INSTITUTO TECNOLÓGICO de Pabellón de Arteaga

TEC

TECNM TECNOLOGICO NACIONAL DE MÉXICO

Sistema virtual de archivos en Linux

SISTEMAS OPERATIVOS I

Alumno: Vicente Mata Velasco
Profesor: Eduardo Flores Gallegos

El elemento de datos principal en cualquier sistema tipo Unix es el "archivo", y un nombre de ruta único identifica cada archivo dentro de un sistema en ejecución. Cada archivo aparece como cualquier otro archivo en la forma en que se accede y se modifica: las mismas llamadas al sistema y los mismos comandos de usuario se aplican a cada archivo. Esto se aplica independientemente del medio físico que contiene información y de la forma en que se presenta la información en el medio. La abstracción del almacenamiento físico de la información se logra enviando la transferencia de datos a diferentes controladores de dispositivo. La abstracción del diseño de la información se obtiene en Linux a través de la implementación de VFS.

El camino de Unix

Linux analiza su sistema de archivos, adoptando los conceptos de superbloque, directorio y archivo. El árbol de archivos accesibles en cualquier momento está determinado por la forma en que se ensamblan las diferentes partes, cada parte es una partición del disco duro u otro dispositivo de almacenamiento físico que está "montado" en el sistema.

El **superbloque** debe su nombre a su herencia, desde cuando se utilizó el primer bloque de datos de un disco o partición para contener meta información sobre la partición en sí. El superbloque ahora está separado del concepto de bloque de datos, pero aún contiene información sobre cada sistema de archivos montado.
El kernel 2.0 mantiene una matriz estática de tales estructuras para manejar hasta 64 sistemas de archivos montados.

Un **inodo** está asociado con cada archivo. Tal "nodo índice" contiene toda la información sobre un archivo nombrado, excepto su nombre y sus datos reales. El propietario, el grupo, los permisos y los tiempos de acceso para un archivo se almacenan en su inodo, así como también el tamaño de los datos que contiene, el número de enlaces y otra información.

El **directorio** es un archivo que asocia inodos a nombres de archivos. El kernel no tiene una estructura de datos especial para representar un directorio, que se trata como un archivo normal en la mayoría de las situaciones. Las funciones específicas de cada tipo de sistema de archivos se utilizan para leer y modificar los contenidos de un directorio independientemente del diseño real de sus datos.

El archivo en sí está asociado con un inodo. Por lo general, los **archivos** son áreas de datos, pero también pueden ser directorios, dispositivos, fifos (primeros en entrar, primero en salir) o en los sockets. Un "archivo abierto" se describe en el kernel de Linux por un elemento de archivo struct; la estructura contiene un puntero al inodo que representa el archivo. las estructuras de archivo se crean

mediante llamadas al sistema como abrir, canalizar y socket, y son compartidas por padre e hijo a través del tenedor.

Orientación a Objetos

Un sistema operativo debe ser capaz de manejar diferentes formas de distribución de información en el disco. Aunque teóricamente es posible buscar un diseño óptimo de la información en discos y usarla para cada partición de disco, la mayoría de los usuarios de computadoras necesitan acceder a todos sus discos duros sin formatear, montar volúmenes NFS en la red e incluso acceder a veces esos graciosos CD-ROM y disquetes cuyos nombres de archivo no pueden exceder los 8 + 3 caracteres.

El problema de manejar diferentes formatos de datos de una manera transparente se ha abordado haciendo superbloques, inodos y archivos en "objetos"; un objeto declara un conjunto de operaciones que deben usarse para manejarlo. El kernel no se quedará estancado en las sentencias de conmutación grandes para poder acceder a los diferentes diseños físicos de datos, y los nuevos tipos de sistema de archivos se pueden agregar y eliminar en tiempo de ejecución.

Toda la gestión de datos y el almacenamiento en búfer realizado por el kernel de Linux es independiente del formato real de los datos almacenados. Toda comunicación con el medio de almacenamiento pasa a través de una de las estructuras de operaciones. El tipo de sistema de archivos, entonces, es el módulo de software que se encarga de mapear las operaciones al mecanismo de almacenamiento real, ya sea un dispositivo de bloque, una conexión de red (NFS) o prácticamente cualquier otro medio de almacenamiento y recuperación de datos. Estos módulos pueden estar vinculados al kernel que se está iniciando o compilado como módulos cargables.

La implementación actual de Linux permite el uso de módulos cargables para todos los tipos de sistemas de archivos, pero es root (el sistema de archivos raíz debe montarse antes de cargar un módulo desde él). En realidad, la maquinaria initrd permite cargar un módulo antes de montar el sistema de archivos raíz, pero esta técnica generalmente se explota solo en disquetes de instalación.

struct file_system_type es una estructura que declara solo su propio nombre y una función read_super. En el momento de la instalación, se pasa información a la función sobre el medio de almacenamiento que se está montando y se le pide que complete una estructura de superbloque, así como cargar el nodo del directorio raíz del sistema de archivos como sb-> s_mounted (donde sb es el super -bloqueado solo lleno). El tipo de sistema de archivos utiliza el campo adicional requires_dev para indicar si accederá a un dispositivo de bloque: por ejemplo, los tipos NFS y proc no requieren un dispositivo, mientras que ext2 e iso9660 sí. Una vez que se completa el superbloque, struct file_system_type ya no se usa; solo el

superbloque recién llenado tendrá un puntero al mismo para poder devolver información de estado al usuario (/ proc / mounts es un ejemplo de dicha información). La estructura se muestra en el Listado 1.

Listado 1

El kernel usa la estructura de super_operations para leer y escribir inodos, escribir información de superblock en el disco y recopilar estadísticas (para tratar las llamadas al sistema statfs y fstatfs). Cuando finalmente se desmonta un sistema de archivos, se llama a la operación put_super en la definición estándar del kernel "obtener" significa "asignar y completar", "leer" significa "llenar" y "poner" significa "liberar".

Listado 2

Después de que se haya creado una copia de memoria del inodo, el kernel actuará sobre ella utilizando sus propias operaciones. struct inode_operations es el segundo conjunto de operaciones declarado por los módulos del sistema de archivos, y se enumera a continuación; se ocupan principalmente del árbol de directorios. Las operaciones de manejo de directorios son parte de las operaciones del inodo porque la implementación de una estructura dir_operations traerá condicionales adicionales en el acceso al sistema de archivos. En cambio, las operaciones de inodo que solo tienen sentido para los directorios harán su propia comprobación de errores. El primer campo de las operaciones de inode define las operaciones de archivo para archivos regulares. Si el inodo es un fifo, se usará un socket o una operación de archivo específica del dispositivo.

Listado 3

Las operaciones de archivo, finalmente, especifican cómo se manejan los datos en el archivo real: las operaciones implementan los detalles de bajo nivel de lectura, escritura, lseek y otras llamadas al sistema de manejo de datos. Dado que la misma estructura de file_operations se usa para actuar en dispositivos, también incluye algunos campos que solo tienen sentido para los dispositivos de caracteres o de bloques. Es interesante observar que la estructura que se muestra aquí es la estructura declarada en los núcleos 2.0, mientras que 2.1 modificó los prototipos de lectura, escritura y lseek para permitir una gama más amplia de compensaciones de archivos.