# Proyecto de Sistemas Operativos

## 1.1 Procesos, threads y Comunicación

### 1.1.1 Banco

Para la solución del problema se plantearon diferentes funciones según el tipo de cliente.
En el caso de que sea un cliente común o de empresa, se utilizan las funciones cliente_comun() y cliente_empresa() respectivamente, las cuales mediante el uso de semáforos simularán el comportamiento de los clientes dentro del banco.
Por otro lado, si se trata de un cliente político, se usa la función cliente_politico() que añade la prioridad que se le otorga a un político en el banco.
La idea principal es simular cada cola del banco con un semáforo, de tal manera que cada cliente va ir decrementando el semáforo y esperará hasta acceder a la próxima cola liberarando la cola anterior, es decir, incrementa el semáforo correspondiente. Una vez que esté en la cola de su categoría consultara a un empleado del banco el cual va a ser simulado también por un semáforo, y si puede procesar su solicitud libera la cola de su categoría, es atendido y luego se retira del banco.

Con respecto a ventajas y desventaja de las implementación podemos destacar:

Cuando usamos hilos y semáforos las ventajas principales son la facilidad para la implementación, el alto rendimientos por que resulta menos costoso la creación de hilos que de procesos así como la cantidad de cambios de procesos que se generan entre ellos. Otro aspecto importante es la comunicación entre los hilos ya que comparten recursos de forma más sencilla. Como punto a criticar es la mala utilización de semáforos por que el uso incorrecto de los mismos no llevarían a situaciones no deseadas o de interbloqueo.

Para la segunda implementación con procesos elegimos utilizar colas de mensajes sobre pipes por que los pipes tienen lectura bloqueante y para nuestra implementación no resultaba tan bien y como las colas de mensajes si pueden realizar lecturas no bloqueantes optamos por su elección.
Con respecto, a ventaja se puede destacar la robustez que nos brindan los procesos ya que cada proceso tiene su espacio de memoria y nos llevaría a condiciones de carrera innecesarias, además, si un proceso falla el resto no se vería afectado. Como punto negativo, los procesos requieren mucho más recursos. Además, la comunicación entre procesos a través de colas de mensajes generalmente tiene un mayor costo en términos de tiempo y recursos que la comunicación entre hilos en un mismo proceso. Otro punto no favorable es la complejidad de programación, ya que resulta más engorroso a la hora de implementarlo.
Para la ejecución de todos los códigos, dentro de cada carpeta de los ejercicios se va a encontrar un script llamado run.sh el cual se va a encargar de compilar y ejecutar cada código, mediante la instrucción “./run.sh”.

### 1.1.2 Mini Shell

La implementación del problema de la mini shell consiste en simular operaciones básicas de una shell, en este caso imitando el comportamiento de la consola de Unix. Al momento de implementarlo, la opción elegida fue separar todos los archivos de los comandos del archivo principal. Podemos destacar los siguiente archivo donde se encuentran los comando y el archivo principal para lanzar la minishell:

-**shell**:: archivo principal, lee la entrada del usuario y verifica si es válida, si lo es, ejecuta el comando correspondiente,se crea un nuevo proceso y se realiza un execv donde se carga la imagen ejecutable del comando correspondiente.

-**cat:**:permite concatenar y mostrar el contenido de archivos.

-**chmod**:: permite cambiar los permisos de lectura, escritura y ejecución de un archivo.

-**clear**:: utilizado para limpiar los comandos en la terminal, borrar el texto de la pantalla o consola.

-**help**:: en caso de que no se indique un comando en particular,muestra un listado de comandos disponibles. De otra manera, muestra la funcionalidad de un comando específico.

-**ls**:: se utiliza para listar archivos o directorios.

- **mkdi**r:: crea un nuevo directorio.

-**rm**:: elimina archivos o directorios.

-**touch**:: crea archivos vacíos.

Para la ejecución del código, dentro de la carpeta de los ejercicios se encuentra un script llamado run.sh el cual se va a encargar de compilar y ejecutar cada código, mediante la instrucción “./run.sh”.

## 1.2 Sincronización

### 1.2.1 Secuencia

La idea general es tener un proceso o hilo por cada letra e ir sincronizando cada proceso haciendo uso de semáforos como de pipes de modo que se vaya respetando la secuencia. Para la ejecución de todos los códigos, dentro de cada carpeta de los ejercicios se va a encontrar un script llamado run.sh el cual se va a encargar de compilar y ejecutar cada código, mediante la instrucción “./run.sh”.

### 1.2.2 Reserva de aula

Para la resolución de este problema se tiene 3 tareas principales que puede realizar cada alumno, reservar, cancelar y consultar. Las dos primeras se va a necesitar un lock para que evitar que cuando un alumno esa reservando otro que quiere reservar no pise su reserva. Otra situación que se puede presentar es cuando alguien quiere reservar y otra quiere cancelar, por tal motivo, se hace uso de un lock para no llevar a situaciones extrañas.
Para la ejecución de todos los códigos, dentro de cada carpeta de los ejercicios se va a encontrar un script llamado run.sh el cual se va a encargar de compilar y ejecutar cada código, mediante la instrucción "./run.sh".

## 2 Problemas

### 2.2.1 Lectura(Embedded Linux)

Para este ejercicio optamos por la idea de realizar un podcast donde se plantea el diálogo entre 2 personas, donde una de ella está intentando aprender más sobre el sistema operativo y la otra persona le va contando los detalles más relevantes del mismo. Toda la información presentada en el mismo fue extraída del libro *Embedded Linux System Design and Development* de los autores P. Raghavan, Amol Lad y Sriram Neelakandan.
El audio y el guión del mismo se encuentra en su respectiva carpeta dentro del proyecto donde el audio esta como podcast.mp3 y el guión como guión.pdf

### 2.2.2 Problemas Conceptuales

**1. Considere un sistema de gestión de memoria basado en paginación. El tamaño total de la memoria física es de 2 GB, distribuido en páginas de tamaño 8 KB. El espacio de direcciones lógicas de cada proceso se ha limitado a 256 MB.**

**a) Determine el número total de bits en la dirección física.**
**b) Determine el número de bits que especifican la sustitución de página y el número de bits para el número de marco de página.**
**c) Determine el número de marcos de página.**
**d ) Determine el formato de la dirección lógica.**

a) Para determinar el número total de bits en la dirección física, calculamos en bytes el tamaño de la memoria física:

Tamaño de la memoria física = 2 GB = 2 * 1024 MB = 2 * 1024 * 1024 KB = 2 * 1024 * 1024 * 1024 bytes = **2.147.483.648 bytes**.

Ahora, para determinar los bits totales de la dirección física , aplicamos el $\log_2$(tamaño de la memoria física):

$\log_2$(2147483648 bytes) = **31 bits**

Por lo tanto, llegamos a la conclusión que se necesitan 31 bits para la dirección física.

b) Para determinar la cantidad de bits para especificar la sustitución de página, debemos calcular:

Tamaño de página = 8 KB = 8 * 1024 bytes = **8192 bytes**.

$\log_2$(8192 bytes) = 13 bits.

El número necesarios de bits para la sustitución de páginas es **13 bits**.

m = el número de bits total de la memoria física = 31 bits.

n = el número de bits para la sustitución de páginas = 13 bits.

p = número de bits para el número de marcos de página = m - n = 31 - 13 = 18 bits.

El número necesario de bits para los marcos de página es **18 bits.**

**Dirección física**

| 18 Bits | 13 Bits |
|---|---|
| **Número de marco de páginas** | **Sustitución de páginas** |

c) Para calcular el número de marcos de paginas, necesitamos realizar el siguiente cálculo:

Número de marcos de página = Tamaño de la memoria física / Tamaño de páginas

**2.147.483.648 bytes / 8192 bytes = 262144 marcos**

La cantidad de marcos es 262144

d) Para determinar el formato de la dirección lógica debemos realizar los siguientes cálculos:

Tamaño de dirección lógica = 256 Mb  = 268.435456 bytes.

Tamaño de página = 8 Kb= 8192 bytes

Número de bit total de la dirección lógica = $\log_2$(268.435456 bytes ) = 28 bits

Número de bit de sustitución de páginas = $\log_2$(8192 bytes) = 13 bits

Número de bit de páginas virtuales = 28 - 13 = 15 bits

**Dirección lógica**

| 15 Bits | 13 Bits |
|---|---|

**Numero de pagina virtual** **Sustitución de páginas**

**2. Considera un sistema de segmentación simple que tiene la siguiente tabla de segmentos**

| Dirección Inicial | Largo (bytes) |
|---|---|
| 830 | 346 |
| 648 | 110 |
| 1508 | 408 |
| 770 | 812 |

**Para cada una de las siguientes direcciones lógicas, determina la dirección física o indica si se produce un fallo de segmento:**

**a) 0, 228**
**b) 2, 648**
**c) 3, 776**
**d ) 1, 98**
**e) 1, 240**

a) Dirección lógica (0, 228):

- El índice del segmento es 0, lo que significa que estamos buscando en el primer segmento.
- La dirección inicial del primer segmento es 830, y su longitud es 346 bytes.
- La dirección lógica 228 está dentro de los límites de este segmento.
- La dirección física sería la dirección inicial del segmento (830) más la dirección lógica dentro del segmento (228), es decir, 830 + 228 = 1058.

b) Dirección lógica (2, 648):

- El índice del segmento es 2, lo que significa que estamos buscando en el tercer segmento.
- La dirección inicial del tercer segmento es 1508, y su longitud es 408 bytes.
- La dirección lógica 648 está fuera de los límites de este segmento, lo que resulta en un fallo de segmento.

c) Dirección lógica (3, 776):

- El índice del segmento es 3, lo que significa que estamos buscando en el cuarto segmento.
- La dirección inicial del cuarto segmento es 770, y su longitud es 812 bytes.
- La dirección lógica 776 está dentro de los límites de este segmento.
- La dirección física sería la dirección inicial del segmento (770) más la dirección lógica dentro del segmento (776), es decir, 770 + 776 = 1546.

d) Dirección lógica (1, 98):

- El índice del segmento es 1, lo que significa que estamos buscando en el segundo segmento.
- La dirección inicial del segundo segmento es 648, y su longitud es 110 bytes.
- La dirección lógica 98 está dentro de los límites de este segmento.
- La dirección física sería la dirección inicial del segmento (648) más la dirección lógica dentro del segmento (98), es decir, 648 + 98 = 746.

e) Dirección lógica (1, 240):

- El índice del segmento es 1, lo que significa que estamos buscando en el segundo segmento.
- La dirección inicial del segundo segmento es 648, y su longitud es 110 bytes.
- La dirección lógica 240 está fuera de los límites de este segmento, lo que resulta en un fallo de segmento.