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:

▶ Para tratar el problema de la acumulación de clientes cada vez que estos tienen que esperar, usamos 4 buffers (arreglos de caracteres) para simular las respectivas filas de espera (fila de la mesa de entrada, fila de políticos, fila de usuarios, y fila de empresas).

El planteo es: Un cliente llega al banco y debe comprobar si hay lugar en la mesa de entrada antes de ingresar, sino lo hay, debe retirarse. Una vez en la mesa de entrada, cada cliente es despachado (en orden de llegada) a una fila correspondiente con su tipo (usuario, empresa o político). En nuestro diseño, si un cliente intenta entrar a la fila de su tipo y dicha fila está llena, el cliente vuelve a la mesa de entrada, pero al último lugar de la fila. Si el cliente puede entrar, se coloca al final de la fila y espera hasta ser atendido. Mientras se atiende a un cliente, este sigue ocupando su lugar en la fila, es cuando la atención al cliente termina que este ultimo se va de la fila y se retira del banco.

```
raspbian01@raspberry:~/MyCodes $ ./compilado
Llego un cliente de tipo P al banco.
Ingresa un cliente de tipo P a la mesa de entrada.
Clientes en Mesa de Entrada: 1.
Llego un cliente de tipo E al banco.
Ingresa un cliente de tipo E a la mesa de entrada.
Clientes en Mesa de Entrada: 2.
Llego un cliente de tipo U al banco.
Ingresa un cliente de tipo U a la mesa de entrada.
Clientes en Mesa de Entrada: 3.
Llego un cliente de tipo P al banco.
Ingresa un cliente de tipo P a la mesa de entrada.
Clientes en Mesa de Entrada: 4.
Ingresa un cliente a la fila de politicos.
Clientes en la fila de Politicos: 1.
Llego un cliente de tipo E al banco.
Ingresa un cliente de tipo E a la mesa de entrada.
Clientes en Mesa de Entrada: 4.
Ingresa un cliente a la fila de empresas.
Clientes en la fila de Empresas: 1.
Llego un cliente de tipo U al banco.
Ingresa un cliente de tipo U a la mesa de entrada.
Clientes en Mesa de Entrada: 4.
Ingresa un cliente a la fila de usuarios.
Clientes en la fila de Usuarios: 1.
Llego un cliente de tipo U al banco.
Ingresa un cliente de tipo U a la mesa de entrada.
Clientes en Mesa de Entrada: 4.
Ingresa un cliente a la fila de politicos.
Clientes en la fila de Politicos: 2.
Llego un cliente de tipo P al banco.
Ingresa un cliente de tipo P a la mesa de entrada.
Clientes en Mesa de Entrada: 4.
Llego un cliente de tipo U al banco.
Ingresa un cliente de tipo U a la mesa de entrada.
Clientes en Mesa de Entrada: 5.
Ingresa un cliente a la fila de empresas.
Clientes en la fila de Empresas: 2.
Llego un cliente de tipo E al banco.
Ingresa un cliente de tipo E a la mesa de entrada.
Clientes en Mesa de Entrada: 5.
Ingresa un cliente a la fila de usuarios.
Clientes en la fila de Usuarios: 2.
Llego un cliente de tipo U al banco.
Ingresa un cliente de tipo U a la mesa de entrada.
Clientes en Mesa de Entrada: 5.
Ingresa un cliente a la fila de usuarios.
Clientes en la fila de Usuarios: 3.
Llego un cliente de tipo P al banco.
Ingresa un cliente de tipo P a la mesa de entrada.
Clientes en Mesa de Entrada: 5.
Ingresa un cliente a la fila de politicos.
Clientes en la fila de Politicos: 3.
Llego un cliente de tipo P al banco.
Ingresa un cliente de tipo P a la mesa de entrada.
Clientes en Mesa de Entrada: 5.
Ingresa un cliente a la fila de usuarios.
Clientes en la fila de Usuarios: 4.
Llego un cliente de tipo P al banco.
```

Acordamos usar un solo hilo para las funciones mesa_de_entrada y filas_especificas. Y tres hilos (3 empleados) para la función de empleados.

▶ Con 4 semáforos contadores controlamos las capacidades máximas de cada fila. Estos semáforos están inicializados en la máxima capacidad de la mesa de entrada o de las filas específicas, según corresponda. Cada vez que un cliente entra a la fila, el semáforo disminuye, impidiendo el paso cuando decrementa hasta cero. Siendo la salida de un cliente de la fila lo único que puede habilitar el semáforo nuevamente.

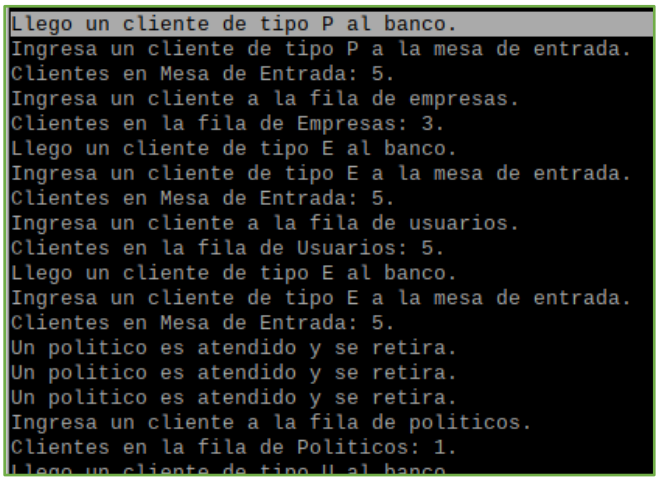
Además, para verificar este control, usamos 4 variables contadoras para que durante la ejecución podamos llevar un registro de la cantidad de personas esperando en las colas. Por supuesto, estas variables aumentan o decrementan con el ingreso o salida de un cliente de la fila, respectivamente.

▶ El objetivo es insertar y extraer los clientes de los buffers en orden FIFO. Para lograr eso usamos 4 índices numéricos para insertar, los cuales están ligados a la última posición de la fila (donde van los últimos clientes que llegan). Y, por otro lado, usamos otros 4 índices para extraer, los cuales están ligados a la primera posición de la fila (donde están los clientes que llegaron primero).

▶ Utilizamos un semáforo binario llamado “mutex” para controlar y proteger el acceso a las secciones del código donde se manipulan los

datos de los buffers. La idea es garantizar que dos hilos no estén modificando ninguno de los buffers a la vez.

▶ Tomamos la decisión de retrasar el comienzo de atención a los clientes para permitir que las filas se “carguen” con algunos clientes antes de comenzar a atender periódicamente. En la primera imagen, podemos ver como se realizan varios ingresos de clientes sin atender a ninguno.



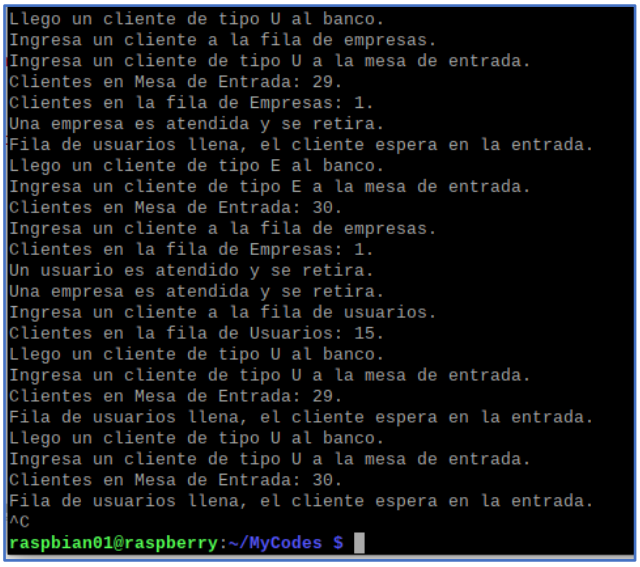
En la siguiente imagen, que es la continuación de la anterior, vemos la primera atención de los tres empleados, al momento de atender hay 3 políticos, 3 empresas y 5 usuarios esperando en las filas de atención. Notar que los tres políticos son atendidos primero a pesar de que el segundo político llego 4to al banco y el tercero llego 8vo.

▶ Hay tres empleados, identificados como 0, 1 y 2, respectivamente. Si hay algún político en la fila debe ser atendido antes que los otros, por lo que, lo primero que hace cualquiera de los empleados es checkear si hay algún político esperando ser atendido. Si no hay ninguno, cada empleado atenderá un cliente de la fila que le fue asignada, en este caso, el empleado 0 y 1 atienden la fila de empresas, y el empleado 2 atiende la fila de usuarios.

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

▶ Creamos 4 colas de mensajes para cada una de las filas de espera, siendo cada mensaje un cliente esperando. Y con las operaciones *receive* y *send* simulamos el ingreso y salida de los clientes de las filas.

Al principio se analizó la idea de combinar las tres colas de clientes específicos en una sola, usando el “tipo” de los mensajes para identificar a cada cliente, y usar 3 variables contadoras para llevar un control de las capacidades máximas. Pero un ayudante nos indicó, que esto no era necesario. Por lo tanto, decidimos implementarlo de esta forma, ya que nos facilita llevar un mejor control de cada uno de los tipos de clientes en espera.



▶ Para el control de capacidades máximas de las filas, usamos la función msgctl() para guardar la información de una cola en una variable que llamamos “info”, y al realizar la operación *info.msg_qnum* nos devolverá la cantidad de clientes esperando en la fila, de esta forma verificamos que no acceda ningún cliente a una fila llena.

En la imagen, mostramos que ocurre cuando en un algún punto de la ejecución una fila alcanza su capacidad máxima.

▶ Los empleados como en la implementación anterior, intentan recibir un mensaje de la cola de políticos antes que de cualquier otra.

Usamos la etiqueta IPC_NOWAIT para que los empleados no se bloquearan esperando que un cliente llegue a la fila.

▶ Un inconveniente por el que atravesamos durante la implementación fue al descubrir que la operación msgsnd() enviaba el mensaje a todas las colas que habíamos creado, lo que provocaba grandes inconsistencias en la ejecución. La solución a este problema fue la etiqueta IPC_PRIVATE, la cual usamos para crear las colas siendo la key de la cola dicha etiqueta.

c) La implementación del banco A tiene varias ventajas sobre la implementación de banco B, esto porque, por ejemplo, el banco hecho con colas de mensajes no hace un uso eficiente de los recursos debido al número de colas que se usan. Además, la sincronización es mejor en el banco A, ya que gracias a los semáforos podemos brindar exclusión mutua para que no haya dos hilos ejecutando a la vez en la sección critica. Una desventaja del banco A seria la cantidad de índices y contadores que se necesitan para utilizar y llevar un seguimiento del estado de un buffer.

2. MINISHELL

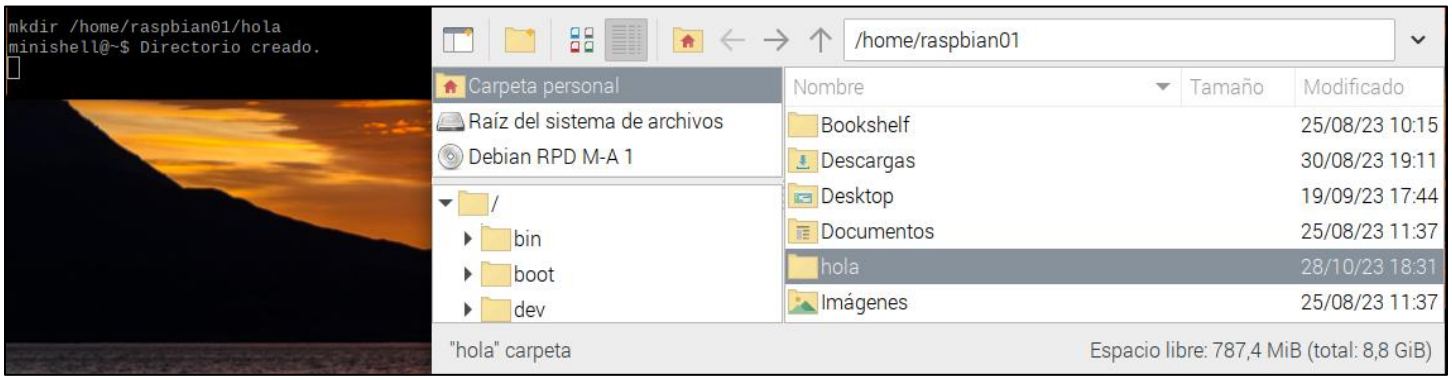
Para este problema usamos un proceso para cada comando y cada comando lo implementamos en un archivo separado.

A continuación, mostramos los comandos funcionando:

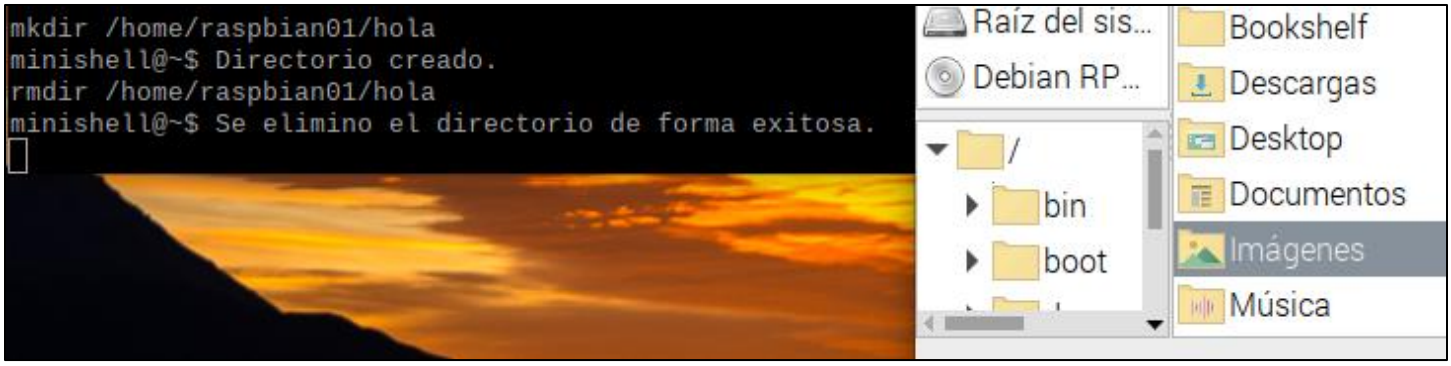
a) help: ayuda con los comandos.

```
-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
```

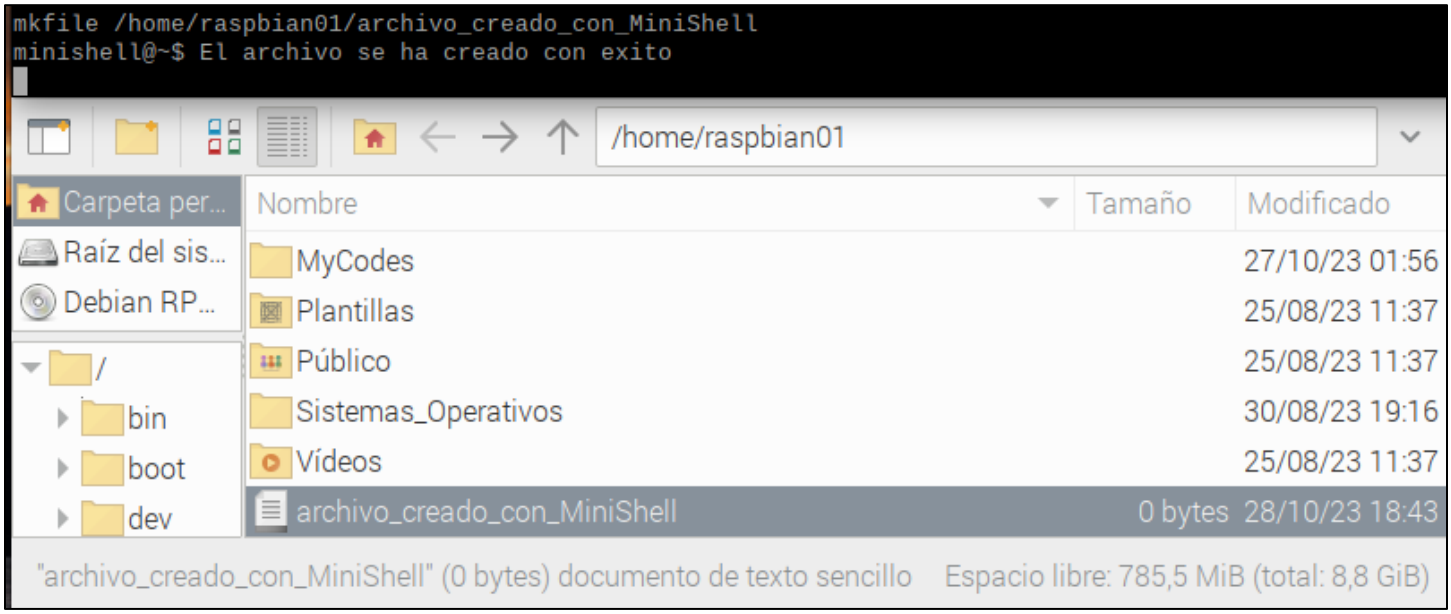
b) mkdir: crear un directorio nuevo.



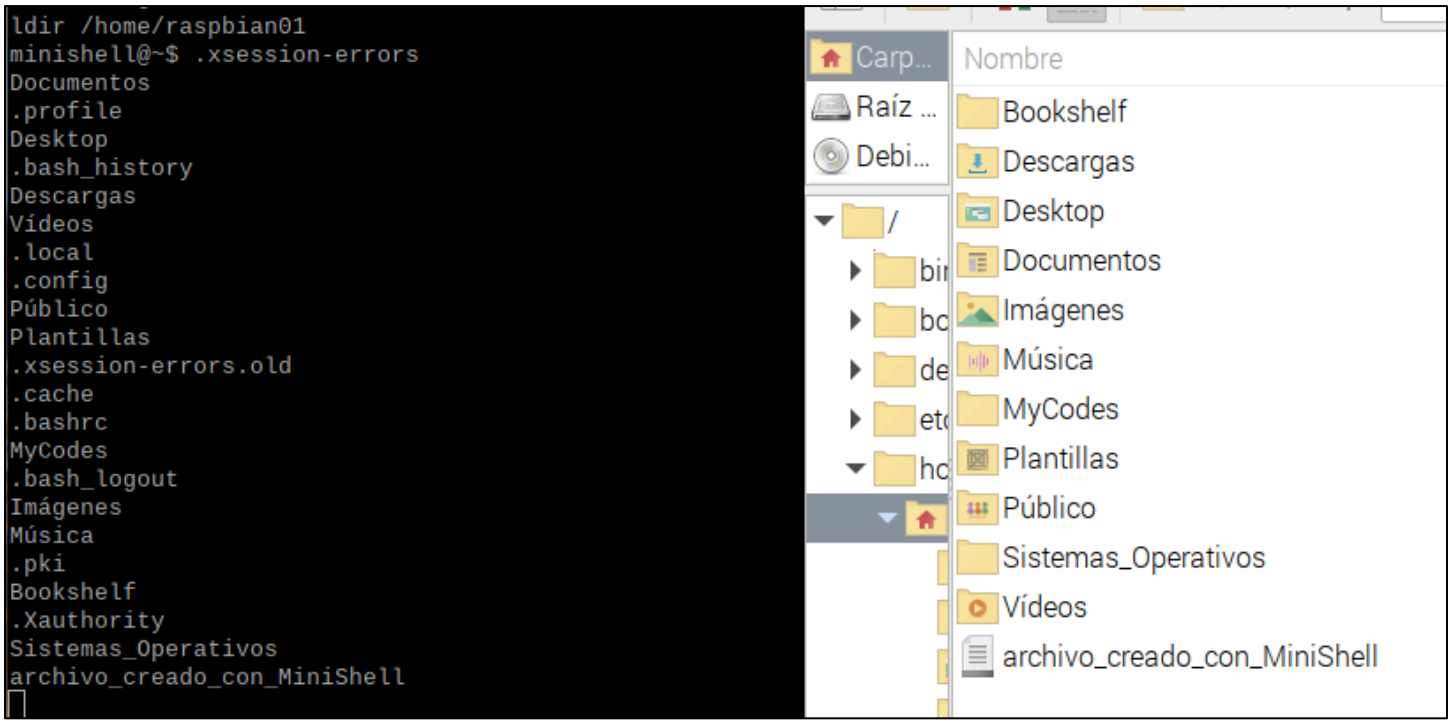
c) rmdir: remover directorio.



d) mkfile: crear un archivo.



e) ldir: lista el contenido de un directorio.



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

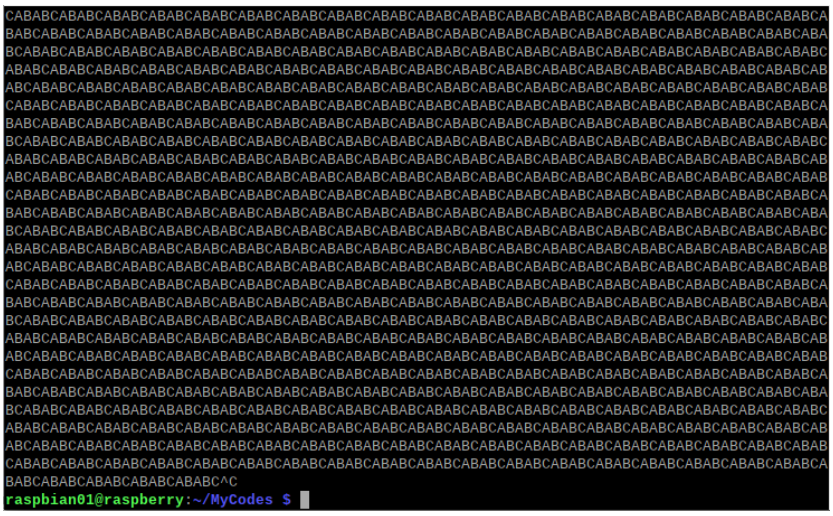
```
chmod /home/raspbian01/archivo_creado_con_MiniShell 700
minishell@~$ Permiso actualizado correctamente.
```

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

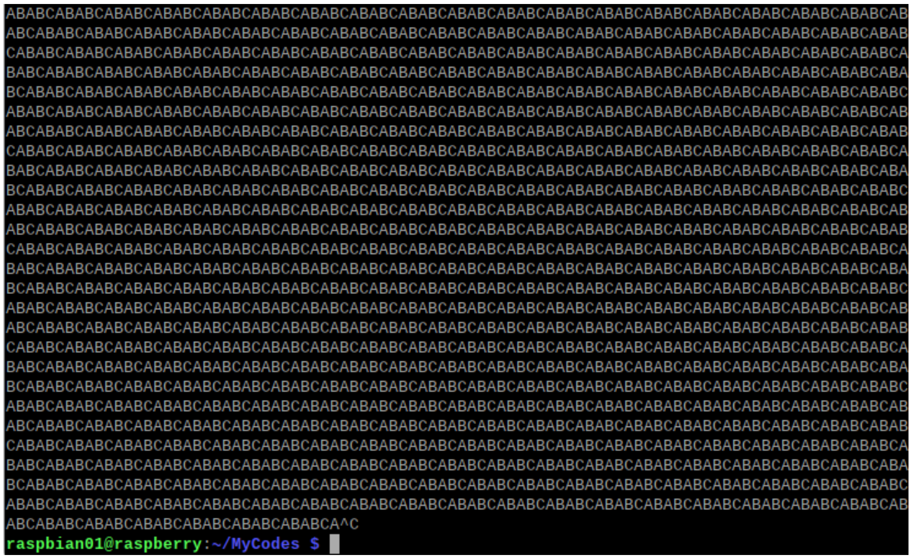


Resultados para la secuencia: ABABCABCDABABCABCD

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



```
ABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCAB
ABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABAB
CABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCA
BABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABA
BCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABC
ABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCAB
ABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABAB
CABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCA
BABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABA
BCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABC
ABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCAB
ABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABAB
CABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCA
BABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABA
BCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABC
ABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCAB
ABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABAB
CABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCA
BABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABA
BCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABC
ABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCAB
ABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABAB
CABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCA
BABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABA
BCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABC
ABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCABABCAB
ABCABABCABABCABABCABABCABABCABABCA^C
raspbian01@raspberry:~/MyCodes $
```

Resultados para la secuencia: ABABCABCDABABCABCD

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

```
BCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABC
DABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDA
BABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABA
BCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABC
ABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCAB
CDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCD
ABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDAB
ABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABAB
CABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCA
BCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABC
DABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDA
BABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABA
BCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABC
ABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCAB
CDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCD
ABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDAB
ABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABAB
CABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCA
BCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABC
DABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDA
BABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABA
BCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABC
ABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCAB
CDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCD
ABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDAB
ABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABAB
CABCDABABCABCDABABCABCDABABCABCDABABCABCDABABCABCDABAB^C
raspbian01@raspberry:~/MyCodes $
```

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