CC4302 Sistemas Operativos

Tarea 3 – Semestre Otoño 2018 – Prof.: Luis Mateu

En esta tarea Ud. deberá programar un driver de Linux que coordina los vigías del Titanic por medio del dispositivo */dev/vigia*. Un vigía escribe su nombre con el comando *\echo* en */dev/vigia* para anunciar su disponibilidad. Actuará de vigía hasta que *\echo* termine. Desde un shell se lee con el comando *cat* el dispositivo */dev/vigia* para desplegar la entrada y salida de vigías. Se requiere que hayan 2 vigías en busca de icebergs para prevenir el hundimiento del Titanic. Estos deben ser los 2 vigías que hayan llegado más recientemente. El dispositivo */dev/vigia* debe tener el número major 61.

El siguiente ejemplo muestra el comportamiento esperado de */dev/vigia*. Su driver debe reproducir exactamente el mismo comportamiento. Si hay aspectos que el ejemplo no aclara, decida Ud. mismo tratando de simplificar su tarea. Su tarea será probada con este mismo ejemplo. 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 (como *open*, *read* o *write*).

Shell 1	***Shell 2***	***Shell 3***	***Shell 4***	***Shell 5***
$ **cat /dev/vigia** (1)				
entra: pedro	$ **\echo pedro > /dev/vigia** (2)			
entra: juan		$ **\echo juan > /dev/vigia** (3)		
entra: diego sale: pedro	$ (5)		$ **\echo diego > /dev/vigia** (4)	
entra: luis sale: juan		$		$ **\echo luis > /dev/vigia** (6)
entra: jorge sale: diego	$ **\echo jorge > /dev/vigia** (6)		$	
entra: alberto sale: luis		$ **\echo alberto > /dev/vigia** (6)		$
entra: ruben sale: jorge	$		$ **\echo ruben > /dev/vigia** (6)	
entra: jose sale: alberto		$		$ **\echo jose > /dev/vigia** (6)
<control-C> (7) $			***<control-C>*** (7) $	***<control-C>*** (7) $

Notas:

(1) Se invoca el comando *cat* en el *shell 1* para leer de */dev/vigia*. Cada vez que *cat* lee con *read* el dispositivo, se invocará el *read* del driver que Ud. programará. Ud. debe hacer que *read* espere hasta que llegue un vigía.

(2) Llega el vigía *pedro*. El comando *\echo* llamará a *write* que invocará el *write* del driver que Ud. programará. Ud. debe hacer que el *read* pendiente en el *shell 1* lea la línea *"entra: pedro"*. No se preocupe por el *newline* porque *write* le entregó *"pedro\n"*, es decir ya viene el *newline*. Cat volverá a invocar *read*, que debe esperar un nuevo vigía. Además este *write* debe esperar hasta que lleguen 2 nuevos vigías. Como no habían vigías previamente, nadie termina su vigilancia.

Es importante que se use el *backslash* en *\echo* y no *echo*. Esto es para asegurarse que se use el comando

/bin/echo y no el *echo* integrado en algunos shell, que podría manejar incorrectamente el *control-C*, haciendo que sea muy difícil terminar el proceso y por ende descargar el driver más tarde.

(3) Similar a (2), llega *juan*. El *write* de su driver debe hacer que el *read* pendiente lea *"entra: juan"*. Nadie termina su vigilancia.

(4) Llega *diego*. Además de hacer lo que se hace en (2) y (3), se debe hacer que *cat* también lea la línea *"sale: pedro"* y el *write* pendiente del shell 2 debe retornar exitosamente.

(5) El comando */bin/echo* del shell 2 termina.

(6) Llega un nuevo vigía. El *write* debe esperar hasta que lleguen 2 nuevos vigías. Debe hacer que el *write* del antepenúltimo vigía termine con lo que su respectivo comando *\echo* también debe terminar.

(7) Se terminan todos los procesos pendientes con *control-C*. Las operaciones *read* o *write* pendientes en su driver deben retornar el código de error *-EINTR*.

Recursos

Baje de U-cursos el archivo *modules2018-1b.tgz*. Descomprímalo con el comando:

```
$ tar zxvf modules2018-1b.tgz
```

Contiene enunciados y soluciones de tareas de semestres anteriores con instrucciones para compilarlas y ejecutarlas (ver archivos README.txt en cada directorio). Le serán de especial utilidad los directorios *Syncread* con el enunciado y la solución de la tarea 3 del semestre otoño de 2013 (que se vio o se verá en clase auxiliar) y *Pipe* con el enunciado y la solución de la tarea 3 del semestre otoño de 2017 que corresponde a un pipe compartido entre todos los procesos.

Baje también de U-cursos *t3.zip* y descomprímalo con *unzip t3.zip*. Contiene el directorio *Vigia* con los archivos que Ud. necesitará para resolver esta tarea. Lea el archivo *README.txt*. Programe su solución en el archivo *vigia-impl.c*.

Ayuda

Siga las instrucciones de *https://users.dcc.uchile.cl/~lmateu/CC4302/relator/* para ver una presentación sobre cómo programar módulos y drivers. Al final se explican los mutex y condiciones que Ud. necesita para resolver la tarea. 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 de 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. Abuse del comando *printk* para escribir en los logs del núcleo lo que hace su driver. Le ayudarán a depurar los errores que seguramente cometerá. No intente usar ddd o gdb.

Base su solución modificando pipe-impl.c. Ud. debe agregar nuevas variables globales, cambiar la función *pipe_write* y hacer cambios menores al resto del driver. La función *pipe_read* queda igual, salvo quizás por algunos cambios de nombre a variables y llamadas a *printk* que Ud. prefiera agregar o quitar.

Entrega

La tarea se entrega *funcionando* en U-cursos. Para ello entregue solo el archivo *vigia-impl.c* que implementa el driver pedido. Se descontará medio punto por día de atraso, excepto días sábado, domingo o festivos. Intente resolver la tarea antes del control 3, le servirá de estudio.