Arquitectura y Organización de Computadoras - DC - UBA

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 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.

Contexto

Nos encargaron el desarrollo de un kernel para la nueva consola Orga Génesis que cuenta con un procesador x86. Para ahorrar trabajo vamos a tomar nuestro kernel y expandirlo para permitir la ejecución de juegos mediante cartuchos, que contendrán el código y los recursos gráficos (sprites de personajes, fondos). Se incorpora entonces un lector de cartuchos que, entre otras cosas, copiará los gráficos del cartucho a un buffer de video de 4 kB en memoria. Cada vez que el buffer esté lleno y listo para su procesamiento, el lector se lo informará al kernel mediante una interrupción externa mapeada al IRQ 40 de nuestro x86.

Primer ejercicio

Distintas tareas se ocuparán de mostrar los gráficos en pantalla y de realizar otros pre/postprocesamientos de imagen on-the-fly.

Para esto, el sistema debe soportar que las tareas puedan acceder al buffer de video a través de los siguientes mecanismos:

- DMA (Direct Memory Access) - se mappea la dirección virtual `0xBABAB000` del espacio de direcciones de la tarea directamente al buffer de video.
- Por copia - se realiza una copia del buffer en una pagína física específica y se mapea en la dirección virtual `0xBABAB000`. Cada tarea debe tener una copia única.

El buffer de video se encuentra en la dirección física de memoria `0xF151C000` y **solo debe ser modificable por el lector de cartuchos.**

a) Programar una función `void buffer_dma(pd_entry_t* pd)` que dado el page directory de una tarea realice el mapeo del buffer en modo DMA.

b) Programar una función `void buffer_copy(pd_entry_t* pd, paddr_t phys)` que dado el page directory de una tarea realice la copia del buffer a la dirección física pasada por parámetro y realice el mapeo correspondiente.

Segundo ejercicio

Las tareas configuran en memoria una variable que sirve para comunicar al sistema de qué manera van a acceder al buffer de video.

Cada tarea guarda en la dirección virtual `0xACCE50` (mapeada como r/w para la tarea) un `uint8_t acceso` con posibles valores 0, 1 y 2. El valor 0 indica que la tarea no accederá al buffer de video, 1 que accederá mediante DMA y 2 que accederá por copia. De acceder por copia, la dirección virtual donde realizar la copia estará dada por el valor del registro `ecx` al momento de llamar a `opendevice`.

Cada vez que se indique como completo el buffer, se deberá mappear el mismo a las tareas que **utilicen DMA** y hayan **solicitado acceso** al buffer, o actualizar la copia del buffer de las tareas que **acceden por copia** y hayan **solicitado acceso** al buffer.

Es deseable que cada tarea que accede por copia mantenga una única copia del buffer para no ocupar la memoria innecesariamente.

Para solicitar acceso al buffer de memoria, las tareas deberán informar que desean acceder a él mediante una syscall `opendevice`. El sistema no debe retomar la ejecución de estas tareas hasta que se detecte que el buffer está listo y se haya realizado el mapeo DMA o la copia correspondiente. Una vez que la tarea termine de utilizar el buffer, deberá indicarlo mediante la syscall `closedevice`. Mientras esté en uso el buffer, la tarea deberá poder visualizar su valor actualizado, ya sea por DMA o por copia.

Se muestra el comportamiento esperado en el siguiente esquema, donde las flechas indican que la información actualizada del buffer se disponibiliza para la tarea señalada.

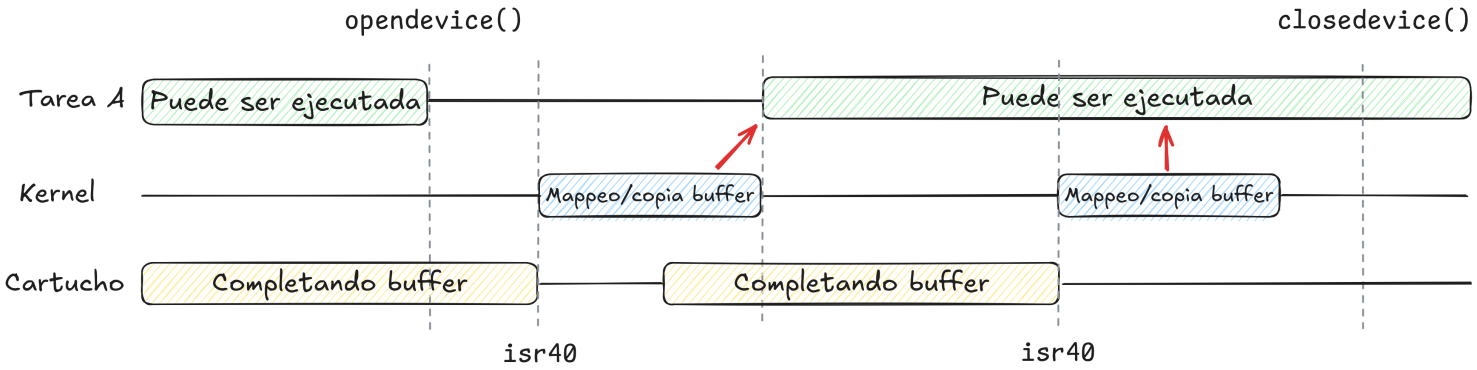

Figure 1: Diagrama schedule tarea, cartucho y kernel

Utilizando las funciones del primer ejercicio, se pide:

a) Programar la rutina que atenderá la interrupción que el lector de cartuchos generará al terminar de llenar el buffer.

Consejo: programar una función `deviceready` y llamarla desde esta rutina.

b) Programar las syscalls `opendevice` y `closedevice`.

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.