Instituto Tecnológico de Costa Rica

Escuela de Computación

Principios de Sistemas Operativos GR2

Proyecto #2

Sincronización de Procesos

Profesora:

Erika Marin Schumann

Integrantes

Moisés Higuerey Hernández (2018092411)

Josue Alvarado Mares (2018143069)

Valeria Quesada Benavides (2018085308)

Fecha de Entrega:

24/5/2021

I Semestre 2022

Índice

Introducción

Para que la comunicación entre diferentes programas sea posible, la memoria compartida es parte esencial. Esto ayuda a que los diferentes procesos puedan compartirse, de manera que los datos viajan y se actualizan de manera constante. El problema a resolver en el proyecto actual, se encuentra en la creación de cuatro programas, que con diferentes funciones, se encargan de diferentes situaciones compartiendo un conjunto de memoria y procesos según se requiera.

Para la resolución del problema, se crean cuatro programas que utilizan memoria compartida. Esto con el fin de que el primero, llamado inicializador, cree un espacio en donde sea posible insertar diferentes procesos. Posteriormente el programa llamado productor, se encarga de llamar al ambiente creado por el inicializador y su objetivo es crear, insertar, eliminar y matar procesos de dicha memoria. El programa espía es el tercero encontrado en la lista, este se encarga de una interacción con el usuario, de manera que se pueda consultar los datos del proceso actual. Por último, se encuentra el finalizador, este es el encargado de cerrar el ambiente compartido creado por el inicializador.

Estrategia de Solución

En este apartado se explica la manera en la que se abordó la estrategia de la solución, el tipo de semáforos que se utilizó así como la explicación de cómo se logró la sincronización.

Semáforos utilizados

Para brindar una solución que satisfaga los requerimientos planteados por en el enunciado se decidió tomar los puntos fuertes de semáforos tales como el mutex, la panadería y los filósofos, debido a que en nuestra solución permite el acceso un solo proceso para escribir en memoria (exclusión mutua) con la presencia de una cola donde se encuentran en espera (bloqueados) según el orden de solicitud para n procesos.

Sincronización

Se efectuó sincronización de procesos para lograr la solución del problema anteriormente explicado. Para ello se utilizaron 2 semáforos para el manejo de los procesos y el envío de información entre programas respectivamente, esto apoyado con el uso de hilos para mejorar el desempeño obtenido, cada semáforo permite un acceso único para un mejor control de la zona crítica.

Análisis de resultados

En este apartado se encuentra una tabla en donde se observa qué se logró dentro del proyecto.

Se logró abarcar el 100% del proyecto. Además, con el fin de abarcar los más importante, se clasificaron 3 categorías:

- High: Es de suma importancia para poder utilizar de la mejor manera la memoria compartida.
- Medium: Ayuda a un mejor uso de la memoria compartida y beneficia al usuario, sin embargo, no depende de esto.
- Low: No es relevante para el correcto funcionamiento de la memoria compartida.

#	Tarea	Prioridad	% Completado	Listo
1	**Simulación de memoria**			○
1.1	**Programa inicializador**	**HIGH**	**100%**	✓
1.1.1	Creacion de memoria compartida	**HIGH**	100%	✓
1.1.2	Solicitud del tamaño al usuario	LOW	100%	✓
1.2	**Programa productor de procesos**	**HIGH**	**100%**	✓
1.2.1	Creación de procesos	**HIGH**	100%	✓
1.2.2	Selección de algoritmo para la simulación	MEDIUM	100%	✓
1.2.3	Organización de memoria conforme llegan los procesos.	**HIGH**	100%	✓
1.2.4	Generación de randoms	MEDIUM	100%	✓
1.2.5	Algoritmo de paginación y segmentación	**HIGH**	100%	✓
1.2.6	Creación de semáforos.	**HIGH**	100%	✓
1.2.7	Escritura de los estados de los procesos en bitácora	MEDIUM	100%	✓
1.3	**Programa espía**	MEDIUM	**100%**	✓
1.3.2	Estado de los procesos den determinado momento	MEDIUM	100%	✓
1.3.3	Menú de usuario	MEDIUM	100%	✓
1.4	**Programa finalizador**	**HIGH**	100%	✓
1.4.1	Devolución de recursos solicitados	**HIGH**	100%	✓
1.4.2	Cierre de la bitácora	LOW	100%	✓

Lecciones aprendidas

En este apartado se encuentra un listado de lecciones aprendidas por cada uno de los integrantes del grupo.

Josue Alvarado

1. En el desarrollo de este proyecto aprendí sobre el uso de semáforo tipo mutex en el lenguaje de programación C dentro del sistema operativo de linux.
2. Se desarrollaron habilidades de comprensión y solución de problemas que requieren una sincronización entre sí teniendo en cuenta la posibilidad de poseer amplios recursos o por el contrario escasos recursos compartidos entre si.
3. Se aprendió sobre la búsqueda de información como lo pueden ser arreglos de enteros, datos primitivos e incluso estructuras por medio del uso de llamadas al sistema y el uso de librerías para la implementación de semáforos.
4. Se vio necesario el uso de herramientas visuales para representar las ideas de solución por parte del equipo sobre el camino de desarrollo para el proyecto así como para debatir ideas de desarrollo propias del equipo y de esta manera unificar los diferentes puntos de vista.

Moises Higuerey

1. Se aprendió sobre las diferencias de los diferentes tipos de semáforos que existen en el lenguaje C del sistema operativo Linux, así de para qué casos es mejor aplicar cada uno.
2. Se desarrolló la comprensión del tema de memoria compartida y sincronización de los procesos. Se vio la importancia de mitigar los errores que pueden ocurrir por el mal acceso a la memoria compartida y su modificación.
3. Se aprendió de los diferentes métodos para utilizar memoria compartida en el lenguaje C de Linux. También se aprendió en qué casos es mejor aplicar las diferentes funciones (shmget, mmap, etc.) y de los diferentes métodos de acceso a la memoria.
4. Se pudo hacer más discusión sobre los diferentes elementos del proyecto por medio de reuniones para el trabajo cooperativo en el código. Esto nos permitió

encontrar problemas durante el desarrollo más fácilmente y poder solucionarlos entre todos.

Valeria Quesada

1. El tener un control de tiempo y poder administrar bien cada una de las actividades es de suma importancia para el correcto avance del proyecto así como el desempeño correcto de cada uno de los integrantes. Al trabajar desde que se mandaron las indicaciones por parte de la profesora, ayudó a que no se trabajara más de 3 horas diarias en dicho proyecto. Esto demostró la eficiencia, ya que los temas no cansaban por lo que el avance fue mayor.
2. Aunque parezca tarea sencilla, el lograr sincronizar los procesos con los semáforos requiere de mucho cuidado y pruebas. Siempre se debe tomar en cuenta las prioridades del proyecto, por lo que, aunque hayan muchas soluciones, el buscar la mejor y además el considerar el tiempo que se posee, ayuda a un mejor desempeño y aprendizaje.
3. La comunicación entre los miembros del equipo es esencial, se demostró que todos tienen diferentes maneras de solucionar cada tarea. Por lo que las reuniones para expresar y explicar cada una de las ideas ayuda a un mejor entendimiento, además, por medio de esta se puede proponer diferentes soluciones y extraer la que se encuentre más optimizada que mejor se adapte al objetivo.
4. Se demostró que en muchas ocasiones los miembros del equipo tenían la misma idea de solución para alguna de las tareas, por lo que, el dibujar y hacer pseudocódigo ayudó a que el tiempo de explicación fuera menor y no existieran ambigüedades entre las ideas expuestas.

Casos de pruebas

En esta sección se muestran una serie de casos de pruebas en donde se explican los resultados esperados y obtenidos.

Programa inicializador

Se espera que al inicializarlo solicite al usuario escribir el tamaño de la memoria. Esto fue exitoso y se obtuvo:

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./inicializador

Enter shared memory size (please write it in numbers): 15
```

Posteriormente, se espera que el programa cree una memoria compartida con el tamaño de los datos numéricos ingresados por el usuario, mostrando un mensaje que indica que ya la creó. Esto fue exitoso y se obtuvo:

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./inicializador

Enter shared memory size (please write it in numbers): 15
Shared memory was created.
```

Esto se puede ver mejor al momento de correr el procesador, en donde cuando el primer proceso busca espacio, se espera que en la bitácora escriba estos espacios vacíos en -1. Esto con el fin de representar el espacio de manera visual. Esto fue exitoso y se obtuvo:

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./inicializador

Enter shared memory size (please write it in numbers): 15
Shared memory was created.

[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./productor

Select a memory management scheme
1. Paging
2. Segmentation
2

You have selected segmentation.
------------------------------

Process 0 with size 5 is searching for memory space.
```

```
Actual Time: 2022-05-24 14:47:01
Process 0 with size 5 is searching for memory space.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Programa productor

Al correr dicho programa, se espera que se le solicite al usuario seleccionar un tipo de asignación a memoria, entre los cuales se muestra paginación y segmentación. Esto fue exitoso y se obtuvo:

```
Select a memory management scheme
1. Paging
2. Segmentation
```

A la hora del usuario seleccionar el manejo de memoria deseado, se espera que el programa muestre un mensaje verificando lo ingresado por el usuario y comience la ejecución de los procesos, escribiendo los cambios en bitácora y mostrando un resumen de su avance en pantalla. Esto fue exitoso y se obtuvo:

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./inicializador

Enter shared memory size (please write it in numbers): 15
Shared memory was created.

[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./productor

Select a memory management scheme
1. Paging
2. Segmentation
2

You have selected segmentation.
-------------------------------

Process 0 with size 5 is searching for memory space.
Process 0 found space in memory.
Process 1 with size 4 is searching for memory space.
Process 1 found space in memory.
Process 0 finished, its space is free.
Process 2 with size 8 is searching for memory space.
Process 2 found space in memory.
```

```
Actual Time: 2022-05-24 14:47:01
Process 0 with size 5 is searching for memory space.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:47:01
Process 0 was inserted into the next memory location(s):
0, 1, 2, 3, 4,
Actual array (-1 means free space):
0 0 0 0 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:47:47
Process 1 with size 4 is searching for memory space.
Actual array (-1 means free space):
0 0 0 0 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:47:47
Process 1 was inserted into the next memory location(s):
5, 6, 7, 8,
Actual array (-1 means free space):
0 0 0 0 0 1 1 1 1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:47:52
Process 0 finished, its space  is free.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:48:33
Process 2 with size 8 is searching for memory space.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:48:33
Process 2 was inserted into the next memory location(s):
0, 1, 2, 3, 4, 9, 10, 11,
Actual array (-1 means free space):
2 2 2 2 2 1 1 1 1 2 2 2 -1 -1 -1
```

Paginación

Es uno de los tipos de administración de memoria que puede seleccionar el usuario a la hora de correr el programa productor. Se espera que el programa sea capaz de crear procesos con valores random de 1 a 10 páginas. Esta prueba fue

exitosa debido a que se pueden observar procesos con tamaño de 1 a 10.

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./productor

Select a memory management scheme
1. Paging
2. Segmentation
1

You have selected paging.
------------------------------

Process 0 with size 1 is searching for memory space.
Process 0 found space in memory.
Process 1 with size 7 is searching for memory space.
Process 1 died, it did not found memory space.
Process 0 finished, its space is free.
Process 2 with size 2 is searching for memory space.
Process 2 found space in memory.
Process 3 with size 6 is searching for memory space.
Process 3 died, it did not found memory space.
Process 2 finished, its space is free.
Process 4 with size 7 is searching for memory space.
Process 4 died, it did not found memory space.
Process 5 with size 10 is searching for memory space.
Process 5 died, it did not found memory space.
Process 6 with size 3 is searching for memory space.
Process 6 found space in memory.
Process 6 finished, its space is free.
Process 7 with size 2 is searching for memory space.
Process 7 found space in memory.
Process 7 finished, its space is free.
Process 8 with size 5 is searching for memory space.
Process 8 found space in memory.
Process 8 finished, its space is free.
```

Otro de los factores esperados, es que dicho algoritmo sea capaz de buscar y asignar espacios en memoria, sin embargo, se espera que si no hay suficientes, los procesos mueran. Se realizó una paginación de tamaño 5, por lo que los segmentos que tengan un tamaño mayor, no puedan ingresar. Al realizar la prueba, se obtuvo un resultado exitoso, en donde solo los menores o iguales a 5 entran en memoria:

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./inicializador

Enter shared memory size (please write it in numbers): 5
Shared memory was created.

[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./productor

Select a memory management scheme
1. Paging
2. Segmentation
1

You have selected paging.
------------------------------

Process 0 with size 10 is searching for memory space.
Process 0 died, it did not found memory space.
Process 1 with size 9 is searching for memory space.
Process 1 died, it did not found memory space.
Process 2 with size 9 is searching for memory space.
Process 2 died, it did not found memory space.
Process 3 with size 2 is searching for memory space.
Process 3 found space in memory.
```

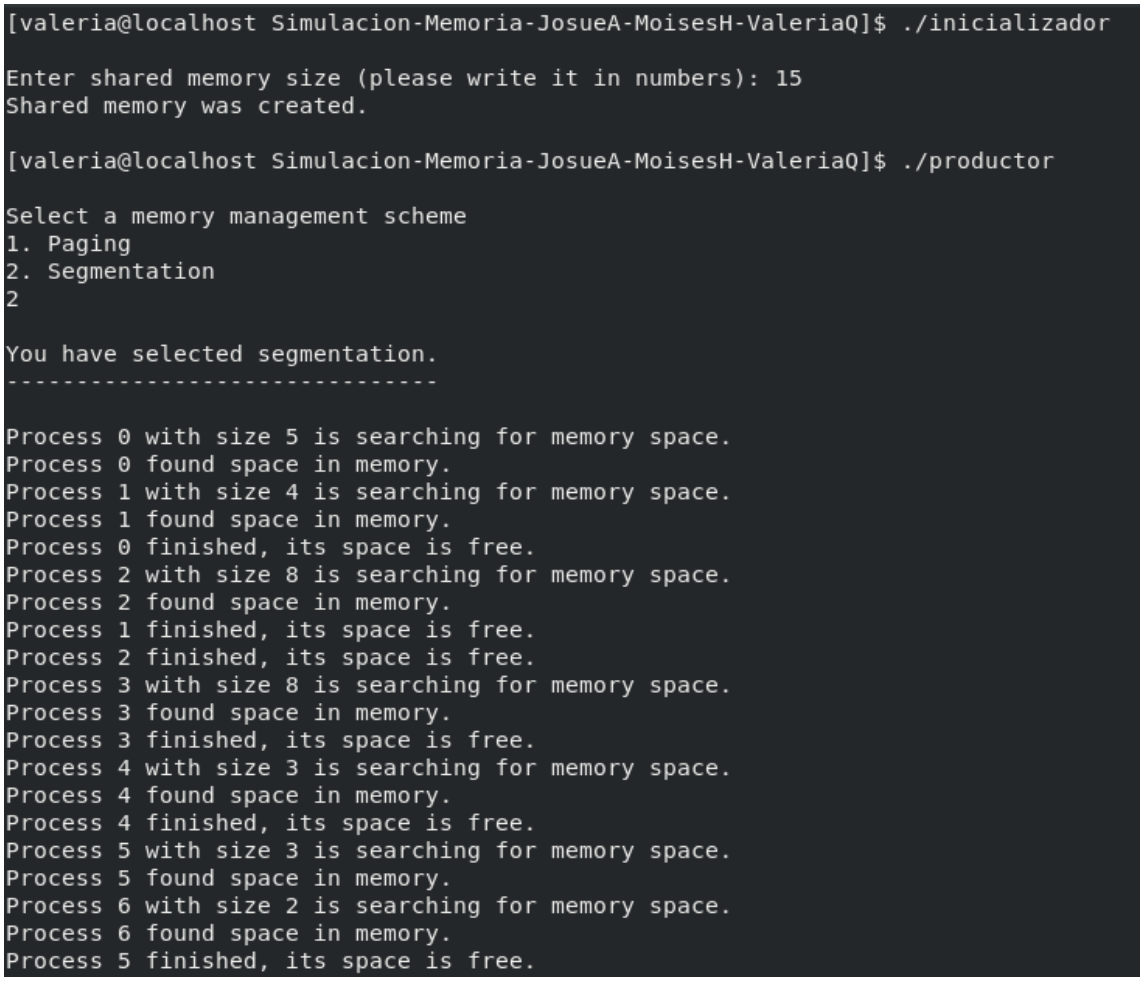
Se espera que cada uno de los procesos tenga un tiempo de 20 a 60 segundos en memoria. Esto fue exitoso y se puede visualizar por medio de la bitácora. El proceso 0 se inserta en el minuto 39.18 y sale en el segundo 40.09, mostrando un tiempo en memoria de 51 segundos. El proceso 1 entra en el min 40.17 y sale en el 41.13, con una duración de 56 segundos en memoria.

```
Select a memory management scheme
1. Paging
2. Segmentation
1

You have selected paging.
------------------------------

Process 0 with size 10 is searching for memory space.
Process 0 found space in memory.
Process 0 finished, its space is free.
Process 1 with size 7 is searching for memory space.
Process 1 found space in memory.
Process 2 with size 9 is searching for memory space.
Process 2 died, it did not found memory space.
Process 1 finished, its space is free.
```

```
Actual Time: 2022-05-24 20:39:18
Process 0 with size 10 is searching for memory space.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 20:39:18
Process 0 was inserted into the next memory location(s):
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
Actual array (-1 means free space):
0 0 0 0 0 0 0 0 0 0 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 20:40:09
Process 0 finished, its space  is free.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 20:40:17
Process 1 with size 7 is searching for memory space.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 20:40:17
Process 1 was inserted into the next memory location(s):
0, 1, 2, 3, 4, 5, 6,
Actual array (-1 means free space):
1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 20:40:57
Process 2 with size 9 is searching for memory space.
Actual array (-1 means free space):
1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 20:40:57
Process 2 died, it did not found memory space.
Actual array (-1 means free space):
1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1


Actual Time: 2022-05-24 20:40:57
Process 2 died, it did not found memory space.
Actual array (-1 means free space):
1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 20:41:13
Process 1 finished, its space  is free.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Segmentación

Es uno de los tipos de administración de memoria que puede seleccionar el usuario a la hora de correr el programa productor. Se espera que el programa sea capaz de crear procesos con valores random de 1 a 5 segmentos en donde cada uno tendrá un valor de 1 a 3 segmentos en memoria, esperando que el tamaño total no sobrepase los 15 espacios. En el caso de esta prueba se puede visualizar la manera en la que lo esperado es lo obtenido. Ningún tamaño es mayor a 15.



Otro de los factores esperados, es que dicho algoritmo sea capaz de buscar y asignar espacios en memoria, sin embargo, se espera que si no hay suficientes, los procesos mueran. Se realizó una segmentación de tamaño 5, por lo que los segmentos que tengan un tamaño mayor, no puedan ingresar. Al realizar la prueba, se obtuvo un resultado exitoso:

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./productor

Select a memory management scheme
1. Paging
2. Segmentation
2

You have selected segmentation.
------------------------------

Process 0 with size 9 is searching for memory space.
Process 0 died, it did not found memory space.
Process 1 with size 4 is searching for memory space.
Process 1 found space in memory.
```

Por otra parte, los sub segmentos deben ser capaz de dividirse dependiendo de su tamaño, de manera que puede entrar a memoria. Se esperaba que al correr el algoritmo, hubiera un proceso en donde sus segmentos no se encuentren de manera seguida. Esto fue exitoso y se puede observar la manera en la que el proceso 2 está separado, de forma en la que sus 2 segmentos pueden ingresar a memoria:

```
Actual Time: 2022-05-24 14:48:33
Process 2 with size 8 is searching for memory space.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:48:33
Process 2 was inserted into the next memory location(s):
0, 1, 2, 3, 4, 9, 10, 11,
Actual array (-1 means free space):
2 2 2 2 2 1 1 1 1 2 2 2 -1 -1 -1
```

Además, se espera que cada uno de los procesos tenga un tiempo de 20 a 60 segundos en memoria. Esto fue exitoso y se puede visualizar por medio de la bitácora. El proceso 0 se inserta en el minuto 47.01 y sale en el minuto 47.52, es decir, dura 51 segundos en memoria. El proceso 1 se inserta en el minuto 47.47 y sale en el segundo 48.36, mostrando un tiempo en memoria de 49 segundos. El proceso 2 entra en el min 48.33 y sale en el 48.55, con una duración de 22 segundos en memoria. Por último, es posible observar el proceso 3, que entra en el minuto 49.05 y sale en el minuto 49.50, durando así 45 segundos en memoria.

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./inicializador

Enter shared memory size (please write it in numbers): 15
Shared memory was created.

[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./productor

Select a memory management scheme
1. Paging
2. Segmentation
2

You have selected segmentation.
------------------------------

Process 0 with size 5 is searching for memory space.
Process 0 found space in memory.
Process 1 with size 4 is searching for memory space.
Process 1 found space in memory.
Process 0 finished, its space is free.
Process 2 with size 8 is searching for memory space.
Process 2 found space in memory.
Process 1 finished, its space is free.
Process 2 finished, its space is free.
Process 3 with size 8 is searching for memory space.
Process 3 found space in memory.
Process 3 finished, its space is free.
```

```
Actual Time: 2022-05-24 14:47:01
Process 0 with size 5 is searching for memory space.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:47:01
Process 0 was inserted into the next memory location(s):
0, 1, 2, 3, 4,
Actual array (-1 means free space):
0 0 0 0 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:47:47
Process 1 with size 4 is searching for memory space.
Actual array (-1 means free space):
0 0 0 0 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:47:47
Process 1 was inserted into the next memory location(s):
5, 6, 7, 8,
Actual array (-1 means free space):
0 0 0 0 0 1 1 1 1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:47:52
Process 0 finished, its space  is free.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:48:33
Process 2 with size 8 is searching for memory space.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:48:33
Process 2 was inserted into the next memory location(s):
0, 1, 2, 3, 4, 9, 10, 11,
Actual array (-1 means free space):
2 2 2 2 2 1 1 1 1 2 2 2 -1 -1 -1
```

```
Actual Time: 2022-05-24 14:48:36
Process 1 finished, its space  is free.
Actual array (-1 means free space):
2 2 2 2 2 -1 -1 -1 -1 2 2 2 -1 -1 -1

Actual Time: 2022-05-24 14:48:55
Process 2 finished, its space  is free.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:49:05
Process 3 with size 8 is searching for memory space.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:49:05
Process 3 was inserted into the next memory location(s):
0, 1, 2, 3, 4, 5, 6, 7,
Actual array (-1 means free space):
3 3 3 3 3 3 3 3 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 14:49:50
Process 3 finished, its space  is free.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Programa espia

Se espera que al correr el programa, se muestre un menu con opciones para poder visualizar la memoria, los estados de los procesos o salir del programa. Esto fue exitoso:

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$
 ./espia

Select type of consultation
1. Memory state
2. Processes states
3. Exit
```

Si el usuario selecciona la opción 1, se espera que el programa muestre el tamaño de la memoria junto con su llenado en tiempo real, esta prueba fue exitosa, al

comparar lo que llevaba el productor junto con el programa espía, se puede observar que son los datos de memoria actual.

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./inicializador

Enter shared memory size (please write it in numbers): 15
Shared memory was created.

[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./productor

Select a memory management scheme
1. Paging
2. Segmentation
2

You have selected segmentation.
------------------------------

Process 0 with size 5 is searching for memory space.
Process 0 found space in memory.
```

```
Select type of consultation
1. Memory state
2. Processes states
3. Exit
1

Memory size: 15
| 0 | 0 | 0 | 0 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -
1 | -1 |
```

Si el usuario selecciona la opción 2, se espera que se muestre el estado actual de los procesos. Comparándolos con el productor, se observa que se encuentran en tiempo real y son correctos.

```
Select type of consultation
1. Memory state
2. Processes states
3. Exit
2

Id: 0, State: already done.
Id: 1, State: in memory.
Id: 2, State: in memory.
```

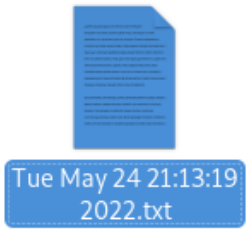
```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./productor

Select a memory management scheme
1. Paging
2. Segmentation
1

You have selected paging.
------------------------------

Process 0 with size 1 is searching for memory space.
Process 0 found space in memory.
Process 1 with size 7 is searching for memory space.
Process 1 found space in memory.
Process 0 finished, its space is free.
Process 2 with size 10 is searching for memory space.
Process 2 found space in memory.
```

Programa finalizador

Se espera que finalice la memoria, muestre un mensaje indicando su acción y cree un archivo con la fecha actual, en donde, la bitácora se guardará. Esta prueba fue exitosa y realizó todas las acciones mencionadas:

```
You have selected segmentation.
------------------------------

Process 0 with size 2 is searching for memory space.
Process 0 found space in memory.
Process 0 finished, its space is free.
Process 1 with size 6 is searching for memory space.
Process 1 found space in memory.
```

```
Shared memory ended.


Tue May 24 21:13:19 2022.txt is saved.

[valeria@localhost Simulacion-Memoria-Jo
```



```
Actual Time: 2022-05-24 21:11:51
Process 0 with size 2 is searching for memory space.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 21:11:51
Process 0 was inserted into the next memory location(s):
0, 1,
Actual array (-1 means free space):
0 0 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 21:12:13
Process 0 finished, its space  is free.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 21:12:44
Process 1 with size 6 is searching for memory space.
Actual array (-1 means free space):
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Actual Time: 2022-05-24 21:12:44
Process 1 was inserted into the next memory location(s):
0, 1, 2, 3, 4, 5,
Actual array (-1 means free space):
1 1 1 1 1 1 -1 -1 -1 -1
```

Diferencias entre mmap y shmget

En esta sección se demuestra la diferencia entre el uso de mmap y shmget, de manera en la que se puede concluir cuál es la mejor para la solución según los miembros del equipo.

Tanto el mmap como el método shmget sirven para lograr la creación y utilización de memoria compartida en el sistema. Se sabe que el método shmget es más restrictivo que el mmap, sin embargo, su desventaja es que no es más sencillo de utilizar. Entre sus principales ventajas es que puede utilizarse para procesos independientes, mientras que mmap se utiliza cuando hay posibilidad de presencia de hijos por medio de la función “fork()”.

Manual de usuario

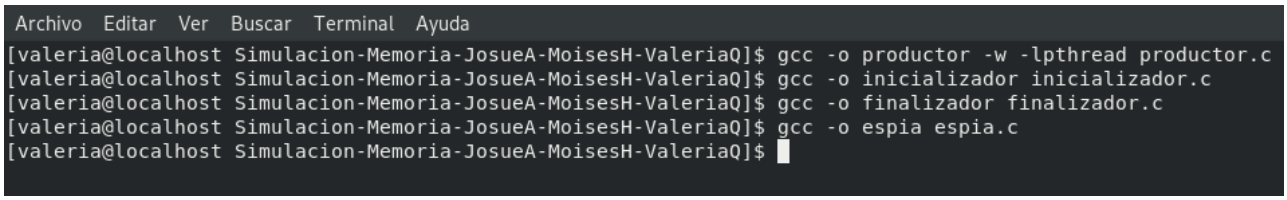
En esta sección se explica la manera en la que el usuario debe utilizar el programa, de manera que este no presente programas y sepa su correcto funcionamiento.

Lo primero que se debe realizar es descargar, en Linux, el programa encontrado. Ya sea por medio del .zip o desde el repositorio de github llamado:

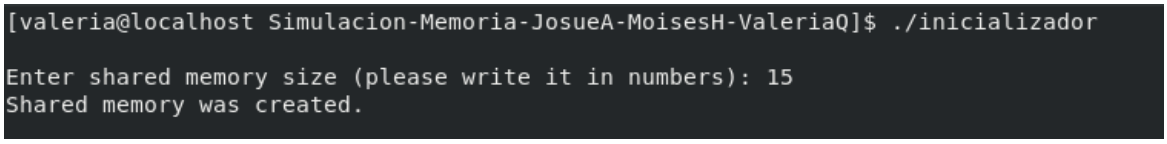
Simulacion-Memoria-JosueA-MoisesH-ValeriaQ

Para inicializar los programas por primera vez, se debe utilizar los comandos para crear un archivo ejecutable:

- gcc -o inicializador inicializador.c
- gcc -o productor -w -lpthread productor.c
- gcc -o espia espia.c
- gcc -o finalizador finalizador.c



Posteriormente, se pueden correr los diferentes archivos. El inicializador creará el espacio de memoria requerido, esto será posible con el comando “./inicializador”. Se le solicitará ingresar el tamaño de la memoria compartida, en donde se debe escribir en números y presionar la tecla “enter”. La terminal le notificará la creación de la memoria con el número ingresado.



Después, el productor creará los procesos a utilizar, esto será posible con el comando “./productor”. Se le presentará un menú en donde se seleccionará el tipo de partición de memoria, paginación o segmentación, debe seleccionar alguna de las

opciones (en número).

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./productor

Select a memory management scheme
1. Paging
2. Segmentation
2
```

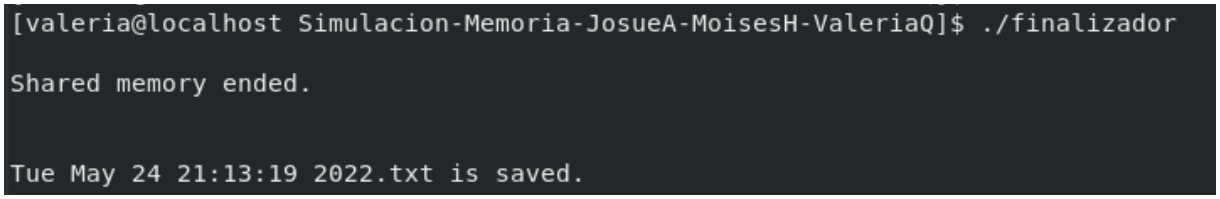
Al seleccionarse, se le mostrará al usuario su selección y seguidamente los procesos empezarán a ejecutarse. Se podrá observar el avance para que el usuario sepa qué es lo que se encuentran realizando.

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$ ./productor

Select a memory management scheme
1. Paging
2. Segmentation
1

You have selected paging.
------------------------------

Process 0 with size 1 is searching for memory space.
Process 0 found space in memory.
Process 1 with size 7 is searching for memory space.
Process 1 found space in memory.
Process 0 finished, its space is free.
Process 2 with size 10 is searching for memory space.
Process 2 found space in memory.
```

El programa espía permite la interacción directa con el usuario. Este se inicia con el comando “./espia” y mostrará un menú. En donde se puede seleccionar el ver el estado de todos los procesos actuales, los procesos en memoria, así como la opción de salida de este. Es importante mencionar que este último comando solo finalizará el programa espía, no finaliza la memoria compartida ni el resto de programas.

```
[valeria@localhost Simulacion-Memoria-JosueA-MoisesH-ValeriaQ]$
 ./espia

Select type of consultation
1. Memory state
2. Processes states
3. Exit
```

Por último, el finalizador elimina la memoria compartida. Solo se debe ejecutar el comando “./finalizador” para que esto suceda.



Bitácoras

En esta sección se encontrarán las bitácoras de los avances del proyecto.

Bitácoras individuales

En esta sección se mostrará el avance personal de cada uno de los integrantes del proyecto.

Estudiante: Josue Alvarado Mares

Fecha	Actividades	Comentarios / problema encontrado	Horas invertidas
01/05/2022	1-Investigar sobre semáforos en c linux 2-Investigar sobre la diferencia de mmap y shmget	1-El uso de semáforos y llaves 2-Mmap enfocado a mapear direcciones de memoria y shmget para obtención de memoria compartida.	1-1 Hr 30 Min 2-1 Hr

Estudiante: Moises Higuerey Hernandez

Fecha	Actividades y conclusiones	Comentarios / problema encontrado	Horas invertidas
04/05/22	Investigación sobre funciones mmap y shmget. Investigación sobre semáforos. La funcion shmget	Recursos utilizados: https://www.tutorialspoint.com/inter_process_communication/inter_process_communication_semaphores.htm https://stackoverflow.com/questions/8359322/how-to-share-semaphores-between-processes-using-shared-memory https://programmer.group/linux-c-c-shared-memory-with-message-queue-and-semaphore-encapsulation.html https://stackoverflow.com/questio	7:00pm-8:00pm

		ns/19464102/posix-shared-memory-initialization	
05/05/22	Investigación sobre acceso a memoria compartida en c con uso de semáforos.	Recursos utilizados: How to use POSIX semaphores in C language - GeeksforGeeks https://stackoverflow.com/questions/40181096/c-linux-pthreads-sending-data-from-one-thread-to-another-using-shared-memory-gi	2:00pm-2:30pm
13/05/22	Código de prueba para compartir estructuras a través de memoria compartida (structpass.c y structrecieve.c).	Se tuvieron problemas al intentar pasar punteros a la estructura por la memoria compartida.	4:10pm-5:00pm
24/05/22	Pruebas finales sobre el código.	Se probaron exitosamente las funcionalidades del proyecto.	2:00pm-3:00pm

Estudiante: Valeria Quesada Benavides

Fecha	Actividades y conclusiones	Comentarios / problema encontrado	Horas invertidas
02/05/22	Investigación sobre semáforos en C con sem.h: - semget ayuda a crear semáforos. Su sintaxis sería (llave,cantidad entrante,lectura/escritura | IPC_CREAT). - semctl inicializa el semáforo en 1. Su sintaxis es (semget,numero semáforo, SETVAL, el valor entrante). - fork() hereda. - semop es una llamada con la sintaxis (identificador del semáforo, sembuf, semáforos con los que se usa).	Se consultó: - Sistemas Operativos, semáforos en Linux	11:35am-11:52pm
02/05/22	Semáforos con semaphore.h:	Se consultó:	12:02pm-12:37pm

	- sem_t crea un semáforo sin nombre. - sem_init inicializa el semáforo compartido. - sem_wait disminuye el semáforo y bloquea el recurso. Es el inicio de la región crítica. - sem_post incrementa el semáforo, es decir, busca un desbloqueo del recurso compartido. Es el final de la región crítica. - sem_destroy destruye el semáforo. Función mmap(): - Mapeo de espacios virtuales, no de físicos. - Ayuda a que se mapeen los procesos y los acceda como una lista. - La función tiene 6 argumentos (dirección de inicio del mapeo, largo de bytes a mapear,protección, banderas, filedes, offset). - La protección presenta: PROT_READ | PROT_WRITE | PROT_EXEC | PROT_NONE. - La bandera puede ser: MAP_SHARED o MAP_PRIVATE (si otros procesos pueden verlo o no), MAP_ANONYMOUS / MAP_ANON (mapeo anónimo) y MAP_FIXED (el sistema debe utilizar obligatoriamente la dirección especificada). Función shmget(): - Se hace por medio de segmentos en donde se pueden crear, consultar y mapear. - Retorna un identificador de memoria asociado con una llave.	- Usar un semáforo en C - <semaphore.h> - Synchronizing Threads with POSIX Semaphores - How to use mmap function in C language? - mmap(2) — Linux manual page - shmget() — Get a shared memory segment - shmget(2) — Linux manual page	

	- Sus valores ayudan a indicar si algo ya se creó o no. Conclusiones: Aunque el shmget parece uno de los métodos más robustos, mmap puede ser uno de los más fáciles y rápidos de utilizar. shmget puede utilizarse para procesos independientes, mmap se suele utilizar cuando hay presencia de hijos (fork()).		
15/05/22	Se investigó sobre la posibilidad de cambiar y tomar los valores que se encuentran dentro del semáforo: - Se debe tener cuidado con las prioridades ya que pueden empezar a variar. - Existen funciones que trabajan similar al sem_wait, sin embargo, pueden retornar errores. - sem_trywait no guarda en cola los procesos que van llegando, lo que hace es retornar error si no se pueden ejecutar. - sem_timedwait sigue una estructura de tiempo de espera antes de que el proceso sea eliminado.	Se consultó: - sem_wait - sem_wait(3) - programación concurrente - usar semáforo en c	2:00pm - 3:00
16/05/22	Se investigó sobre cómo crear archivos para los programas: - El abrir archivos solo se puede hacer en un programa a la vez. - No se puede pasar por memoria los punteros de archivos abiertos. - En shm la memoria compartida de archivos no es posible. Se intentó programar un archivo compartido en memoria.	Se consultó: - Put a file into shared memory. - Load shared memory in C. - Reading files to shared memory.	1:00pm - 2:33pm
16/05/22	Se unió el código de prueba brindado por Moises con la tarea		8:02pm - 9:01pm

	programada. Se realizaron pruebas y fueron exitosas.		
17/05/22	Se analizó los links consultados en el 16/05/22. Se investigó sobre archivos compartidos: - El abrir y escribir no se puede realizar en diferentes programas. Se tomó la creación del archivo en el inicializador y su edición en el productor. Se escribe en la bitácora y no dan errores.	Se consultó: - Writing input from a file into shared memory.	1:32pm - 3:07pm
19/05/22	Se creó el código de escritura en la bitácora en el productor.		9:00pm-9:30pm
24/05/22	Se modificaron los mensajes y lo escrito en la bitácora, de forma en la que se pueden ver los detalles de estas. Se realizó un archivo de finalización en el programa finalizador, este presenta la fecha y hora en la que se ejecutó, de manera que la bitácora guarde su estado actual. Se realizaron pruebas en el código: - Se probó el inicializador si existe memoria y si no existe. - Se probó el finalizador si existe memoria y si no existe. - Se probó el espía si existe memoria y si no existe. - Se probó el productor si existe memoria y si no existe.		11:00am-2:00pm

Minutas (reuniones)

En esta sección se presentará lo abarcado en cada reunión grupal, mostrando los trabajos así como los integrantes.

Minuta-30/04/2022

1. Información de la reunión			
Fecha de la Reunión: (DD/MM/YYYY)	30/04/2022	**Hora:**	3:00 PM
Ubicación	Virtual		

2. Agenda de la Reunión		
Motivo de la reunión	**Agenda**	**Tiempo**
Proyecto #2 -SO	Leer el enunciado del proyecto y determinar actividades	1 Hr

3. Participantes	
Nombre	**Iniciales**
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

4. Puntos tratados			
Asunto	**Número**	**Propietario**	**Tiempo**
Lectura del enunciado	1	M.H, V.Q, J.A	1 hr

5. Acuerdos y declaraciones		
Compromisos	**Propietario**	**Fecha de Vencimiento**
Investigar sobre semáforos en C.	M.H, V.Q, J.A	02/05/2022

Investigar sobre memoria compartida mmap y shmget.	M.H, V.Q, J.A	02/05/2022
Investigar sobre la división de memoria en paginación.	M.H, V.Q, J.A	02/05/2022

6.Puntos pendientes
Descripción
N/A

7. Siguiente Reunión (Si aplica)					
Fecha: (DD/MM/YYYY)	02/05/2022	**Hora:**	3:00 PM	**Ubicación:**	Virtual
Objetivo:	Determinar próximas acciones a realizar para la realización del proyecto.				

8. Firma de los participantes	
Participante	Firma
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

Minuta-02/05/2022

1. Información de la reunión			
Fecha de la Reunión: (DD/MM/YYYY)	02/05/2022	**Hora:**	3:00 PM
Ubicación	Virtual		

2. Agenda de la Reunión		
Motivo de la reunión	**Agenda**	**Tiempo**
Proyecto #2 -SO	Análisis de los resultados de las investigaciones realizadas: Semáforos, mmap() y shmget(). RoadMap Análisis para código base	30 mins 20mins 10mins

3. Participantes	
Nombre	**Iniciales**
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

4. Puntos tratados			
Asunto	**Número**	**Propietario**	**Tiempo**
Análisis de investigaciones obtenidas.	1	M.H, V.Q, J.A	30 mins
Creación de roadmad	2	M.H, V.Q, J.A	20 mins
Análisis para código base	3	M.H, V.Q, J.A	10 mins

5. Acuerdos y declaraciones		
Compromisos	**Propietario**	**Fecha de Vencimiento**

Realizar y probar código base para semáforos con fock y shmget.	M.H, V.Q, J.A	05/05/2022

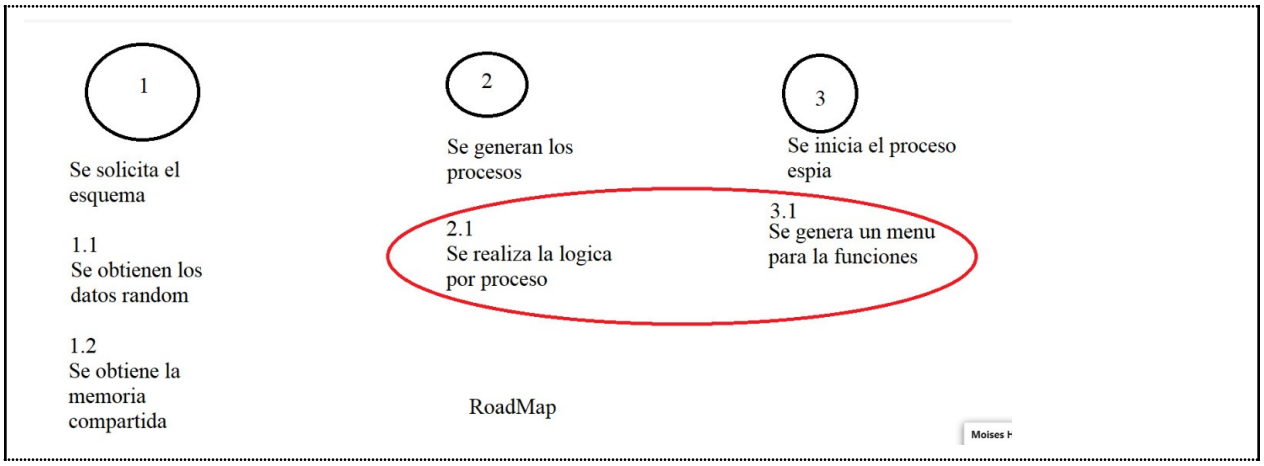
6. Puntos pendientes
Descripción
N/A

7. Bitácora		
Asunto #	**Actividad**	**Tiempo**
1	Cada uno de los propietarios expone su investigación.	30 mins (10 mins por participante)
2	Se crea un roadmap con el fin de entender las funcionalidades del proyecto y lo que se necesita.	20 mins
3	Se analiza cuál es la mejor función para la solución. Se concluye que se utilizará shmget ya que se considera la que más se ajusta a lo requerido.	1 hrs

8. Siguiente Reunión					
Fecha: (DD/MM/YYYY)	05/05/2022	**Hora:**	4:00 PM	**Ubicación:**	Virtual
Objetivo:	Analizar el código brindado por cada integrante				

9. Firma de los participantes	
Participante	Firma
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

9. Apéndices
RoadMap


1
Se solicita el esquema
1.1
Se obtienen los datos random
1.2
Se obtiene la memoria compartida
2
Se generan los procesos
2.1
Se realiza la logica por proceso
3
Se inicia el proceso espia
3.1
Se genera un menu para la funciones
RoadMap
Moises H

Minuta-05/05/2022

1. Información de la reunión			
Fecha de la Reunión: (DD/MM/YYYY)	05/05/2022	**Hora:**	3:45PM
Ubicación	Virtual		

2. Agenda de la Reunión		
Motivo de la reunión	**Agenda**	**Tiempo**
Revisión Códigos Base	Revisar Códigos propuestos por los integrantes para la inicialización de memoria.	35 mins

3. Participantes	
Nombre	**Iniciales**
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

4. Puntos tratados			
Asunto	**Número**	**Propietario**	**Tiempo**
Revisión de códigos.	1	M.H, V.Q, J.A	30mins

5. Acuerdos y declaraciones		
Compromisos	**Propietario**	**Fecha de Vencimiento**
Preguntarle a la profesora Erika sobre el uso de un arreglo para la memoria.	M.H, V.Q, J.A	06/05/2022

6. Puntos pendientes
Descripción
Agendar próxima reunión

7. Bitácora		
Asunto #	**Actividad**	**Tiempo**
1	Se solicitó a los integrantes de la reunión mostrar el código realizado junto con un pequeño resumen de este.	15 mins (5 mins por persona)
1	Se seleccionó por votación unánime el código de Moises por lo que se analizó a profundidad.	10 mins
1	Josué aclaró aspectos relevantes mencionados por la profesora Erika la clase anterior: - Cada proceso tiene un hilo. - El semáforo es el que se encarga de los procesos, no hay ningún hilo.	5 mins

8. Firma de los participantes	
Participante	Firma
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

Minuta-10/05/2022

1. Información de la reunión			
Fecha de la Reunión: (DD/MM/YYYY)	10/05/2022	**Hora:**	7:00 PM
	Virtual		

2. Agenda de la Reunión		
Motivo de la reunión	**Agenda**	**Tiempo**
Creación de semáforos en el código	Análisis de la mejor implementación de semáforos para el proyecto. Implementación de semáforos para el proyecto.	1:30 hrs

3. Participantes	
Nombre	**Iniciales**
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

4. Puntos tratados			
Asunto	**Número**	**Propietario**	**Tiempo**
Análisis de posible código para el semáforo.	1	M.H, V.Q, J.A	15 mins
Creación del código de semáforos.	2	M.H, V.Q, J.A	1: 15 hrs

5. Acuerdos y declaraciones		
Compromisos	**Propietario**	**Fecha de Vencimiento**
Preguntarle a la profesora si es posible crear colas para obtener información de los datos.	V.Q	11/05/2022

6. Puntos pendientes
Descripción
Hora y fecha sobre la próxima reunión.

7. Bitácora		
Asunto #	**Actividad**	**Tiempo**
1	Revisión de apuntes de clases para creación del mejor código y selección de tipo de semáforo.	10 mins
1	Se concluyó que lo ideal es utilizar los semáforos de tipo contador, en donde se utiliza como base la combinación de un algoritmo de la panadería.	5 mins
2	Se creó un código implementando el semáforo en donde se utilizó wait() y signal() para que se seleccionaran los procesos	1 hrs
2	Se crearon ejemplos para utilizarlos como prueba de que el código funcionaba y para buscar posibles errores. Estas pruebas fueron exitosas.	15 mins

8. Firma de los participantes	
Participante	Firma
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

Minuta-12/05/2022

1. Información de la reunión			
Fecha de la Reunión: (DD/MM/YYYY)	12/05/2022	**Hora:**	8:30 AM
Ubicación	Virtual		

2. Agenda de la Reunión		
Motivo de la reunión	**Agenda**	**Tiempo**
Creación de código	Creación de código para el proyecto.	1:30 hrs

3. Participantes	
Nombre	**Iniciales**
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

4. Puntos tratados			
Asunto	**Número**	**Propietario**	**Tiempo**
Creación del PCB.	1	M.H, V.Q, J.A	40 mins
Consulta de posible.	2	M.H, V.Q, J.A	5 mins
Creación del código de inicializador.	3	M.H, V.Q, J.A	15 mins

5. Acuerdos y declaraciones		
Compromisos	**Propietario**	**Fecha de Vencimiento**
Solicitarle a la profesora Erika una reunión para el 13 de mayo.	J.A	12/05/2022

Consultarle a la profesora Erika sobre si los programas se corren en diferentes terminales.	V.Q	12/05/2022
Creación de escribir y leer archivos.	V.Q	16/05/2022

6. Puntos pendientes
Descripción
Creación de diferentes programas, en caso de ser requerido.

7. Bitácora		
Asunto #	**Actividad**	**Tiempo**
1	Se acordó y realizó una estructura para la creación del PCB con el fin de acceder a la información de procesos.	40 mins
2	Consulta a la profesora sobre los diferentes programas.	5 mins
3	Se analizó el inicializador acorde al documento.Al tener la consulta sobre si los diferentes programas corren en diferentes terminales no se pudo realizar. Se acordó buscar una reunión con la profesora.	15 mins.

8. Firma de los participantes	
Participante	Firma
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

Minuta-15/05/2022

1. Información de la reunión			
Fecha de la Reunión: (DD/MM/YYYY)	15/05/2022	**Hora:**	2:00 PM
Ubicación	Virtual		

2. Agenda de la Reunión		
Motivo de la reunión	**Agenda**	**Tiempo**
Separación de programas	Separar el código según los programas asignados.	45 mins

3. Participantes	
Nombre	**Iniciales**
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

4. Puntos tratados			
Asunto	**Número**	**Propietario**	**Tiempo**
Separación del Inicializador.	1	M.H, V.Q, J.A	5 mins
Separación del productor.	2	M.H, V.Q, J.A	10 mins
Creación del finalizador.	3	M.H, V.Q, J.A	10 mins
Paso de parámetros de estado entre programas	4	M.H, V.Q, J.A	20 mins

5. Acuerdos y declaraciones		
Compromisos	**Propietario**	**Fecha de Vencimiento**
Investigar sobre cómo pasar structs en memoria compartida.	M.H, V.Q, J.A	15/05/2022

6. Puntos pendientes
Descripción
Preguntar si el size es variable.
Preguntar los espacios por segmento.

7. Bitácora		
Asunto #	**Actividad**	**Tiempo**
1	Se cortó el código del main para la creación del inicializador.	4 mins
1	Se buscó la diferencia del shmget y shm_open en este sitio. Se concluyó que se utilizaría shmget con la llave.	1 mins
2	Se separó el código del productor.	7 mins
2	Se probó la conexión entre el inicializador y el productor. Esta corrió con éxito.	3 mins
3	Se separó el código de finalizar memoria del inicializador.	10 mins
4	Investigación sobre pasar punteros entre memoria compartida.	20 mins

8. Firma de los participantes	
Participante	Firma
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

Minuta-17/05/2022

1. Información de la reunión			
Fecha de la Reunión: (DD/MM/YYYY)	17/05/2022	**Hora:**	7:00 PM
Ubicación	Virtual		

2. Agenda de la Reunión		
Motivo de la reunión	**Agenda**	**Tiempo**
Análisis y arreglo de la bitácora	Analizar el código encargado de la bitácora. Corregir el código encargado de la bitácora.	1 hrs

3. Participantes	
Nombre	**Iniciales**
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

4. Puntos tratados			
Asunto	**Número**	**Propietario**	**Tiempo**
Explicación del fallo y código de la bitácora.	1	M.H, V.Q, J.A	10 mins
Búsqueda de nuevas soluciones.	2	M.H, V.Q, J.A	50 mins

5. Acuerdos y declaraciones		
Compromisos	**Propietario**	**Fecha de Vencimiento**
Investigar sobre cómo solucionar el problema de la bitácora.	M.H, V.Q, J.A	19/05/2022

6. Puntos pendientes
Descripción
Esperar respuesta de la profesora sobre la reunión.

7. Bitácora		
Asunto #	**Actividad**	**Tiempo**
1	Se analizó el código y el por qué falló la memoria compartida en los archivos.	10 mins
2	Se buscó y analizó diferentes tipos de soluciones en donde se encontró: Sharing text file between processes. How to use shared memory with Linux in C. Share variables in different .c files. File pointer between files in C. Shared files in linux. Memoria compartida en C para linux.	50 mins

8. Firma de los participantes	
Participante	Firma
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

Minuta-18/05/2022 (Reunión 1)

1. Información de la reunión			
Fecha de la Reunión: (DD/MM/YYYY)	18/05/2022	**Hora:**	10:00 AM
Ubicación	Virtual		

2. Agenda de la Reunión		
Motivo de la reunión	**Agenda**	**Tiempo**
Revisión de código.	Revisión del código actual. Corrección de errores código actual.	1.30 hrs

3. Participantes	
Nombre	**Iniciales**
Josue Alvarado	J.A
Valeria Quesada	V.Q

4. Puntos tratados			
Asunto	**Número**	**Propietario**	**Tiempo**
Explicación del código de la bitácora realizado.	1	V.Q	5 mins
Análisis del código actual.	2	V.Q, J.A	10 mins
Análisis del documento del proyecto.	3	V.Q, J.A	25 mins
Corrección de errores y optimización del código actual.	4	V.Q, J.A	50 mins

5. Acuerdos y declaraciones		
Compromisos	**Propietario**	**Fecha de Vencimiento**
Agendar una nueva reunión con la profesora Erika.	J.A	18/05/2022

6. Puntos pendientes
Descripción
Aclarar dudas en la reunión con la profesora: - Espacio según tipo de simulación. - Si no hay espacio para el proceso muere o queda en espera de espacio?

7. Bitácora		
Asunto #	**Actividad**	**Tiempo**
1	Se explicó el código realizado en el productor.c de manera en la que se fue verificando que todo se encontraba en un orden correcto.	20 mins
2	Se buscó mejoras en donde se creó una función para la optimización de escritura en bitácora.	10 mins
3	Se leyó con detenimiento el documento brindado por la profesora en donde se clasificaron pendientes: - Ramdons - Sleep - Programa Espia	25 mins
4	Se creó la función de optimización del código encargada de escribir en la bitácora y realizar prints en la terminal.	10 mins
4	Se investigó sobre cómo agregar fecha y hora y se añadió en la bitácora.	25 mins
4	Se realizaron pruebas para observar la velocidad y si la optimización había funcionado. Fueron exitosas, sin embargo, se modificó la escritura por bloques para que fuera más entendible.	15 mins

8. Firma de los participantes	
Participante	Firma
Josue Alvarado	J.A
Valeria Quesada	V.Q

Minuta-18/05/2022 (Reunión 2)

1. Información de la reunión			
Fecha de la Reunión: (DD/MM/YYYY)	18/05/2022	**Hora:**	4:40 PM
Ubicación	Virtual		

2. Agenda de la Reunión		
Motivo de la reunión	**Agenda**	**Tiempo**
Revisión de avance.	Revisión de avance del proyecto con la profesora Erika	20 mins

3. Participantes	
Nombre	**Iniciales**
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

4. Puntos tratados			
Asunto	**Número**	**Propietario**	**Tiempo**
Aclaración de dudas.	1	M.H, V.Q, J.A	10 mins
Muestra del código.	2	M.H, V.Q, J.A	10 mins

5. Acuerdos y declaraciones		
Compromisos	**Propietario**	**Fecha de Vencimiento**
Arreglar bitácora.	V.Q	19/05/2022

6. Puntos pendientes
Descripción
N/A

7. Bitácora		
Asunto #	**Actividad**	**Tiempo**
1	Se aclararon dudas con la profesora: - Lo ideal para ver la memoria es tener un arreglo de los procesos. - En la segmentación los procesos pueden tener diferentes segmentos en donde se abarquen diferentes espacios de memoria. - La bitácora no debe permanecer abierta entre programas, se debe abrir y cerrar en cada proceso.	10 mins
2	Se le mostró el código a la profesora.	10 mins

8. Firma de los participantes	
Participante	Firma
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

Minuta-19/05/2022 (Reunión 1)

1. Información de la reunión			
Fecha de la Reunión: (DD/MM/YYYY)	19/05/2022	**Hora:**	10:00 AM
Ubicación	Virtual		

2. Agenda de la Reunión		
Motivo de la reunión	**Agenda**	**Tiempo**
Arreglo del código de paginación y segmentación	Corregir el código de paginación y segmentación.	3.00 hrs

3. Participantes	
Nombre	**Iniciales**
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

4. Puntos tratados			
Asunto	**Número**	**Propietario**	**Tiempo**
Corrección del código de paginación.	1	M.H, V.Q, J.A	1.30 hrs
Corrección del código de segmentación	2	M.H, V.Q, J.A	1.30 hrs

5. Acuerdos y declaraciones		
Compromisos	**Propietario**	**Fecha de Vencimiento**
N/A		

6. Puntos pendientes
Descripción
N/A

7. Bitácora		
Asunto #	**Actividad**	**Tiempo**
1	Se arregló el código de paginación que se tenía previamente: - Se tomó como base la segmentación que ya se tenía. - Se agregó creación de tamaños de procesos por página. - Se creó un random para las pausas de creación entre procesos y su duración en memoria.	1.10 hrs
1	Se creó una serie de pruebas con el código de paginación en donde se analizó con detenimiento los random y comportamientos de procesos. No existieron errores.	20 mins
2	Se arregló el código de segmentación que se tenía previamente: - Se tomó como base la segmentación que ya se tenía pero se agregó la separación por segmentos junto con sus tamaños. - Se creó un random para las pausas de creación entre procesos y su duración en memoria.	50 mins
2	Se probó la función para la asignación de espacios. Se observó que hay errores en la asignación de memoria por lo que el código se cambió. Al ejecutar una serie de pruebas se demostró que la corrección fue exitosa.	40 mins

8. Firma de los participantes	
Participante	Firma
Josue Alvarado	J.A

Valeria Quesada	V.Q
Moises Higuerey	M.H

Minuta-19/05/2022 (Reunión 2)

1. Información de la reunión			
Fecha de la Reunión: (DD/MM/YYYY)	19/05/2022	**Hora:**	7:00 PM
Ubicación	Virtual		

2. Agenda de la Reunión		
Motivo de la reunión	**Agenda**	**Tiempo**
Revisión de código para paso de datos	Adaptar el código actual en donde se permita compartir de manera optimizada los datos entre diferentes programas.	1.10 hrs

3. Participantes	
Nombre	**Iniciales**
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

4. Puntos tratados			
Asunto	**Número**	**Propietario**	**Tiempo**
Creación de una cola basada en el PCB.	1	M.H, V.Q, J.A	1 hrs
Paso del tamaño de memoria.	2	M.H, V.Q, J.A	10 mins

5. Acuerdos y declaraciones		
Compromisos	**Propietario**	**Fecha de Vencimiento**
N/A		

6. Puntos pendientes
Descripción
N/A

7. Bitácora		
Asunto #	**Actividad**	**Tiempo**
1	Se adaptó el código actual de manera en la que los procesos quedarán guardados como una cola que se pueda compartir.	45 mins
1	Se crearon una serie de pruebas con los códigos. Se vió el error que el estado no se actualizaba y se corrigió haciendo referencia a los punteros de manera en la que se actualizaron.	15 mins
2	Se creó una memoria compartida temporal en donde el tamaño pasa del inicializador al productor.	50 mins
2	Se probó el código actual. Tanto el paso de datos como la actualización de las colas. Esto fue exitoso.	10 mins

8. Firma de los participantes	
Participante	Firma
Josue Alvarado	J.A
Valeria Quesada	V.Q
Moises Higuerey	M.H

Minuta-21/05/2022

1. Información de la reunión			
Fecha de la Reunión: (DD/MM/YYYY)	21/05/2022	**Hora:**	3:00 PM
Ubicación	Virtual		

2. Agenda de la Reunión		
Motivo de la reunión	**Agenda**	**Tiempo**
Creación del PCB compartido entre programas.	Crear un PCB compartido entre programas.	3 hrs

3. Participantes	
Nombre	**Iniciales**
Josue Alvarado	J.A
Moises Higuerey	M.H

4. Puntos tratados			
Asunto	**Número**	**Propietario**	**Tiempo**
Creación de un array compartido.	1	M.H, J.A	2.30 hrs
Conexión del array en tiempo real por medio de un hilo.	2	M.H, J.A	30 mins

5. Acuerdos y declaraciones		
Compromisos	**Propietario**	**Fecha de Vencimiento**
N/A		

6. Puntos pendientes
Descripción
N/A

7. Bitácora		
Asunto #	**Actividad**	**Tiempo**
1	Se creó un struct para el arreglo.	10 mins
1	Se creó una memoria compartida y se insertó el arreglo en el productor. Posteriormente se probó, sin embargo, la conexión no fue exitosa por lo que se analizó el código y se encontró la solución, el fallo radica en que la memoria se borraba constantemente.	2.20 hrs
2	Se creó un hilo con el fin de que el arreglo se actualice de manera constante en el programa. Al correrlo, las pruebas fueron exitosas.	30 mins

8. Firma de los participantes	
Participante	Firma
Josue Alvarado	J.A
Moises Higuerey	M.H

Bibliografía

Bae, H. (2016). Reading files to shared memory. Stack Overflow. Recuperado el 16 de mayo del 2022. https://stackoverflow.com/questions/37163438/reading-files-to-shared-memory.

C, H. (2011). How to use shared memory with Linux in C. Stack Overflow. Retrieved 16 May 2022, from https://stackoverflow.com/questions/5656530/how-to-use-shared-memory-with-linux-in-c.

Ghosh, B. (2020). How to use mmap function in C language?. Recuperado el 2 de mayo del 2022. https://linuxhint.com/using_mmap_function_linux/.

González, A. (2013). Programación Concurrente. Recuperado el 2 de mayo del 2022. https://www.ctr.unican.es/asignaturas/mc_procon/Doc/ProCon_II_06-sincronizacion_3en1.pdf

How do I share variables between different .c files?. (2013). Recuperado el 17 de mayo del 2022. https://stackoverflow.com/questions/1045501/how-do-i-share-variables-between-different-c-files

Kadosh, Y. (2015). How to share semaphores between processes using shared memory. Recuperado el 4 de mayo del 2022. https://stackoverflow.com/questions/8359322/how-to-share-semaphores-between-processes-using-shared-memory

Khintibidze, L. (2022). Usar un semáforo en C. Delft Stack. Recuperado el 2 de mayo del 2022. https://www.delftstack.com/es/howto/c/semaphore-example-in-c/#:~:text=Un%20sem%C3%A1foro%20es%20un%20n%C3%BAmero,y%20liberar%20el%20recurso%20compartido.

How to use POSIX semaphores in C language. (2020). Recuperado el 5 de mayo del 2022. https://www.geeksforgeeks.org/use-posix-semaphores-c/

IBM Docs. Ibm.com. (2022). Recuperado el 2 de mayo del 2022. https://www.ibm.com/docs/en/zos/2.1.0?topic=functions-shmget-get-shared-memory-segment.

Inter Process Communication. (2022). Recuperado el 4 de mayo del 2022. https://www.tutorialspoint.com/inter_process_communication/inter_process_communication_semaphores.htm

Leffler, J. (2016). Recuperado el 5 de mayo del 2022. https://stackoverflow.com/questions/40181096/c-linux-pthreads-sending-data-from-one-thread-to-another-using-shared-memory-gi

Linux C/C++ shared memory with message queue and semaphore encapsulation. (2021) Recuperado el 4 de mayo del 2022. https://programmer.group/linux-c-c-shared-memory-with-message-queue-and-semaphore-encapsulation.html

P. (2013). Unix/C: put a file into shared memory. Stack Overflow. Recuperado el 16 de mayo del 2022. https://stackoverflow.com/questions/14164316/unix-c-put-a-file-into-shared-memory.

mmap(2) - Linux manual page. Man7.org. (2022). Recuperado el 2 de mayo del 2022. https://man7.org/linux/man-pages/man2/mmap.2.html.

POSIX Semaphores. Csc.villanova.edu. (2022). Recuperado el 2 de mayo del 2022. http://www.csc.villanova.edu/~mdamian/threads/posixsem.html#wait.

Shared Files in Linux. (2016). Recuperado el 17 de mayo del 2022. https://stackoverflow.com/questions/72282442/shared-files-in-linux

shmget vs. shm_open. Comp.os.linux.development.apps.narkive.com. (2007). Recuperado el 15 de mayo del 2022. https://comp.os.linux.development.apps.narkive.com/sPuJ4fI1/shmget-vs-shm-open.

shmget(2) - Linux manual page. Man7.org. (2022). Recuperado el 2 de mayo del 2022. https://man7.org/linux/man-pages/man2/shmget.2.html.

<semaphore.h>. Pubs.opengroup.org. (2022). Recuperado el 2 de mayo del 2022. https://pubs.opengroup.org/onlinepubs/007908775/xsh/semaphore.h.html.

sem_wait. Pubs.opengroup.org. (2022). Recuperado el 2 de mayo del 2022. https://pubs.opengroup.org/onlinepubs/7908799/xsh/sem_wait.html.

[Solved] Linux shared memory: shmget() vs mmap()? - Local Coder. Localcoder.org. (2022). Recuperado el 2 de mayo del 2022. https://localcoder.org/linux-shared-memory-shmget-vs-mmap.

Vogt, A. (2013). Posix shared memory initialization. Recuperado el 4 de mayo del 2022. https://stackoverflow.com/questions/19464102/posix-shared-memory-initialization

Writing input from a file into shared memory. Cboard.cprogramming.com. (2004). Recuperado el 16 de mayo del 2022. https://cboard.cprogramming.com/c-programming/57041-writing-input-file-into-shared-memory.html.

C, H., Giesen, F., Knight, N., Melo, C., & Anderson, J. (2022). *How Do I Store and Retrieve a Struct into a Shared Memory Area in C*. Stack Overflow. Retrieved 24 May 2022, from https://stackoverflow.com/questions/4263444/how-do-i-store-and-retrieve-a-struct-into-a-shared-memory-area-in-c.

C, H. (2022). *How to use shared memory with Linux in C*. Stack Overflow. Retrieved 24 May 2022, from https://stackoverflow.com/questions/5656530/how-to-use-shared-memory-with-linux-in-c.

Shared Memory. Tutorialspoint.com. (2022). Retrieved 24 May 2022, from https://www.tutorialspoint.com/inter_process_communication/inter_process_communication_shared_memory.htm.