Recuperatorio Segundo Parcial

Primer Cuatrimestre 2024

Normas generales

- El parcial es INDIVIDUAL
- Puede disponer de la bibliografía de la materia y acceder al repositorio de código del taller de system programming, desarrollado durante la cursada
- Las resoluciones que incluyan código, pueden usar assembly o C. No es necesario que el código compile correctamente, pero debe tener un nivel de detalle adecuado para lo pedido por el ejercicio.
- Numere las hojas entregadas. Complete en la primera hoja la cantidad de hojas entregadas
- Entregue este enunciado junto al examen. El mismo no se incluye en el total de hojas entregadas.
- Luego de la entrega habrá una instancia coloquial de defensa del examen

Régimen de Aprobación

- Para aprobar el examen es necesario obtener cómo mínimo **60 puntos**.

NOTA: Lea el enunciado del parcial hasta el final, antes de comenzar a resolverlo.

Enunciado

En un sistema similar al que implementamos en los talleres del curso (modo protegido con paginación activada), se tienen varias tareas en ejecución. Similarmente al taller, todas las tareas disponen de un área de memoria compartida en la dirección virtual `TASK_LOCKABLE_PAGE_VIRT` que en caso de estar mapeada apunta a `TASK_LOCKABLE_PAGE_PHY`. La idea general del ejercicio es mejorar la forma en que las tareas acceden a la memoria compartida, generando un mecanismo tal que en en un determinado momento dado una sola tarea pueda tener acceso a la misma.

El mecanismo en términos generales sería el siguiente:

- Una tarea que quiera acceder a la memoria compartida, deberá solicitar por medio de una syscall (**lock**) el acceso exclusivo a la misma.
- Si la memoria compartida no está siendo utilizada por otra tarea, la tarea que solicitó el acceso podrá acceder a la misma (*adquirir el lock*)
- Si otra tarea posee el lock, la tarea que solicitó el acceso deberá abandonar la ejecución inmediatamente (es decir, no deberá esperar todo el slot de tiempo de ejecución correspondiente) y el scheduler debe seguir con la siguiente tarea.

- La tarea que quiere acceder al lock de la memoria compartida, verificará el acceso cada vez que el scheduler la ponga nuevamente en ejecución.
- Notar que una vez que la syscall es invocada, el nivel de usuario no vuelve a recibir el control *hasta que el lock es adquirido*. La syscall lock no debe fallar, desde el punto de vista del nivel de usuario. Si retorna, es porque el lock fue adquirido.
- La tarea que tiene el lock de la memoria compartida, deberá liberarlo (*liberar el lock*) por medio de otra syscall (**release**).
- Una vez que la tarea que solicitó el acceso a la memoria compartida la libere, otra tarea podrá acceder de forma exclusiva a la misma, adquiriendo el lock.
- Puede haber más de una tarea esperando por el acceso a la memoria compartida.
- No hay un sistema de turnos, simplemente se le dará acceso a la memoria compartida a la tarea que primero vuelva a solicitarlo, una vez que se libere el lock.

Se recomienda organizar la resolución en el orden en que aparecen los ejercicios e ir explicando las decisiones que se toman.

Aclaraciones:

- Para este sistema modificado se asume que `TASK_SHARED_PAGE` de los talleres se denomina `SYSTEM_INFO`, y no intervendrá en las modificaciones a realizar.

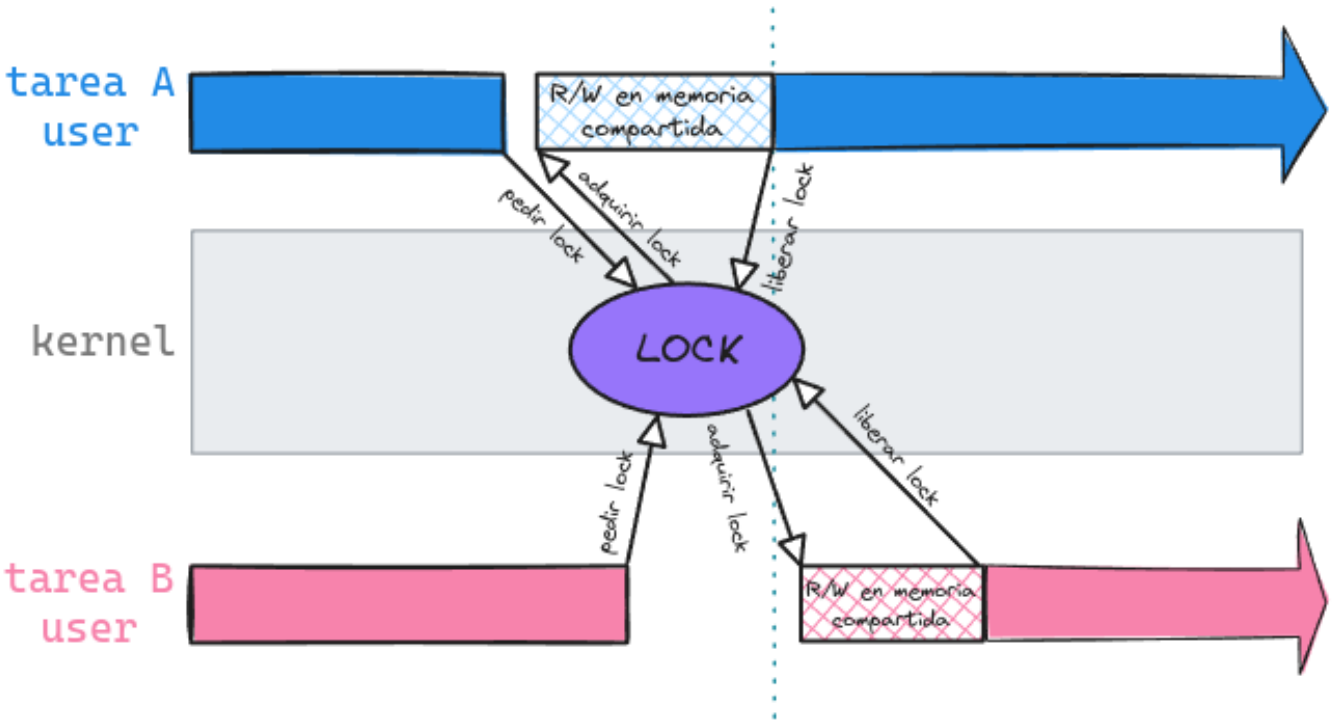

Figura 1: Mecanismo de locking de la memoria compartida.

Ejercicio 1 - (35 puntos)

En este punto haremos la función `get_lock` para el mecanismo de acceso a la memoria compartida. Esta función será llamada por una de las syscalls que desarrollaremos en el próximo punto.

a. La función `get_lock` tiene la siguiente firma:

```
void get_lock(vaddr_t shared_page);
```

Debe hacer lo siguiente:

- Comprobar si la dirección virtual `shared_page` corresponde a la página compartida. En caso contrario no deberá hacer nada.
- Modificar un flag, accesible por el kernel en todos los contextos, que indique que la memoria compartida está bloqueada por la tarea invocante. Es decir, dicho flag debe identificar a la tarea que posee el lock.
- Desmapear la página compartida de todas las tareas que no posean el lock.

Ejercicio 2 - (65 puntos)

Ya definida las función `get_lock` en el ejercicio anterior, se pide implementar lo siguiente:

a. Las syscalls invocables desde nivel de usuario:

- Una syscall (*lock*) que permita a una tarea solicitar el acceso exclusivo a la memoria compartida. La syscall debe recibir la dirección virtual de la página compartida.
- Una syscall (*release*) que permita a una tarea liberar el acceso exclusivo a la memoria compartida. La syscall debe recibir la dirección virtual de la página compartida.
- Introducir las modificaciones necesarias en el código existente para que si una tarea quiere **leer** de la página compartida y el lock está disponible (es decir, no está adquirido por ninguna otra tarea), pueda hacerlo sin necesidad de invocar la syscall *lock*.
  - Evaluar y especificar como se detecta si un acceso inválido es de lectura o escritura.

Implementar TODAS las modificaciones necesarias en el código del taller para que dichas syscalls puedan ser invocadas por cualquier tarea de nivel 3.

Aclaraciones:

- Recordar que si la tarea invocante de la syscall *lock* no puede acceder a la memoria compartida, debe ser bloqueada y el scheduler debe continuar con la siguiente tarea.
- Si una tarea que ya tiene el lock, invoca a la syscall *lock*, la tarea debe seguir su ejecución normalmente.
- Si una tarea que no tiene el lock, invoca a la syscall *release*, la tarea debe seguir su ejecución normalmente.

b. En el punto anterior asumimos un comportamiento "correcto" de las tareas, dónde sólo intentarán accesos de lectura sin obtener el lock. Dado que en el ejercicio 1 la función `get_lock` desmapea la página compartida, se pide extender el mecanismo para que funcione de forma automática en el caso de que una tarea intente escribir en dicha página sin llamar a la syscall *lock*:

   - Un acceso a la memoria compartida por parte de una tarea que no posee el lock, debe comportarse como si la tarea hubiese solicitado el lock, de manera transparente para el nivel de usuario.
   - El lock adquirido se liberará también automáticamente luego de 5 desalojos de la tarea que lo obtuvo.

   En este ejercicio **no es necesario entregar pseudocódigo detallado**. Basta con plantear la idea general de la potencial implementación de este mecanismo.

A tener en cuenta para la entrega (para todos los ejercicios):

- Está permitido utilizar las funciones desarrolladas en los talleres.
- Es necesario que se incluya una explicación con sus palabras de la idea general de las soluciones.
- Es necesario escribir todas las asunciones que haga sobre el sistema.
- Es necesaria la entrega de código o pseudocódigo que implemente las soluciones, salvo que algún ítem especifique lo contrario.