cerrar el archivo de bitácora y mostrarlo en pantalla. Para ello, utilice el siguiente comando en su terminal:

```
./Pend
```

Además, es importante que se eliminen los archivos compilados que se crearon. Para ello, utilice el siguiente comando en su terminal:

```
./clear.sh
```

Bitácora

A continuación se muestra una tabla con las actividades realizadas durante el desarrollo de este proyecto, las fechas corresponden a los días que se hizo reunión; no obstante, se excluye el tiempo utilizado para la búsqueda de información y referencias.

Bitácora		
Fecha	Actividad	Detalles relevantes
26/04/2023	Reunión Inicial	Estudiar el proyecto, ver requisitos y distribuir tareas
29/04/2023	Revisar estructura y manejo de pthreads	Gran parte del proyecto se utilizan los semáforos, por eso se hizo una reunión específica para estudiarlos y aplicarlos
2/05/2023	Revisión y Pruebas del Servidor	Se corrobora que la información solicitada y el código creado por el equipo sobre el inicializador sea la correcta y tenga funcionamiento Conclusión : Aprobada
6/05/2023	Revision de Clientes	Se verifica que la información enviada por el semaforos sea recibida y pueda ser manejada por los writers y readers Conclusión : Aprobada

13/05/2023	Documentación y revisar avances	Se empieza la documentación, además de esto se revisan los avances del writers, readers, reader egoísta y espía Conclusión : En todos los algoritmos
16/05/2023	Corrección de errores	Se corrigen los errores de los writers y readers Conclusión : Pendiente egoísta y espía
21/05/2023	Corrección de errores	Se corrigen errores egoísta y espía
22/05/2023	Pruebas	Se corren pruebas en general para revisar el funcionamiento completo del programa. Conclusión : Todos los datos y pruebas son correctos

Bibliografía

Chapter 12: shared memory virtual filesystem. Shared Memory Virtual Filesystem. *(s.f.).*https://www.kernel.org/doc/gorman/html/understand/understand015.html

Understanding memory mapping. IBM. (s.f.).

https://www.ibm.com/docs/en/aix/7.2?topic=memory-understanding-mapping

Linux shared memory: shmget() vs mmap() – iTecNote. (s. f.).

https://itecnote.com/tecnote/linux-shared-memory-shmget-vs-mmap/

Repositorio:

https://github.com/AndresMasis/SO_Proyecto2