Proyecto Sistemas Operativos 2023 – Comisión 42

▶ Integrantes:

- Nahuel Diaz
- Sabrina Barrionuevo

1.1 Procesos, threads y Comunicación.

1. BANCO.

a) Para el problema del "Banco" implementado con hilos y semáforos, planteamos una solución utilizando el siguiente esquema:

Creación de hilos: Tanto los empleados como los clientes de cualquier tipo son hilos. En nuestra solución creamos 100 hilos para los clientes y 3 hilos para los empleados. A cada uno de los 100 hilos clientes se les asigna una identidad aleatoria, y dependiendo de si son usuarios, políticos o empresas se ejecutarán en su función correspondiente. En cuanto a los tres empleados, dos hilos ejecutaran la función `empleadosEmpresas`, y el hilo restante ejecuta `empleadoUsuario`.

Los tres empleados tienen un número de identificación (0, 1 y 2), donde, 0 y 1 atienden empresas, 2 atiende usuarios y todos atienden políticos. Los clientes por su lado también tienen un número que los identifica, al momento de crear los hilos, usamos la variable de iteración del ciclo `for` pasándosela al hilo como argumento. El objetivo de esto es que cada cliente tenga un número único asignado para lograr un mejor seguimiento de él durante la ejecución.

Comportamiento de los clientes: Cuando un cliente llega al banco lo primero que hace es tratar de hacer un wait() en el semáforo sem_mesa que representa los lugares disponibles en la mesa de entrada. Si sem_mesa es 0, significa que no hay lugar en la mesa, entonces el cliente se retira. En caso contrario, decrementa el semáforo e ingresa a la mesa.

Hay 6 semáforos que controlan la cantidad de clientes en las filas de espera de cada tipo:

- `sem_filaUsu_vacio`
- `sem_filaEmp_vacio`
- `sem_filaPol_vacio`

Representan los lugares libres en las filas.

- `sem_filaUsu_lleno`
- `sem_filaEmp_lleno`
- `sem_filaPol_lleno`

Representan los lugares ocupados en las filas.

Después de que un cliente ingreso a la mesa de entrada se queda esperando que haya un lugar vacío en su correspondiente fila, y una vez que esto ocurre, decrementa los lugares vacíos en la fila, incrementa los lugares vacíos en la mesa de entrada y los lugares ocupados en la fila. Una vez en la fila espera a ser atendido.

Comportamiento de los empleados: Un empleado cualquiera lo primero que va a hacer es revisar el semáforo de los lugares ocupados en la fila de los políticos (`sem_filaPol_lleno`), si este semáforo no se puede decrementar significa que no hay políticos esperando ser atendidos, en este caso, los empleados buscaran atender a empresas o usuarios, según les corresponda.

Hay 6 semáforos para modelar la atención a un cliente:

- `sem_empleadoUsu`
- `sem_empleadoEmp`

- `sem_empleadoPol`

Representan el inicio de una atención.

- `sem_empresa_atendida`
- `sem_usuario_atendido`
- `sem_politico_atendido`

Representan el final de una atención.

Cuando un empleado va a atender un cliente de cualquier tipo, incrementa su semáforo de empleado avisándole a un cliente que puede dejar de esperar y pasar a la atención. Ahora el cliente esta siendo atendido, y cuando la atención termina el empleado incrementa el semáforo de atendidos para que el hilo cliente continue su ejecución y se retire.

Captura de un trozo de la ejecución:

b) Para el problema del "Banco" implementado con procesos y colas de mensajes, planteamos una solución utilizando el siguiente esquema:

Creación de procesos y colas: Nuestra solución se basa en utilizar los tipos de mensajes y las colas como si fueran varios semáforos, y controlar así la concurrencia de procesos. Hay 100 procesos clientes que atraviesan el banco, y 3 procesos empleados atendiendo clientes. Igual que en el inciso anterior, los procesos empleados y clientes tienen un numero asignado que identifica a cada uno. Como antes, decidimos que los empleados 0 y 1 atiendan empresas y políticos, y el empleado 2 atienda usuarios y políticos.

Definimos muchos tipos de mensajes:

- `#define mesa_entrada 1`
- `#define fila_usuarios_vacio 2`
- `#define fila_usuarios_lleno 3`
- `#define fila_politicos_vacio 4`
- `#define fila_politicos_lleno 5`
- `#define fila_empresas_vacio 6`
- `#define fila_empresas_lleno 7`
- `#define usuario_atendido 8`
- `#define empleado_usuario 9`
- `#define politico_atendido 10`
- `#define empleado_politico 11`
- `#define empresa_atendida 12`
- `#define empleado_empresa 13`

Cada uno de estos tipos de mensajes cumplen la misma función que los semáforos del inciso a). Por ejemplo, `fila_usuarios_vacio` y `fila_usuarios_lleno`, representan los lugares libres y ocupados en la fila de usuarios, `politico_atendido` representa la finalización de una atención a un político, y así.

Antes de crear los procesos y mandarlos a sus respectivas funciones, se deben enviar algunos mensajes a la cola, tantos como sea necesario para representar todos los lugares disponibles al comienzo de la ejecución. Es por eso que con un bucle `for` enviamos 30 mensajes de tipo `mesa_entrada` y 15 mensajes de tipo `fila_usuarios_vacio`, `fila_empresas_vacio` y `fila_politicos_vacio`.

Comportamiento de los clientes: Entonces, cuando un cliente llega al banco lo primero que hace es intentar recibir un mensaje de tipo `mesa_entrada`. Si el mensaje es recibido con éxito, significa que hay un lugar vacío menos en la mesa y que un cliente entro, en caso contrario, es decir, si no hay ningún mensaje de tipo `mesa_entrada` significa que no hay lugares vacíos y el cliente se retirara.

Una vez dentro del banco, en la mesa de entrada, el cliente intentara recibir un mensaje de tipo `fila_(tipo)_vacio`, pero en este caso, el proceso se va a bloquear hasta que pueda recibir un mensaje de ese tipo, que es lo mismo que decir, que el cliente va a esperar en la mesa de entrada hasta que haya un lugar libre en la fila de su tipo.

Una vez recibido el mensaje `fila_(tipo)_vacio`, el proceso ya esta en la fila de espera para ser atendido, por eso debe enviar un mensaje de tipo `fila_(tipo)_lleno` para indicarle a los empleados que hay lugares ocupados en alguna de las filas de espera.

Comportamiento de los empleados: Los tres empleados intentar recibir un mensaje de tipo `fila_politicos_lleno` antes que cualquier otro, ya que, en caso de recibir un mensaje, significa que hay un político esperando ser atendido y como su prioridad es mayor, el empleado debe atenderlo. Solo cuando no haya mensajes de tipo `fila_politicos_lleno`, es que los empleados atienden a empresas o usuarios, según corresponda.

Al momento de atender un cliente, apenas el empleado recibe el mensaje de que hay alguien esperando, envía un mensaje de tipo `empleado_(tipo)` que va ser recibido por el cliente que está esperando la atención, esto le indica que va a ser atendido.

Cuando la atención termina, el cliente envía un mensaje de tipo `(tipo)_atendido` para que el empleado lo reciba y termine de atenderlo. Luego, el cliente se retira, no sin antes enviar un mensaje de tipo `fila_(tipo)_vacio` para indicar que libera su lugar en la fila.

Captura de un trozo de la ejecución:

```
raspbian01@raspberry:~/MyCodes $ ./compilado
Politico 1 llega al banco y quiere entrar.
El politico 1 ingresa a la mesa de entrada.
El politico 1 ingresa a la fila de atencion de politicos.
Usuario 9 llega al banco y quiere entrar.
El usuario 9 ingresa a la mesa de entrada.
El usuario 9 ingresa a la fila de atencion de usuarios.
Un empleado esta atendiendo un usuario. (id_empleado: 2)
El usuario 9 es atendido
El empleado termino de atender un usuario. (id_empleado: 2)
Un empleado esta atendiendo un politico. (id_empleado: 2)
El usuario 9 fue atendido y se retira.
El politico 1 es atendido.
El empleado termino de atender un politico. (id_empleado: 2)
El politico 1 fue atendido y se retira.
Politico 7 llega al banco y quiere entrar.
El politico 7 ingresa a la mesa de entrada.
El politico 7 ingresa a la fila de atencion de politicos.
Empresa 3 llega al banco y quiere entrar.
La empresa 3 ingresa a la mesa de entrada.
La empresa 3 ingresa a la fila de atencion de empresas.
Un empleado esta atendiendo una empresa. (id_empleado: 0)
La empresa 3 es atendido
El empleado termino de atender una empresa. (id_empleado: 0)
Un empleado esta atendiendo un politico. (id_empleado: 0)
La empresa 3 fue atendido y se retira.
El politico 7 es atendido.
El empleado termino de atender un politico. (id_empleado: 0)
El politico 7 fue atendido y se retira.
Empresa 10 llega al banco y quiere entrar.
La empresa 10 ingresa a la mesa de entrada.
La empresa 10 ingresa a la fila de atencion de empresas.
Un empleado esta atendiendo una empresa. (id_empleado: 1)
Empresa 12 llega al banco y quiere entrar.
La empresa 12 ingresa a la mesa de entrada.
La empresa 12 ingresa a la fila de atencion de empresas.
Usuario 8 llega al banco y quiere entrar.
El usuario 8 ingresa a la mesa de entrada.
El usuario 8 ingresa a la fila de atencion de usuarios.
Un empleado esta atendiendo un usuario. (id_empleado: 2)
Un empleado esta atendiendo una empresa. (id_empleado: 0)
La empresa 12 es atendido
Empresa 13 llega al banco y quiere entrar.
La empresa 13 ingresa a la mesa de entrada.
La empresa 13 ingresa a la fila de atencion de empresas.
Empresa 11 llega al banco y quiere entrar.
La empresa 11 ingresa a la mesa de entrada.
La empresa 11 ingresa a la fila de atencion de empresas.
El empleado termino de atender una empresa. (id_empleado: 1)
Un empleado esta atendiendo una empresa. (id_empleado: 1)
La empresa 12 fue atendido y se retira.
Usuario 6 llega al banco y quiere entrar.
El usuario 6 ingresa a la mesa de entrada.
```

2. MINISHELL

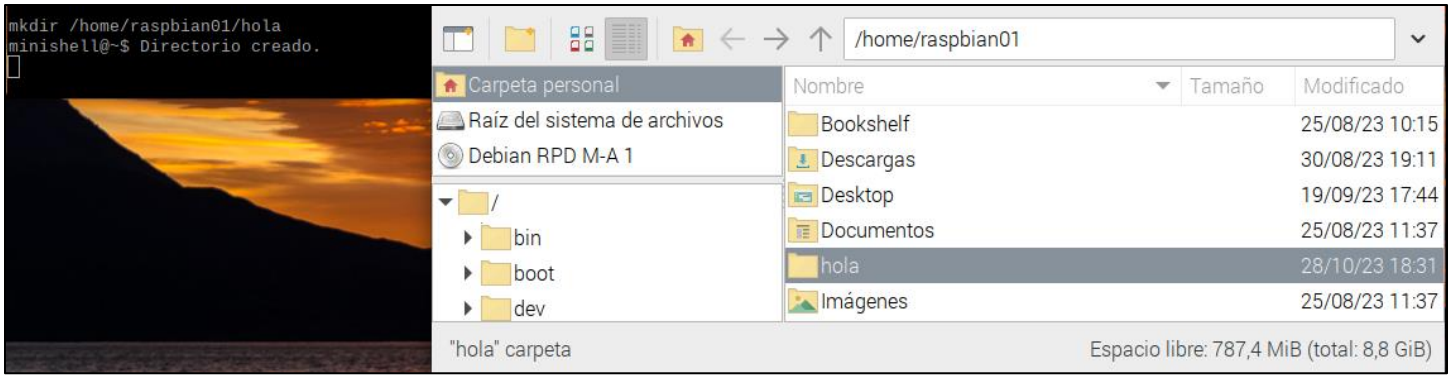
Para este problema usamos un proceso para cada comando y cada comando lo implementamos en un archivo separado.

A continuación, mostramos los comandos funcionando:

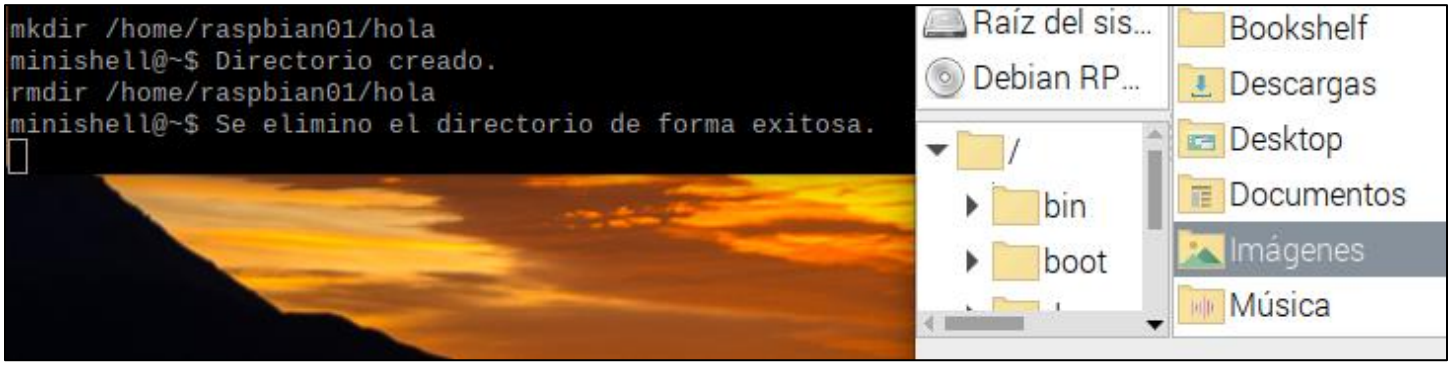
a) help: ayuda con los comandos.

```
-MINISHELL-
En caso de necesitar ayuda, por favor, ingrese el comando 'help'.
Si desea salir de la MiniShell ingrese el comando 'quit'.
Desarrollado por: Sabrina Barrionuevo y Nahuel Diaz.


minishell@~$ help
minishell@~$ Comando 'quit': Finaliza la ejecucion de la MiniShell.

Comando 'help': Muestra una pequeña ayuda con respecto a los comandos válidos para la MiniShell.

Comando 'mkdir [nombreDirectorio]': crea un nuevo directorio.
Comando 'ldir [nombreDirectorio]': lista el contenido de un directorio.
Comando 'rmdir [nombreDirectorio]': elimina el directorio seleccionado y todo su contenido.
Comando 'mkfile [nombreArchivo]': crea un nuevo archivo.
Comando 'lfile [nombreArchivo]': muestra el contenido de un archivo.
Comando 'chmod [nombreArchivo] [permiso]': cambia los permisos otorgados al propietario de un archivo.

Los permisos disponibles son:
  # 400: Lectura
  # 200: Escritura
  # 100: Ejecución
  # 600: Lectura y escritura
  # 500: Lectura y ejecución
  # 300: Escritura y ejecución
  # 700: Lectura, escritura y ejecución
```

b) mkdir: crear un directorio nuevo.

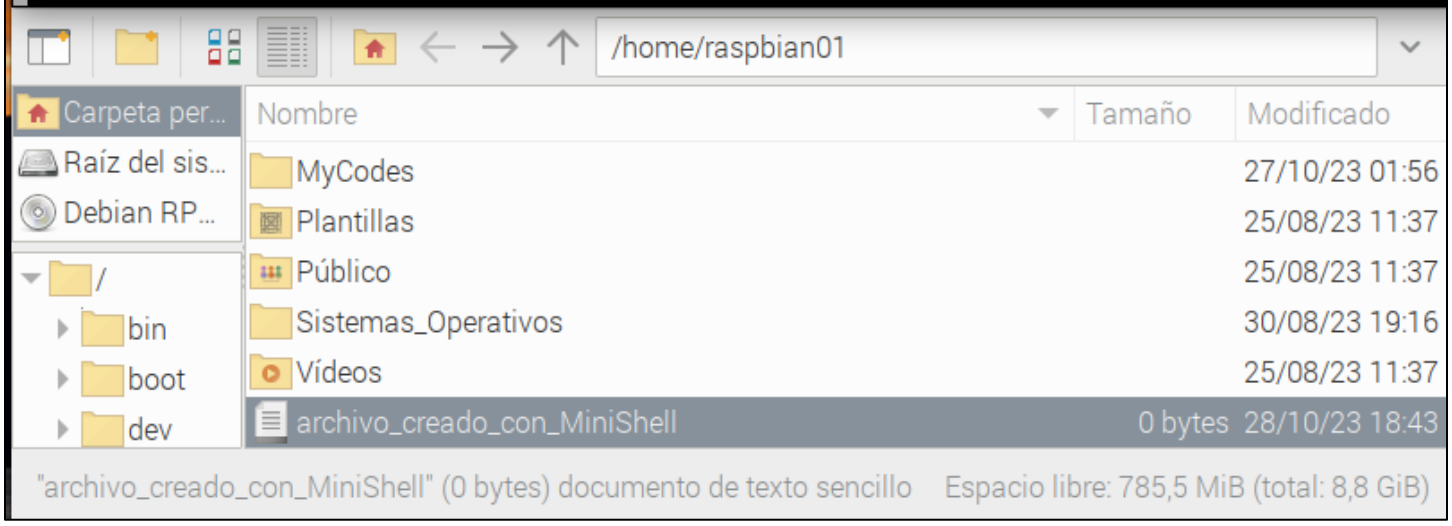


c) rmdir: remover directorio.



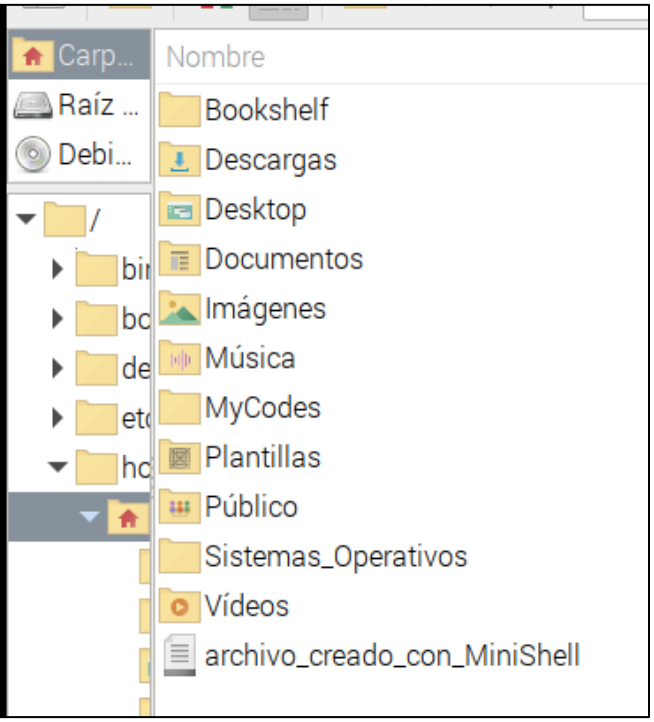
d) mkfile: crear un archivo.

```
mkfile /home/raspbian01/archivo_creado_con_MiniShell
minishell@~$ El archivo se ha creado con exito
```



e) ldir: lista el contenido de un directorio.

```
ldir /home/raspbian01
minishell@~$ .xsession-errors
Documentos
.profile
Desktop
.bash_history
Descargas
Vídeos
.local
.config
Público
Plantillas
.xsession-errors.old
.cache
.bashrc
MyCodes
.bash_logout
Imágenes
Música
.pki
Bookshelf
.Xauthority
Sistemas_Operativos
archivo_creado_con_MiniShell
```



f) lfile: mostrar el contenido del archivo.

```
lfile /home/raspbian01/archivo_creado_con_MiniShell
minishell@~$ Escribiremos este mensaje dentro del archivo creado anteriormente con la MiniShell
Ahora, usamos el comando lfile para mostrar en la MiniShell el contenido del archivo
```

g) chmod

```
chmod /home/raspbian01/archivo_creado_con_MiniShell 700
minishell@~$ Permiso actualizado correctamente.
```

1.2 Sincronización.

1. SECUENCIA.

a) Para este inciso usamos la cantidad mínima de hilos y semáforos necesaria: tres semáforos y tres hilos. No hubo mucha dificultad para realizarlos ya que habíamos visto estos tipos de secuencia en las clases.

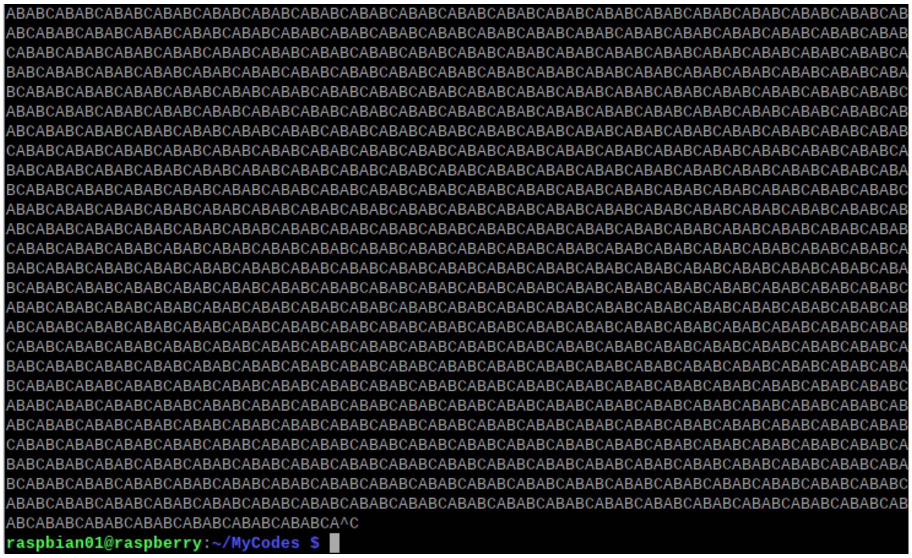
Resultados para la secuencia: ABABCABABCABABC

Resultados para la secuencia: ABABCABCDABABCABCD

Para esta secuencia necesitamos un semáforo y un hilo adicional.

b) Para este inciso usamos tres pipes y tres procesos hijos para cada función.

Resultados para la secuencia: ABABCABABCABABC



Resultados para la secuencia: ABABCABCDABABCABCD

De nuevo, para esta secuencia necesitamos un pipe y un proceso hijo adicional.

2. RESERVA DE AULAS.

I)

▶ Para resolver el problema utilizaremos un recurso compartido que serán las horas a reservar. Estas horas se representarán como un arreglo de "booleanos" de longitud acorde a la cantidad de horas disponibles.

▶ Los “alumnos” se ejecutan de manera concurrente realizando las operaciones aleatoriamente utilizando las probabilidades dadas. El alumno almacenara la hora reservada (en caso de que tuviera una).

▶ Para las reservas se elegirá la hora de manera aleatoria y se intentará registrarlo en la tabla de horas y se guardará su posición.

En caso de que no se pueda realizar la reserva el alumno el alumno tendrá que volver a elegir una operación.

- Para la cancelación se utilizará el horario almacenado.
- Para la consulta también se elijará el horario de manera aleatoria.

▶ Para registrar y cancelar reservas se garantizará la exclusión mutua al momento de modificar la tabla de reservas.

```
El alumno 6 reservo la computadora a las 17 horas.
El alumno 3 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 4 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 10 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 8 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 0 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 9 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 12 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 13 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 15 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 16 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 19 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 20 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 21 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 22 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 23 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 11 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 14 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 7 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 5 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 2 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 1 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 18 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 17 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 24 intenta reservar a las 17 pero ya se encuentra reservada.
el alumno 6 ya tiene una hora reservada.
El alumno 3 reservo la computadora a las 13 horas.
El alumno 4 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 10 reservo la computadora a las 15 horas.
El alumno 8 reservo la computadora a las 9 horas.
El alumno 0 consultó si la computadora esta reservada a las 16 horas y resultó que no lo estaba.
El alumno 9 consultó si la computadora esta reservada a las 19 horas y resultó que no lo estaba.
El alumno 12 reservo la computadora a las 10 horas.
El alumno 13 reservo la computadora a las 14 horas.
El alumno 15 consultó si la computadora esta reservada a las 13 horas y resultó que lo estaba.
El alumno 16 intenta reservar a las 13 pero ya se encuentra reservada.
El alumno 19 intenta reservar a las 15 pero ya se encuentra reservada.
El alumno 20 reservo la computadora a las 12 horas.
El alumno 21 consultó si la computadora esta reservada a las 18 horas y resultó que no lo estaba.
El alumno 22 intenta reservar a las 10 pero ya se encuentra reservada.
El alumno 23 intenta reservar a las 15 pero ya se encuentra reservada.
El alumno 11 no puede cancelar una reserva que no tiene.
El alumno 14 reservo la computadora a las 19 horas.
El alumno 7 intenta reservar a las 14 pero ya se encuentra reservada.
El alumno 5 consultó si la computadora esta reservada a las 10 horas y resultó que lo estaba.
El alumno 2 reservo la computadora a las 16 horas.
El alumno 1 no puede cancelar una reserva que no tiene.
El alumno 18 intenta reservar a las 15 pero ya se encuentra reservada.
El alumno 17 no puede cancelar una reserva que no tiene.
El alumno 24 intenta reservar a las 16 pero ya se encuentra reservada.
^C
raspbian01@raspberry:~/MyCodes $
```

II)

La solución retorna mensajes con los eventos ocurridos durante la ejecución del programa. Los eventos que podrían ocurrir son los siguientes:

▶ El alumno reserva un turno.

- Ya contaba con un turno previo.
- La hora que intenta reservar ya fue reservada.
- Se realiza la reserva correctamente.

▶ El alumno cancela la reserva.

- Intenta cancelar una reserva, pero no tiene ninguna.
- Se cancela la reserva correctamente.

▶ El alumno consulta el estado de un horario.

- El horario se encuentra reservado.
- El horario no se encuentra reservado.

Se consideró la posible inanición de los procesos que quieren realizar reservas/cancelaciones. En esta solución se dio prioridad a las consultas frente a las reservas/cancelaciones debido a que las consultas ocurren con menor frecuencia reduciendo la probabilidad de inanición y simplificando la implementación de la solución.

III) Para esta solución, decidimos usar semáforos para garantizar a exclusión mutua.

Los resultados son los siguientes:

```
raspbian01@raspberry:~/MyCodes $ gcc -o compilado aulaConProcesos.c -lpthread
raspbian01@raspberry:~/MyCodes $ ./compilado
El alumno 0 reservo la computadora a las 15 horas.
El alumno 3 reservo la computadora a las 11 horas.
El alumno 4 reservo la computadora a las 14 horas.
El alumno 8 consultó si la computadora esta reservada a las 11 horas y resultó que lo estaba.
El alumno 6 reservo la computadora a las 9 horas.
El alumno 11 reservo la computadora a las 17 horas.
El alumno 13 reservo la computadora a las 13 horas.
El alumno 14 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 16 consultó si la computadora esta reservada a las 9 horas y resultó que lo estaba.
El alumno 18 consultó si la computadora esta reservada a las 16 horas y resultó que no lo estaba.
El alumno 20 intenta reservar a las 13 pero ya se encuentra reservada.
El alumno 21 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 23 no puede cancelar una reserva que no tiene.
El alumno 10 no puede cancelar una reserva que no tiene.
El alumno 15 consultó si la computadora esta reservada a las 18 horas y resultó que no lo estaba.
El alumno 12 no puede cancelar una reserva que no tiene.
El alumno 17 reservo la computadora a las 19 horas.
El alumno 9 intenta reservar a las 14 pero ya se encuentra reservada.
El alumno 19 consultó si la computadora esta reservada a las 11 horas y resultó que lo estaba.
El alumno 7 intenta reservar a las 13 pero ya se encuentra reservada.
El alumno 22 no puede cancelar una reserva que no tiene.
El alumno 5 no puede cancelar una reserva que no tiene.
El alumno 24 consultó si la computadora esta reservada a las 16 horas y resultó que no lo estaba.
El alumno 2 no puede cancelar una reserva que no tiene.
El alumno 1 intenta reservar a las 17 pero ya se encuentra reservada.
El alumno 0 cancelo la reserva de las 15 horas.
El alumno 3 cancelo la reserva de las 11 horas.
El alumno 4 cancelo la reserva de las 14 horas.
El alumno 8 no puede cancelar una reserva que no tiene.
El alumno 6 consultó si la computadora esta reservada a las 20 horas y resultó que no lo estaba.
el alumno 11 ya tiene una hora reservada.
El alumno 13 cancelo la reserva de las 13 horas.
El alumno 14 consultó si la computadora esta reservada a las 19 horas y resultó que lo estaba.
El alumno 16 reservo la computadora a las 16 horas.
El alumno 18 consultó si la computadora esta reservada a las 14 horas y resultó que no lo estaba.
El alumno 20 no puede cancelar una reserva que no tiene.
El alumno 21 no puede cancelar una reserva que no tiene.
El alumno 23 reservo la computadora a las 15 horas.
El alumno 10 reservo la computadora a las 11 horas.
El alumno 15 consultó si la computadora esta reservada a las 18 horas y resultó que no lo estaba.
el alumno 17 ya tiene una hora reservada.
El alumno 12 no puede cancelar una reserva que no tiene.
El alumno 9 consultó si la computadora esta reservada a las 20 horas y resultó que no lo estaba.
El alumno 19 intenta reservar a las 11 pero ya se encuentra reservada.
El alumno 7 consultó si la computadora esta reservada a las 19 horas y resultó que lo estaba.
El alumno 22 no puede cancelar una reserva que no tiene.
El alumno 5 reservo la computadora a las 10 horas.
El alumno 24 reservo la computadora a las 13 horas.
El alumno 2 intenta reservar a las 16 pero ya se encuentra reservada.
El alumno 1 reservo la computadora a las 18 horas.
^C
raspbian01@raspberry:~/MyCodes $
```

2. Problemas

1. LECTURA.

El Flyer realizado se encuentra en: Problemas/Lectura/VxWorks Folleto.pdf

2. PROBLEMAS CONCEPTUALES.

La resolución de los ejercicios se encuentra en: Problemas/Problemas Conceptuales/Problemas Conceptuales.pdf