CC4302 Sistemas Operativos

Tarea 7 – Semestre Primavera 2022 – Prof.: Luis Mateu

En esta tarea Ud. deberá implementar un driver de Linux para el dispositivo */dev/prodcons* con el comportamiento que se muestra en la tabla de más abajo. El número *major* del driver debe ser 61. El problema es similar al de un productor y muchos consumidores con un buffer de tamaño 1. El siguiente ejemplo usa el comando estándar de Unix *cat* para demostrar qué se espera de */dev/prodcons*. Su driver debe reproducir exactamente el mismo diálogo. 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 cuándo 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
$ **cat > /dev/prodcons** *(1)*			
	$ **cat /dev/prodcons** *(2)*	$ **cat /dev/prodcons** *(2)*	
no creo en el horoscopo *(3)*	no creo en el horoscopo		
porque soy cancer *(4)*		porque soy cancer	
			$ cat/dev/prodcons *(5)*
y los cancer *(6)*	y los cancer		
somos escépticos *(7)*		somos escépticos	
defina universo *(8)*			defina universo
y de 2 ejemplos	y de 2 ejemplos		
	control-C *(9)* $		
para entender la recursividad		para entender la recursividad	
hay que entender			hay que entender
la recursividad		la recursividad ***control-C*** *(9)* $	
control-D *(10)* $			***control-C*** *(9)* $

Notas:

- En (1) se invoca el comando *cat* redirigiendo la salida estándar a */dev/prodcons*. Este comando ejecuta un ciclo leyendo de la entrada estándar (o el archivo que se especifique como parámetro) y escribiéndolo tal cual en la salida estándar hasta el fin de la entrada estándar. En este caso leerá una línea de la entrada estándar (el teclado) y como el usuario no ha escrito nada todavía (con el teclado), *cat* se quedará en espera.
- En (2) se invocan 2 comandos *cat* para que lean de */dev/prodcons*. Usan *read* para leer de */dev/prodcons*. Como aún no se escribe nada con *write* en */dev/prodcons*, su driver debe hacer que *read* espere.
- En (3) el *cat* del shell 1 lee una línea del teclado y la escribe tal cual con *write* en */dev/prodcons*. En este instante hay 2 *read* pendientes, el del shell 2 y el del shell 3. Su driver debe hacer que el *read* que lleva más tiempo esperando, es decir el *read* del shell 2, retorne la línea ingresada desde el teclado. El *read* del shell 3 continúa esperando. El *cat* del shell 2 obtendrá la línea y la mostrará en la salida estándar (el terminal), y leerá otra línea con *read* de */dev/prodcons*. Su driver debe hacerlo esperar nuevamente. Por último, el *cat* del shell 1 se queda nuevamente esperando leer del teclado.
- En (4) el *cat* del shell 1 vuelve a leer una línea del teclado y la escribe tal cual con *write* en */dev/prodcons*. En este instante otra vez hay 2 *read* pendientes, pero ahora es el shell 3 el que lleva más tiempo esperando y por lo tanto su driver debe entregar la línea al *cat* del shell 3. El *cat* del shell 3 muestra esa línea, y vuelve a leer

/dev/prodcons con *read*, y su driver lo deja en espera. El *cat* del shell 1 vuelve a quedar esperando leer del teclado.

- En (5) se invoca otro comando *cat* para que lea de */dev/prodcons*. Ahora hay 3 *read* pendientes, siendo el shell 4 el más reciente.
- En (6) *cat* lee una línea y la escribe en */dev/prodcons*. Su driver se la debe entregar al *read* pendiente del shell 2 porque es el más antiguo. Recuerde que el comando *cat* ejecuta un ciclo y por lo tanto cuando *read* retorna, escribe lo leído en su salida estándar y vuelve a leer con *read*.
- En (7) su driver debe entregar la línea leída del teclado al *read* pendiente del shell 3 porque es el más antiguo.
- En (8) su driver debe entregar la línea leída del teclado al *read* pendiente del shell 4 porque ahora ese es el más antiguo.
- En (9), el *control-C* envía la señal SIGINT al proceso *cat*. No se puede procesar la señal todavía porque el proceso está en espera en la llamada al sistema *read*. Si su driver espera en un *c_wait*, *c_wait* solicitará la propiedad del mutex y retornará de inmediato un valor distinto de 0. Si su driver espera en un *down_interruptible*, este retornará de inmediato un valor distinto de 0. Si no es por la llegada de una señal, *c_wait* y *down_interruptible* retornan 0. Si el proceso recibe una señal mientras espera con *c_wait* o *down_interruptible*, la operación *read* o *write* de su driver debe retornar -EINTR (código de retorno para *interrupted system call*), abortando la operación. Esto provocará que el comando *cat* termine.
- En (10), el *control-D* hace que la lectura del teclado del *cat* del shell 1 entregue 0 bytes. Los programas interpretan la lectura de 0 bytes como el fin de archivo. En este caso *cat* termina al detectar el fin de la entrada estándar.

Recursos

Estudie el material envasado sobre módulos correspondiente a estas cátedras. Descargue además los ejemplos de módulos incluidos en los archivos `modules2020-2.tgz`. Descomprímalos como se explica en esa misma página. 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 sobre módulos del semestre otoño de 2013 (que se vio o en clase auxiliar) y *Pipe* con el enunciado y la solución de la tarea del semestre otoño de 2017 que corresponde a un pipe compartido entre todos los procesos, explicada en el material envasado con el relator.

Descargue los archivos adjuntos a esta tarea y descomprímalos. Programe su solución en el archivo *prodcons-impl.c* del directorio *T7*. Ahí encontrará un *Makefile* para compilar su tarea, las instrucciones para crear el dispositivo */dev/prodcons* en *README.txt* y la implementación de mutex a partir de los semáforos del núcleo de Linux.

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 (como usuario *root* puede ver los logs con el comando *dmesg*). Le ayudarán a depurar los errores que seguramente cometerá. No intente usar ddd o gdb.

Ayuda

Para las operaciones *read* y *write* de su driver, implemente un productor y múltiples consumidores con un buffer de tamaño 1 usando los *KMutex*. Puede usar números para atender las lecturas por orden de llegada, **pero en mi solución no fue necesario** para reproducir el mismo diálogo de este enunciado. No necesita hacer nada en las operaciones *open* y *release*.

Entrega

La tarea se entrega *funcionando* por U-cursos. Para ello entregue solo el archivo *prodcons-impl.c* que implementa el driver pedido. No entregue archivos binarios. Recuerde descargar de U-cursos el archivo que entregó y revisar que sea la versión correcta. Se descontará medio punto por día de atraso, excepto días sábado, domingo o festivos.