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 esta hoja junto al examen. La misma 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**.
- Para promocionar es condición suficiente y necesaria obtener como mínimo **80 puntos** tanto en este examen como en el primer parcial

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

Enunciado

Ejercicio 1 - (50 puntos)

En un sistema similar al que implementamos en los talleres del curso (modo protegido con paginación activada), se tienen varias tareas en ejecución. Se desea agregar al sistema una syscall que le permita a la tarea que la llama espiar la memoria de las otras tareas en ejecución. En particular queremos copiar 4 bytes de una dirección de la tarea a espiar en una dirección de la tarea llamadora (tarea espía). La syscall tendrá los siguientes parámetros:

- El selector de la tarea a espiar.
- La dirección virtual a leer de la tarea espiada.
- La dirección virtual a escribir de la tarea espía.

Si la dirección a espiar no está mapeada en el espacio de direcciones de la tarea correspondiente, la syscall deberá devolver -1 en `eax`, por el contrario, si se pudo hacer correctamente la operación deberá devolver 0 en `eax`.

Se pide:

a. Definir o modificar las estructuras de sistema necesarias para que dicho servicio pueda ser invocado.

b. Implementar la syscall y especificar claramente la forma de pasarle los parámetros correspondientes.

Se recomienda organizar la resolución del ejercicio realizando paso a paso los items mencionados anteriormente y explicar las decisiones que toman.

Aclaraciones:

- Se puede asumir que el selector de tarea pasado es válido.
- Se puede asumir que la dirección de destino de la tarea llamadora se encuentra correctamente mapeada.

Ejercicio 2 - (50 puntos)

Partiendo del sistema trabajado en los talleres, se pide modificar la política del scheduler. El nuevo scheduler distingue tareas *prioritarias* de *no prioritarias*. Las tareas *prioritarias* son aquellas que, al saltar la interrupción del reloj, tengan el valor `0x00FAFAFA` en `EDX`. Las tareas **pausadas** y/o **no ejecutables** no pueden ser *prioritarias*.

La forma en la que el nuevo scheduler determina la siguiente tarea a ejecutar es la siguiente:

1. Si hay otra tarea *prioritaria* distinta se elige esa. En caso de haber más de una se hace de forma **round-robin** (como en el scheduler de los talleres).
2. Si no, se elige la próxima tarea como en el scheduler de los talleres.

La solución propuesta debe poder responder las siguientes preguntas:

- ¿Dónde se guarda el `EDX` de nivel de usuario de las tareas desalojadas por el scheduler?
- ¿Cómo determina el scheduler que una tarea es *prioritaria*?

Se recomienda organizar la resolución del ejercicio explicando paso a paso las decisiones que toman.

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.