Arquitectura y Organización de Computadoras - DC - UBA

Recuperatorio de Segundo Parcial

Segundo 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 esta hoja junto al examen. La misma no se incluye en el total de hojas entregadas.

Régimen de Aprobación

- Para aprobar el examen es necesario demostrar que adquirieron conocimientos de **todos** los temas trabajados en la segunda mitad de la materia, esto es: **interrupciones**, **paginación** y **tareas**. La solución propuesta debe trabajar todos estos aspectos.

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

Contexto

En nuestro afán por obtener un premio *Turing*, estamos trabajando en el desarrollo de un sistema operativo revolucionario; en base al kernel que desarrollamos en la materia.

Nuestro sistema intenta plantear un potencial incremento en la velocidad de comunicación entre procesos mediante una idea radical: en lugar de compartir páginas de memoria, vamos a intercambiar (*swap*) **todos** los **registros de propósito general** a nivel usuario entre 2 tareas, excluyendo a `EBP`, y obviamente a `ESP` y `EIP`.

Para que este intercambio se pueda dar, se deben cumplir los siguientes pasos:

- Una tarea (tarea origen) llamará a una syscall `swap` con el `id` de la tarea con la que desea intercambiar registros (tarea destino).
- Si la tarea destino también llamó anteriormente a `swap` con el `id` de la tarea origen, se realizará el intercambio de registros.

Primer ejercicio

a) Programar una syscall `swap` que permita intercambiar sus registros con la tarea destino de forma *bloqueante*, esto es:

- La tarea que llama a `swap` debe continuar su ejecución solamente cuando los registros ya fueron intercambiados

b) Programar una syscall `swap_now` que permita intercambiar sus registros con la tarea destino de forma *no bloqueante*, esto es:

- La tarea que llama a `swap_now` debe continuar su ejecución en la próxima ronda del scheduler. En este caso, cuando retoma la ejecución pueden haber ocurrido dos cosas:
  - Los registros fueron intercambiados
  - Los registros no fueron intercambiados y no se intercambiarán, salvo que se llame nuevamente a `swap_now`

Ambas syscalls deben funcionar en forma conjunta.

Aclaración: Ante una llamada a `swap_now`, puede haber 2 desenlaces posibles:

- La tarea destino había llamado a `swap` con el `id` de la tarea origen (o a `swap_now` en la misma ronda de scheduler), por lo tanto, se intercambian los registros y se continúa con la ejecución de la tarea llamadora.
- La tarea destino no había llamado a `swap` (o `swap_now`) por lo tanto la tarea llamadora abandona la ejecución y la retoma en la siguiente ronda de scheduler. En este caso los registros pueden haber sido intercambiados o no, según si alguna otra tarea llamó a `swap` o `swap_now` con el `id` de la tarea llamadora.

En el caso de `swap` se espera el mismo comportamiento, **salvo** en el segundo ítem anterior dónde la ejecución de la tarea llamadora se retoma únicamente cuando los registros fueron intercambiados.

Un ejemplo:

- La tarea con `id=3` llama a `swap` con el `id` destino 2.
- Luego de varios ciclos de scheduler, la tarea con `id=2` llama a `swap_now` con el `id` destino 3, por lo tanto se intercambian los registros y la tarea con `id=2` continúa con su ejecución.
- En la siguiente ronda de scheduler, la tarea con `id=3` continúa su ejecución con los registros de la tarea con `id=2` y viceversa.

Segundo ejercicio

Mediante pruebas intensivas, nos dimos cuenta que las tareas al no saber si se realizó el intercambio de registros, fallaban estrepitosamente. Por lo tanto, decidimos que las syscalls `swap` y `swap_now` informen a las tareas si se realizó el intercambio o no. Para ello, las syscalls modificarán una variable de cada tarea, alojada en la dirección virtual de memoria `0xC001C0DE` con el valor `1` si se realizó el intercambio y `0` en caso contrario. Dado que el kernel de nuestro sistema hace verificaciones periódicas de dicha variable para todas las tareas en ejecución, la actualización de dicha variable debe hacerse **de forma inmediata** para las tareas correspondientes luego de intercambiar los registros.

a) Modificar las syscalls para que realicen lo descripto.

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

- Indicar **todas las estructuras de sistema** que deben ser modificadas para implementar las soluciones.
- 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 que implemente las soluciones.