D.S.O

1ª Práctica. Curso 2000-2001

Implementación de un sistema de buffer caché para disco.

Objetivo.- Simular el funcionamiento del caché de disco implementado en el sistema operativo UNIX.

Implementación.- El buffer caché se implementará asignando inicialmente una región de memoria compartida de tamaño configurable. A partir de aquí, todo proceso que desee acceder a disco lo hará a través de este buffer caché siguiendo la misma política que utiliza Unix. Por tanto, tendremos que reescribir estas funciones de acceso (abrir fichero, leer, escribir, posicionar, cerrar) de modo que se utilice este mecanismo.

El buffer caché estará compuesto por un conjunto de buffers de tamaño constante. Cada uno de los buffers estará formado por una cabecera y los datos correspondientes al bloque de disco (p. ej., utilizaremos como tamaño de bloque de disco de 10 bytes). Cada buffer estará asociado de forma unívoca a un bloque de disco y la forma que identificaremos a cada uno de ellos será a través del conjunto "nº de file system, nº de inodo, nº de bloque dentro del fichero". Además para la gestión del caché se utilizarán dos listas, la "cola hash" para la búsqueda rápida de un buffer y la lista de buffers libres o "free-list" implementada de acuerdo a un algoritmo LRU.

Una vez definido el buffer caché, implementar las nuevas funciones para el acceso a disco. Con este fin, se hace necesario la utilización de otras estructuras. Por una parte, se asignará otra región de memoria compartida que va a actuar como tabla de ficheros. Cada entrada dentro de esta región tendrá el mismo significado que en Unix (permisos de acceso, puntero E/S, contador de referencia, nº inodo al que apunta).

Por otra parte, cada proceso tendrá su estructura interna de datos que vamos a denominar como área u. En esta estructura, almacenaremos la siguiente información: Tabla de descriptores de ficheros de usuario, con una entrada por cada fichero abierto por el proceso, una estructura que contenga la información necesaria para realizar las operaciones de E/S (Operación: E-S, Nº bytes restantes, posición de memoria de usuario donde leer o escribir, puntero E/S).

Las funciones a implementar que se corresponderían con nuestras llamadas al sistema van a ser las siguientes:

abrir fichero, leer, escribir, posicionar y cerrar.

Además, para la implementación de estas llamadas al sistema será necesario la creación de las rutinas de más bajo nivel que implementen los algoritmos (getblk, brelse, bread, bwrite, bmap) poniendo especial atención en los bloqueos de buffers y gestión de procesos que compiten por recursos (poner procesos a dormir, despertar procesos cuando se libere un recurso, etc)

Entrega: Código fuente y ficheros auxiliares (disponible en ftp)
Informe con detalles de la implementación y algoritmos empleados en pseudocódigo.
Fecha límite: Día anterior al primer llamamiento de junio.
Plataforma: Unix System Vr4 (gofio, tonique,..)
Linux

Opcional: Sustituir la realización de esta práctica por la implementación de Berkeley (Interesados hablar con el profesor).