**INSTITUTO TECNOLÓGICO DE PABELLON DE ARTEGA**



**TECNOLOGÍCO DE PABELLÓN DE ARTEAGA**

 “”

**UNIDAD 5**

**LECTURA**



*Presenta:* Ricardo Montoya Gómez

*Docente:* Eduardo Flores Gallegos

*Materia:* Sistemas Operativos 1

Pabellón de Arteaga, Ags., 31 de mayo del 2018

**EL "SISTEMA DE ARCHIVOS VIRTUAL" EN LINUX**

Los elementos de datos principales de cualquier sistema parecido a unix es el archivo, lo que diferencia es que cada elemento de datos lleva un nombre de ruta único que identifica cada archivo dentro del el sistema que está en ejecución. Cada archivo aparece como cualquier otro archivo de la forma de que se accede y se modifica, se utiliza los mismos comandos para poder llamar al archivo.

Lo anterior se aplica fuera de la media física que contiene la información. El bloqueo de almacenamiento físico de la información se hace repartiendo la transferencia de datos a diferentes controladores.

**La manera de UNIX**

Linux estudia el sistema de archivo e igual que lo hace unix, adoptando los conceptos de Super Block, inodos, directorios y archivos. Lo que hace es que es más accesible y en cualquier momento es más determinado por cómo se unen las diferentes partes, entonces cada parte es una partición del disco duro que está montado en el sistema.

* **El Super Block** desde que se usó el primer bloque de datos del disco o de la partición para almacenar más información sobre la partición que se creó. El Super Block ya no es un concepto de bloque de datos, pero todavía tiene información de cada sistema de archivos instalado. El kernel 2,0 mantiene una matriz estática de tales estructuras para manejar hasta 64 sistemas de archivos montados
* **Un i-nodo** junto a cada archivo. Este contiene la información de un archivo con nombre, menos su nombre y sus datos reales. Lo que almacena es el tamaño de los datos que contiene, el número de enlaces y otra información. La que se pretende es separa los archivos de la implementación de un enlace duro y el uso de anotaciones. Un i-nodo se puede describir como el kernel por un nodo de estructura.
* **El directorio** es un tipo de archivo que junta los i-nodo a los nombres de los archivos. Donde el kernel no tiene una estructura de datos especial para representar al directorio.
* **El archivo** en si esta junto a un i-nodo, por lo que los archivos son áreas de datos. pero también pueden ser directorios, dispositivos, FIFOs (primero en la primera salida) o Sockets. Un "archivo abierto" se describe en el kernel de Linux mediante un elemento de archivo struct.

**Orientación de objetos**

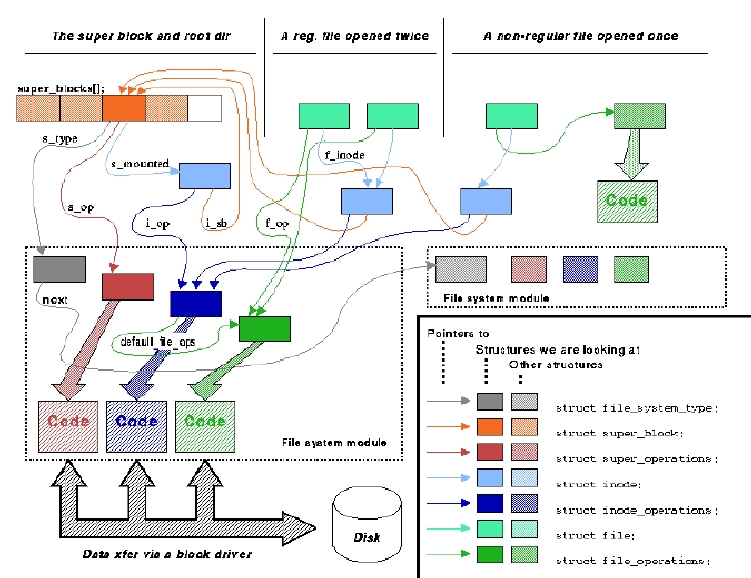
El sistema operativo tiene que ser apto para lidiar de diferentes maneras la información en el disco. Se puede buscar un diseño perfecto de la información y poder usarla para cada partición de un disco, todas las computadoras tienen que tener acceso a todos los discos duros.

El inconveniente de tener diferentes formatos de datos de forma transparente se ha solucionada haciendo superbloqueos, inodos y archivos en "objetos". El kernel no seguirá las instrucciones de switch para poder acceder a los diferentes esquemas.

En la práctica, una estructura súper bloque encierra un campo struct super\_operations \* s\_op, un i-nodo encierra struct inode\_operations \* i\_op y un archivo encierra struct File\_operations \* f\_op.

La implementación actual de Linux no da el uso de modulos para todos los tipos de sistemas de archivo, pero en el modo root el sistema de archivos debe ser instalado antes de cargar el modulo. La maquinaria nos permite usar un módulo antes de instalar el sistema de archivos root.

Esto es en Resumen Cómo todo el manejo de archivos ocurre para cualquier tipo de sistema de archivos dado, y se representa en la figura.



Su función es pasar información sobre el medio de almacenamiento que se está instalado y tiene que rellenar la estructura Super Block, así como cargar el i-nodo del directorio del sistema de archivos. El campo adicional requires\_dev es utilizado por el tipo de sistema de archivos para indicar si tendrá acceso a un dispositivo de bloque

la función se pasa información sobre el medio de almacenamiento que se está montando y se le pide que rellene una estructura Super Block, así como cargar el i-nodo del directorio raíz del sistema de archivos como SB-> s\_mounted (donde SB es el super-bloque acaba de llenar). El campo adicional, La estructura se muestra en el Listado 1.

**Listado 1**

El kernel utiliza la estructura super operations para leer y escribir i-nodos, escribir información de super bloque en disco y recopilar estadísticas. El super operations declarado por cada tipo de sistema de archivos se muestra en el listado 2.

**Listado 2**

Despues de hacer una copia de la memoria de i-nodo. El kernel se ejecutará sobre el mismo utilizando sus propios comandos struct inode\_operations. Las operaciones de gestión de directorios son parte de las operaciones de i-nodo porque la implementación de una estructura dir\_operations traería condicionales adicionales en el acceso al sistema de archivos.

**Listado 3**

El File\_operations, determina como se van a ejecutar los datos en el archivo real. Puesto que la misma estructura File\_operations se utiliza para actuar en dispositivos, también incluye algunos campos que sólo tienen sentido para los dispositivos de carácter o bloque. Es interesante notar que la estructura mostrada aquí es la estructura declarada en los kernels 2,0, mientras que 2,1 cambió los prototipos de lectura, escritura y lseek para permitir una gama más amplia de desvíos de archivos. Las operaciones de archivo (a partir de 2,0) se muestran en el listado 4.

**Listado 5**

El campo low\_ino es el número de i-nodo del archivo que se está desregistrando y se ha asignado dinámicamente en tiempo de carga.

* **/proc/root** este está destinado a ser un dispositivo de boqueo. Por lo que debe tener un conjunto de bits S\_IFBLK, por lo que sus operaciones de i-nodo debe ser la que tiene los dispositivos de bloqueo y el número de dispositivos que el de raíz montado actualmente.
* **/proc/insmod** es un archivo grabable. Este necesita su propio archivo de operación para declarar su método de escritura.
* **/proc/jiffies** es más fácil el archivo es solo de lectura. Las versiones del kernel 2,0 y posteriores ofrecen una interfaz simplificada para archivos de sólo lectura.

**Listado 6**

La instantánea de una sesión TTY en el listado 6 muestra cómo aparecen los archivos y cómo funcionan dos de ellos. El listado 7, por último, muestra las tres estructuras utilizadas para declarar las entradas de archivo en/proc.

**Listado 7**

El sistema de archivo "/proc": es muy fácil de ver por lo que no es de muncho rendimiento.

* **El sistema de archivos "UMSDOS**": es parte del núcleo principal y se ejecuta Piggy-Back en el sistema de archivos "ms-dos". Implementa sólo algunas de las operaciones del VFS para agregar nuevas funcionalidades a un formato de sistema de archivos anticuado.
* **El módulo "userfs":** está disponible tanto en el TSX-11 como en el solar bajo Alpha/userfs; la versión 0.9.3 se cargará a Linux 2,0.
* **"supermount":** el sistema de archivos está disponible en el sitio web y en los espejos. Este tipo de sistema de archivos es capaz de montar dispositivos extraíbles como disquetes o CD-ROMs y manejar la eliminación de dispositivos sin forzar al usuario a umount/Mount el dispositivo.
* **"EXT2":** el sistema de archivos Extended-2 ha sido el sistema de archivos Linux estándar desde hace unos años. Es un código difícil, pero vale la pena leer para aquellos interesados en ver cómo se implementa un sistema de archivos real.
* **"ROMFS":** este es el sistema de archivos más pequeño que he visto en mi vida. Se introdujo en Linux-2.1.21. Es un archivo fuente único, y es bastante agradable para navegar. Como su nombre lo afirma, es de sólo lectura.