CC4302 Sistemas Operativos

Tarea 7 – Semestre Primavera 2024 – Profs.: Mateu, Torrealba, Arenas

En esta tarea, Ud. deberá programar un driver para Linux que permita coordinar a los 5 filósofos en una mesa redonda, es decir, el problema clásico de sincronización de "la cena de filósofos". Cada filósofo desea alternar entre pensar y comer, y debe asegurarse de que sus vecinos no estén comiendo antes de poder hacerlo. Cada filósofo puede manifestar su intención de comer mediante el comando */bin/echo* al dispositivo */dev/cena-filosofos*. El dispositivo */dev/ cena-filosofos* debe tener número major 61.

Un proceso ofrece la intención de "comer" escribiendo la línea *echo comer N > /dev/cena-filosofos*, donde *N* es el número del filósofo. Este comando no terminará hasta que el filósofo pueda efectivamente comenzar a comer, es decir, cuando sus vecinos inmediatos (el filósofo a su izquierda y a su derecha) estén pensando y no comiendo. Para notificar que un filósofo terminó de comer y vuelve a pensar, debe utilizarse el comando *echo pensar N > /dev/cena-filosofos*. Si un filósofo espera comer y presiona *Control-C*, también desistirá de su solicitud de comer. Además, para evitar hambruna un filósofo no puede comer si un filósofo adyacente espera y llegó antes.

La lectura con cat devuelve el estado actual de cada filósofo pensando, en espera de comer o comiendo.

El siguiente ejemplo utiliza los comandos estándar de Linux */bin/echo* y *cat* para demostrar el comportamiento esperado de */dev/cena-filosofos*. Su driver debe reproducir exactamente el mismo comportamiento. Si hay aspectos que el ejemplo no aclara, decida Ud. mismo tratando de simplificar su tarea. Las filas de la tabla están ordenadas cronológicamente. Lo que escribió el usuario aparece en negritas. Observe que el prompt $ indica cuando debe terminar un comando. Si el prompt $ no aparece, es porque hay una llamada al sistema pendiente.

<table>
<tr><th>Shell 1</th><th>Shell 2</th><th>Shell 3</th></tr>
<tr><td>$ cat /dev/cena-filosofos(1)
0 pensando
1 pensando
2 pensando
3 pensando
4 pensando
$</td><td></td><td></td></tr>
<tr><td></td><td colspan="2">$ echo “comer 0” > /dev/cena-filosofos(2)
$</td></tr>
<tr><td rowspan="4">$ echo “comer 1” > /dev/cena-filosofos(3)</td><td></td><td></td></tr>
<tr><td>$ cat /dev/cena-filosofos(4)
0 comiendo
1 esperando
2 pensando
3 pensando
4 pensando
$</td><td></td></tr>
<tr><td></td><td rowspan="2">$ echo “comer 2” > /dev/cena-filosofos(5)</td></tr>
<tr><td>$ cat /dev/cena-filosofos(6)
0 comiendo
1 esperando
2 esperando
3 pensando
4 pensando</td></tr>
</table>

	$ **echo "comer 4" > /dev/cena-filosofos** (7) <control-c> (8) $ **cat /dev/cena-filosofos** (9) 0 comiendo 1 esperando 2 esperando 3 pensando 4 pensando *$*	
$	$ **echo "pensar 0" > /dev/cena-filosofos** (10) $	
	$ **echo "pensar 1" > /dev/cena-filosofos** (11) *$*	$ $ **cat /dev/cena-filosofos** (12) 0 pensando 1 pensando 2 comiendo 3 pensando 4 pensando

Notas:

(1) Este comando se ejecuta para verificar el estado inicial de los filósofos. Todos los filósofos están en el estado *pensando*.

(2) El filósofo 0 solicita *comer*. Este comando finaliza, ya que, los filósofos 1 y 4 (sus vecinos) no están *comiendo*.

(3) El filósofo 1 también solicita *comer*. Este comando no finalizará hasta que sus vecinos (filósofos 0 y 2) estén *pensando*.

(4) Se consulta el estado de los filósofos después de las solicitudes de comer de los filósofos 0 y 1. El resultado muestra que el filósofo 0 está *comiendo*, mientras que el filósofo 1 está *esperando*. Los otros filósofos permanecen *pensando*.

(5) El filósofo 2 solicita *comer*. Como necesita el mismo palito que el filósofo 1 quien este se encuentra *esperando*, entonces queda *pensando*.

(6) Se consulta el estado nuevamente. El resultado muestra que el filósofo 0 está *comiendo*, mientras que los filósofos 1 y 2 están *esperando* y los filósofos 3 y 4 permanecen *pensando*.

(7) El filósofo 4 solicita *comer*. Este filósofo queda en *espera* hasta que sus vecinos (filósofos 3 y 0) estén *pensando*.

(8) El filósofo 4 del shell 2 que había solicitado comer, se aburre de esperar y presiona *control-C.* Con lo que *echo* termina de inmediato, el filósofo 4 deja de esperar, no obtiene ningún palito y vuelve a pensar

(9) Se consulta el estado nuevamente. El resultado muestra que el filósofo 0 está *comiendo*, mientras que los filósofos 1 y 2 están *esperando* y los filósofos 3 y 4 permanecen *pensando*.

(10) El filósofo 0 termina de *comer* y vuelve al estado *pensando*. Esto libera a sus vecinos para que puedan

comer si lo desean. Con esto, retorna el shell 1 (filosofo 1 que había solicitado hace mas tiempo el palito)

(11) El filósofo 1 termina de *comer* y vuelve al estado *pensando*. Esto libera a sus vecinos para que puedan *comer* si lo desean. Con esto, retorna el shell 3 (filosofo 2 que había solicitado hace mas tiempo el palito)

(12) Se vuelve a consultar el estado de los filósofos. Ahora el filósofo 2 está *comiendo*, mientras que el resto de los filósofos están pensando *pensando*.

Recursos

Baje de U-cursos el archivo *modules2020-2.tgz*. Contiene enunciados y soluciones de tareas de semestres anteriores con instrucciones para compilarlas y ejecutarlas (ver archivos README.txt en cada directorio). Además se adjunta un tutorial sobre programación de módulos y drivers en Linux.

Baje además el archivo *t7.zip* y descomprímalo. Programe su solución en el archivo filosofos-impl.*c* del directorio *Remate*. Ahí encontrará un *Makefile* para compilar su tarea y las instrucciones para crear el dispositivo */dev/cena-filosofos*. Resuelva su tarea usando los mutex y condiciones incluidos en el directorio *Remate*. Como ejemplo de utilización estudie la solución de *Syncread*. Estos mutex y condiciones son análogos a los de pthreads y están implementados a partir de los semáforos del núcleo de Linux en el archivo *kmutex.c* con encabezados en *kmutex.h*.

Antes de cargar y probar su tarea asegúrese de ejecutar el comando Unix *sync* para garantizar que sus archivos hayan sido grabados en disco y no están pendientes en un caché de Unix. Recuerde que los errores en su driver pueden hacer que Linux se bloquee indefinidamente y tenga que reiniciar el sistema operativo.

Notas importantes

- Para cargar el módulo en el núcleo Ud. debe tener acceso a una máquina con Debian 12 en donde pueda convertirse en el usuario *root*. No es posible hacer esta tarea bajo *Windows Subsystem for Linux*. Instale Debian bajo VirtualBox en su computador siguiendo estas instrucciones.
- Asegúrese de que estén instalados los encabezados del núcleo. En las distribuciones derivadas de Debian como Ubuntu, esto se logra con: *sudo apt-get install linux-headers-$(uname -r)*
- Podría ocurrir que la carga del módulo en el núcleo falle con uno de estos errores:

  insmod: ERROR: could not insert module remate.ko: required key not available

  insmod: ERROR: could not insert module remate.ko: Operation not permitted

  Este problema solo cuando se usa una instalación nativa de Debian para resolver esta tarea. Se debe a que el PC que están usando tiene activado el modo "*secure boot*" a nivel de la BIOS. Tienen que desactivarlo mientras estén haciendo esta tarea. Para desactivarlo tienen que entrar al menú de la BIOS justo después de encender el computador y antes de que parta el sistema operativo. Esto es altamente dependiente del modelo de PC que usan de modo que no los puedo ayudar más. Después de hacer la tarea 7 vuelvan a activarlo para mejorar la seguridad del computador.

Entrega

La tarea se entrega *funcionando* en U-cursos. Para ello entregue solo el archivo cena-filosofos-impl.*c* que implementa el driver pedido. Se descontará medio punto por día de atraso, excepto días sábado, domingo o festivos.