Segundo Parcial - Recuperatorio

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

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 crear una tarea idéntica a la tarea que invocó la llamada. El contexto y el mapa de memoria de la nueva tarea, se explicará a continuación. Se pide:

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 de manera que:

(a) Se genere una nueva tarea cuyo contexto inicial sea una copia del **último contexto almacenado** de la tarea actual y que pueda ser ejecutada en algún momento por el sistema.

(b) Se realicen los mismos mapeos para código, datos y pila en todos los niveles (kernel y usuario), con la diferencia que las páginas de datos y pila de usuario quedarán en sólo lectura. Explique primero el algoritmo con sus palabras y luego codifíquelo. Recuerde que cada tarea debe tener mapeado al kernel con mapeo identidad y además, que cada tarea deberá tener su propia copia física de la pila de nivel 0. Suponga que la pila de nivel 0 de cada tarea ocupa una única página en el área del kernel.

3. Expliquen con sus palabras un mecanismo que permita generar una copia física de la página cuando se intente escribir en las páginas de datos y/o las de pila de nivel de usuario de la nueva tarea. La nueva página debe permitir las escrituras.

Para obtener nuevas páginas físicas del nivel correspondiente, puede usar las rutinas del taller: `mmu_next_free_kernel_page` y `mmu_next_free_user_page`. Además de estas funciones, puede usar cualquier función definida en el taller como parte de la solución.

Ejercicio 2

A continuación se tiene las entradas de una `TLB` correspondientes a una misma tarea que ejecuta en un sistema basado en un procesador x86. Las entradas están en el orden en el que se han ido generando las traducciones. Para dicha tarea el registro `CR3 = 0x000E4000`, y las tablas de página correspondientes a las direcciones en uso están inmediatamente contiguas al `DTP`, en el orden en el que se han ido requiriendo. Se pide:

#	Nro.Pag.	Descriptor	Ctrl
1	7C047	0EF00121	ccc
2	7EEF0	1EF01067	ccc
3	EC004	001F0163	ccc
4	EC005	001F7123	ccc
5	46104	1F011005	ccc
6	46109	1F010027	ccc

1. Para las entradas 1 y 2 de la `TLB`, escribir el contenido de sus correspondientes descriptores en cada nivel de la jerarquía de tablas de traducción a direcciones físicas, considerando que se trata de las dos primeras páginas requeridas al ejecutar la tarea.

2. Especificar para cada `PDE` que valor deben tener los bits `U/S` y `R/W`, para ser consistentes con el contenido de la TLB. Respuestas posibles en cada caso: 0, 1, o X (indistinto).

3. Suponiendo que las seis entradas están siendo utilizadas por una misma tarea, y por cada task switch se modifica `CR3`. ¿Que ocurre con cada una al momento del task switch?. ¿Cual es el tratamiento para aquellas que se han modificado?

4. ¿Cual es la función dentro de la `TLB` de los bits identificados genéricamente como `Ctrl`?.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Address of page directory[1]																				Ignored							PCD	PWT	Ignored			**CR3**
Bits 31:22 of address of 4MB page frame										Reserved (must be 0)					Bits 39:32 of address[2]				PAT	Ignored			G	**1**	D	A	PCD	PWT	U/S	R/W	**1**	**PDE: 4MB page**
Address of page table																				Ignored				**0**	Ign	A	PCD	PWT	U/S	R/W	**1**	**PDE: page table**
Ignored																															**0**	**PDE: not present**
Address of 4KB page frame																				Ignored			G	PAT	D	A	PCD	PWT	U/S	R/W	**1**	**PTE: 4KB page**
Ignored																															**0**	**PTE: not present**

Ejercicio 3

Explicar cómo funciona el handshake de Interrupt Acknowledge entre el procesador x86 y los PIC 8259. Detallar cómo obtiene el procesador el tipo de cada interrupción, y quien lo provee en cada caso.