Segundo Recuperatorio

Segundo Cuatrimestre 2022

Normas generales

- El parcial es INDIVIDUAL
- Se 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**.

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

Enunciado

Ejercicio 1 - (60 puntos)

En un sistema similar al que implementamos en los talleres del curso (modo protegido con paginación activada), se tienen 5 tareas en ejecución. Todas las tareas realizan una cuenta que, al finalizarla, dejan el resultado en EAX.

Se desea agregar al sistema una syscall para que las tareas puedan notificar al Kernel que terminaron (y que se marque internamente con un flag a dicha tarea). Luego, cuando el Kernel reciba el aviso de las 5 tareas, ejecutará una sexta que procesa los resultados (no nos interesa cómo... por ejemplo; podría sumarlos) y escribe el nuevo resultado final en el EAX de cada tarea.

Cuando una tarea avisa que finalizó, no se le otorgará más tiempo de CPU hasta que la sexta tarea que procesa los datos no haya completado su trabajo.

Cuando eso ocurra, se retomará la ejecución de las 5 tareas y el ciclo se repetirá indefinidamente.

Se pide:

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

b. Implementar la syscall que llamarán las tareas.

c. Dar el pseudo-código de la tarea que procesa resultados.

d. Mostrar un pseudo-código de la función `sched_next_task` para que funcione de acuerdo a las necesidades de este sistema.

Se recomienda organizar la resolución del ejercicio realizando paso a paso los items mencionados anteriormente.

Detalles de implementación:

- Las 5 tareas originales corren en nivel 3.
- La sexta tarea que lanza el Kernel tendrá nivel de privilegio 0.

Ejercicio 2 - (40 puntos)

Se desea implementar una modificación sobre un kernel como el de los talleres: en el momento de desalojar una tarea, se deben escribir a disco todas las páginas de dicha tarea que hayan sido modificadas mientras la tarea corría.

Se les pide que hagan una función que, dado el CR3 de la tarea a desalojar, devuelva un arreglo de direcciones **virtuales** con las páginas que deben ser escritas a disco para esta nueva funcionalidad.

La función a implementar es:

```
vaddr_t* paginas_modificadas(int32_t cr3);
```

Detalles de implementación:

- Si necesitan, pueden asumir que el sistema tiene segmentación *flat*.
- NO DEBEN modificar las estructuras del kernel para llamar a la función que están creando. Solamente deben programar la función que se pide.

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.