SOA2 (21/12/2023)

Nombre:
DNI:

Segundo control de teoría

Responde brevemente a las siguientes preguntas justificando tus respuestas. **Una respuesta sin justificar no será dada como válida.**

1. (3 puntos) Threads

Queremos modificar la implementación de threads de Linux (o ZeOS) vista en clase para que, en vez de que el task_struct identifique un thread y el proceso sea abstracto, tener una estructura donde guardaremos los datos del proceso (llamada EPROCESS) y otra donde guardaremos los datos del thread (llamada ETHREAD).

a) Indica que campos, de los vistos en clase, tienen que ir en el EPROCESS

b) Indica qué campos, de los vistos en clase y en el laboratorio, tienen que ir en el ETHREAD.

También modificamos el planificador del sistema para que siempre se seleccione el thread más prioritario del proceso más prioritario. Para ello, dentro del EPROCESS y del ETHREAD incluimos un campo de prioridad para que exista una prioridad por proceso y otra por thread. En este momento puedes suponer que existen llamadas al sistema y mecanismos para poder poner y modificar estas prioridades.

Recuerda que siempre buscamos una implementación eficaz y eficiente.

c) Nos planteamos una implementación con un **planificador a un nivel**.

c.1) ¿Qué contendrá la cola de ready: estructuras EPROCESS o estructuras ETHREADS?

c.2) ¿Cómo funciona la gestión de la cola de ready en el caso de que queramos indicar que hay un nuevo thread en ready?

d) Nos planteamos una implementación con un **planificador a dos niveles**.

d.1) ¿Qué contendrá la cola de ready: estructuras EPROCESS o estructuras ETHREADS?

d.2) ¿Cómo funciona la gestión de la cola de ready en el caso de que queramos indicar que hay un nuevo thread en ready?

Por último, queremos evitar tener un wrapper de la función del thread. Si recuerdas la estructura de la pila de usuario que se monta en sys_pthread_create, sabemos que si no se invoca la llamada al sistema pthread_exit desde dentro de la función del thread, se producirá una excepción de page_fault ya que se intentará ejecutar la instrucción que está en la dirección lógica 0.

e) Escribe el codigo de la excepción page_fault teniendo en cuenta que tiene que ser capaz de detectar este caso para saber que se ha acabado un thread pero también tiene que ser capaz de detectar si el page_fault viene por un acceso incorrecto a memoria.

Nombre:
DNI:

2. (4 puntos) Sistema de ficheros en disco

Estamos implementando un driver para poder interpretar múltiples sistemas de ficheros en disco. Para esto, la única función que nos ofrece la buffer cache del sistema para poder acceder a disco es:

char *GetBlock(unsigned long BlockID);

que dado un identificador de bloque de disco, *BlockID*, nos devuelve un puntero a la zona de memoria donde ha almacenadoel contenido de ese bloque de disco.

Los identificadores de bloque son de 4 bytes, el tamaño de bloque de disco es BLOCK_SIZE. A no ser que se diga lo contrario en el enunciado, puedes suponer que dentro de un Inodo existe un campo llamado FirstBlockID que contiene el identificador del primer bloque de datos de ese fichero y un campo FileSize que contiene el tamaño, en bytes, del fichero.

a) Queremos implementar la siguiente función:

unsigned long GetBlockID(struct i_node * inode, unsigned long long offset);

que dado un fichero identificado por su Inodo (parámetro inode) y un offset dentro del fichero, nos devuelve el identificador del bloque que contiene ese offset. Para los siguientes apartados no hace falta incluir el código de comprobación de errores. Puedes suponer que el offset siempre estará entre 0 y FileSize.

a.1) Escribe el código de la función si la asignación de bloques a ficheros es una asignación contigúa.

a.2) Escribe el código de la función si la asignación de bloques a ficheros es una asignación encadenada (sin tabla).

a.3) Escribe el código de la función si la asignación de bloques a ficheros es una asignación indexada.

b) Suponiendo que ahora implementamos una asignación encadenada en tabla.

b.1) Escribe en C la declaración de la variable FAT suponiendo que hay TOTAL_BLOCKS bloques de disco.

b.2) Escribe el código de la función GetBlockID con asignación encadenada en tabla.

Nombre:
DNI:

c) Rellena la siguiente tabla indicando, en cada casilla, el numero de bloques de disco que se necesitan para almacenar un fichero con el tamaño en bytes indicado, utilizando cada una de las asignaciones indicadas. Supón que el tamaño de bloque es de 4 KBs.

Tamaño (en bytes)	Contigua	Encadenada	FAT	Indexada	Indexada multinivel
400					
8190					
17000					
81920					

3. (3 puntos) Gestores

Disponemos en nuestro sistema operativo de un sistema de ficheros virtual que abstrae dos sistemas de ficheros: FAT y EXT2. Para ello, tanto el sistema de ficheros virtual como los sistemas de ficheros FAT y EXT2 disponen, cada uno de ellos, de un gestor que realiza las operaciones de entrada/salida. El gestor del sistema de ficheros virtual (VFS) traduce las peticiones que se realizan a peticiones a los gestores de FAT y EXT2 que son los que acaban accediendo a las particiones. El gestor VFS es multiprogramado y los gestores de FAT y EXT2 son monoprogramados. Los gestores de FAT y EXT2 solamente son accesibles a través del gestor de VFS. Un gestor multiprogramado es aquel que tiene varios threads que pueden atender peticiones en paralelo. Un gestor monoprogramado solo tiene un thread para atender peticiones.
Se busca una implementación lo más eficiente y eficaz posible que aumente el paralelismo en el acceso al sistema de ficheros, reduciendo el uso de recursos (memoria, semaforos, threads, ...)

a) Indica cuantas colas de iorbs e iofin se necesitan en esta configuración.

b) Indica cuantas peticiones pendientes de ser procesadas puede tener el gestor de VFS

c) Indica cuantas peticiones pendientes de ser procesadas puede tener, como máximo, el gestor de FAT.

d) Indica cuantos semáforos se requieren con esta configuración para trabajar con el gestor de EXT2.

e) ¿Dónde está el cuello de botella de esta configuración que evita que se procesen más peticiones en paralelo?

f) Escribe el código de uno de los threads que atienden peticiones del gestor de VFS

SOA2 (21/12/2023)

Nombre:
DNI: