Segundo Parcial

Primer Cuatrimestre 2022

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

Enunciado - Tema 2

Ejercicio 1

Estructura general del sistema

En un sistema similar al que implementamos en los talleres del curso (modo protegido con paginación activada), se necesita implementar una llamada al sistema que permita modificar la próxima tarea a ejecutar de la forma solicitada.

La próxima tarea a ejecutar es aquella tarea, distinta a la actual, retornada por la llamada a `sched_next_task` en la rutina de interrupción del reloj.

Se busca ampliar el sistema para que una tarea pueda marcar como no accedidas a todas las páginas de usuario de la tarea próxima. Para ello se define una syscall que solicitará la ejecución de dicha operación **a la hora de realizar al cambio de tarea** (la solicitud y la modificación ocurren en diferido). Dicho en otras palabras, cuando la tarea próxima retome su ejecución, debe tener marcadas todas sus páginas de nivel de usuario como no accedidas. Naturalmente, si una tarea no llama a esta syscall, no se debe alterar la estructura de paginación de la tarea próxima.

1. Modificar las estructuras del sistema necesarias para que todas las tareas puedan invocar a esta nueva syscall, indique las modificaciones que introduciría, y en qué archivos del sistema implementado en los talleres las harían.
2. Implementar la rutina correspondiente a dicha syscall y definir variables externas si fuese necesario.
3. Modificar la rutina de interrupción de reloj para que si se llamó a la syscall en la tarea actual, se haga la modificación correspondiente sobre la estructura de paginación de la tarea próxima.

Ejercicio 2

En un sistema con características similares al del taller, diseñe una función mediante la cual una tarea de sistema (nivel 0), pueda obtener el número de tareas que tienen mapeadas una cierta dirección virtual. La firma de la función es la siguiente:

```
uint32_t getMappings(uint32_t virtual, gdt_entry_t* gdt);
```

Estando declarado el tipo `gdt_entry_t`, como en el taller. Se repite su declaración aquí:

```
typedef struct str_gdt_entry {
        uint16_t limit_15_0;
        uint16_t base_15_0;
        uint8_t base_23_16;
        uint8_t type : 4;
        uint8_t s : 1;
        uint8_t dpl : 2;
        uint8_t p : 1;
        uint8_t limit_19_16 : 4;
        uint8_t avl : 1;
        uint8_t l : 1;
        uint8_t db : 1;
        uint8_t g : 1;
        uint8_t base_31_24;
} __attribute__((__packed__, aligned(8))) gdt_entry_t;
```

En caso de utilizar tipos de datos, macros, definiciones y/o funciones provistas en el taller, enumere sus declaraciones y fundamente su uso.

Ejercicio 3

Un procesador x86 trabajando en Modo 32 bits ejecuta un sistema multitasking, con modo protegido y paginación activos (sin PAE). Las tareas ejecutan con CPL=11b y los handlers de excepción ejecutan en CPL=00b. Si se decide hacer task switch manual (es decir, sin usar los mecanismos automáticos con TSS propuestos por el procesador). ¿Es necesario utilizar TSS?. Justificar.