PRÁCTICO 3

Ejercicio 1

Objetivo: Estudiar los planificadores de I/O

El kernel de linux instalado en la máquina virtual dispone de tres planificadores de I/O diferentes: cfq, deadline y noop.
Se pide: Analizar el comportamiento de cada uno de estos planificadores probando diferentes operaciones de disco. En particular interesa probar el comportamiento ante escrituras y lecturas contiguas y ante lecturas y escrituras aleatorias.
Notas:

- Para crear un archivo de 1GB con ceros en bloques de 4KB se puede ejecutar

  ```
  #dd if=/dev/zero bs=4096 count=262144 of=file_1GB
  ```

- Para leer este archivo completo se puede ejecutar:

  ```
  #dd if=file_1GB of=/dev/null bs=4096
  ```

- Para leer un sector particular del archivo se puede ejecutar:

  ```
  #dd if=file_1GB of=/dev/null skip=<0..262143> count=1 bs=4096
  ```

Ejercicio 2 (Obligatorio)

Objetivo: Usar IPC para comunicar procesos

En esta tarea se desea implementar un sistema de foro de mensajes. Un sistema de este tipo es uno que permite a diferentes usuarios la escritura y lectura de múltiples mensajes en formato texto. En cierta forma, este sistema es similar a un grupo de noticias.
Tenemos el concepto de servidor de foro y de clientes de foros. El servidor de foro será la entidad que permitirá hospedar un foro. Un cliente del foro se conectará al servidor de foro y permitirá la lectura y/o escritura de mensajes al foro que se encuentra hospedado en dicho servidor.
La mecánica de funcionamiento será la siguiente:

- Cada `ServiForo` hospedará la información de un foro y los mensajes en dicho foro. Se deben tomar en cuenta las siguientes consideraciones:
  - Tenemos un solo servidor de foro.
  - Una vez que el servidor está levantado se podrá salir utilizando el comando exit en la consola del `ServiForo`.
  - Una vez que se termina el servidor, no se permitirá nuevas operaciones sobre los foros.
- Cuando un cliente `CliForo` entra al sistema, contará con las siguientes operaciones disponibles:

list este comando despliega los mensajes existentes. Al desplegar la lista de mensajes, se mostrará el número de mensaje, así como el autor del mismo.

write mensaje permite ingresar un nuevo mensaje. Los mensajes se numeran automáticamente una vez que se escriben en el foro. De esta forma, tenemos el mensaje 1, 2, 3, etc. Mientras que el comando write no se ha completado los demas `CliForo` no podrán ejecutar comandos read en el foro. Al guardar un mensaje, se deberá almacenar además del texto del mismo, quien escribió dicho mensaje.

read numero_mensaje despliega en la consola del cliente el mensaje cuyo número se seleccionó. Al desplegar el mensaje, se debe mostrar el texto del mensaje, así como el autor del mismo

exit termina el cliente y sale del sistema.

- Al ejecutar el `CliForo`, se indicará el usuario con que se entra al foro. La ejecución será con un comando de la forma: `CliForo nombre_de_usuario`

En la siguiente tabla se presenta un ejemplo para aclarar el funcionamiento del sistema:

ServiForo			
	CliForo pepe	CliForo cacho	ServiForo
			<ya existe un servidor de foros levantado>
	Exit		
	<Chau>		
		write Hola todos los que me conocen	
<Se ha creado el mensaje numero 1 en el foro>			
	CliForo cacho		
	<El usuario cacho ya se encuentra logueado>	list	
	CliForo pepe	<Mensajes disponibles 1 – cacho>	
	list		CliForo juan
	<Mensajes disponibles 1 – cacho>		
	write Hola todos los que no me conocen		
<Se ha creado el mensaje numero 2 en el foro>		list	
		<Mensajes disponibles 1 – cacho 2 – pepe>	
	read 1	read 1	
exit	cacho: Hola todos los que no me conocen	cacho: Hola todos los que no me conocen	
<El servicio esta agendado para finalizar en cuanto todos los CliForo salgan>	read 1		
	<No se permiten nuevas operaciones con el foro>	exit	
	exit	<Chau>	
	<Chau>		exit
			<Chau>
<Servidor finalizado>			

Notas:

- Para iniciar un `CliForo` se deberá pasar como argumento al ejecutable el nombre del usuario que quiere acceder al sistema.
- Los `CliForo` deberán esperar un tiempo aleatorio entre 1 y 5 segundos luego de ejecutar un comando para que el comando se considere completado.
- `CliForo` y `ServiForo` deben recibir los comandos a través de la entrada estándar.
- La cantidad máxima de mensajes que se podrán enviar al foro estará acotada por la constante MAX_MENSAJES_EN_FORO.
- La cantidad máxima de clientes (`CliForo`) en el sistema estará acotada por la constante MAX_CLIENTES.
- La estructura de datos que alberga la información de los foros y los mensajes debe ser accesible solamente para el `ServiForo`.
- El largo máximo de un mensaje esta acotado por la constante MAX_LARGO_MENSAJE. Se asume que los clientes no van a escribir un mensaje de tamaño mayor.
- Si se intentan superar los topes definidos en las constantes MAX_MENSAJES_EN_FORO y MAX_CLIENTES se deberá mostrar un mensaje de error pertinente.
- Cualquier situación de error no considerada en los comandos antes presentados, deberá manejarse desplegando un mensaje de error apropiado.
- No serán correctas soluciones que utilicen uno o más semáforos por mensaje o situaciones similares.
- Compilar utilizando make con un archivo Makefile (sin extensión). No se aceptarán entregas sin Makefiles o que compilan con scripts o similares.
- Deberá compilarse con la opción –Wall (un parámetro de gcc). Esta opción permite ver errores y advertencias (warnings) de compilación más detalladas. El trabajo entregado NO debe producir warnings al compilarlo.
- Se debe implementar usando memoria compartida y semáforos. No está permitido el uso de colas de mensajes.

Ejercicio 3 (Obligatorio)

Objetivo: Usar dispositivos de caracteres

Se desea crear un dispositivo de caracteres (*tsofifo*) para que funcione como un *pipe FIFO*. Un proceso escribirá sobre el pipe y otro podrá leer la información.
El driver deberá trabajar con 4 colas FIFO de mensajes a través de 8 dispositivos (4 para escribir y 4 para leer). Los números pares serán para escritura y los impares de lecturas. Por ejemplo, para la cola 0 se tienen los dispositivos *tsofifo0* para escritura y *tsofifo1* para lectura.
El *major number* puede ser pasado como parámetro al insertar el módulo o, en otro caso, será asignado en forma dinámica.
Consideraciones:

- Cada cola dispondrá de un buffer de tamaño máximo de 4096 bytes.

- Solo puede haber como máximo un escritor y un lector ejecutando concurrentemente para cada cola. Si un proceso desea abrir una cola que ya está siendo usada, se debe retornar -1.
- Si un lector quiere leer de una cola en donde no hay ningún escritor que haya realizado la apertura se debe retornar 0 en la operación de lectura.
- Si un lector está bloqueado por una lectura y el escritor realiza un *close*, se debe retornar 0.
- Si un escritor realiza un *close* sobre la cola, el lector correspondiente podrá consumir todo el buffer restante, o realizar el *close*. Si sucede el primer caso, se cumplirá el punto anterior por lo que la última lectura retornará 0. Si sucede lo segundo, se deberá liberar la información que reste en el buffer para que en un posterior uso del pipe no quede información de la comunicación anterior.
- El escritor deberá bloquearse si se llena el buffer asignado a la cola. Será desbloqueado cuando un lector consuma parte del buffer.
- Un lector deberá bloquearse si no hay nada en el buffer. Un lector será desbloqueado una vez que el escritor escriba en el buffer.

Notas:

- Las operaciones deben ser reentrantes y se debe generar la mayor concurrencia posible.
- Para la sincronización se pueden utilizar cualquiera de los mecanismos provistos por el kernel de Linux.

Se pide:

- Diseñar, implementar y documentar el módulo del núcleo que realice la funcionalidad mencionada, asegurando el correcto funcionamiento del mismo.