Файловая подсистема Linux.

Лекция 19.02.2022.

Файловая подсистема предназначена для обеспечения возможности хранения и доступа к файлам в системе. Это задача которая стоит перед любой файловой системы любой ОС.

Unix определил подходы к построению тех или иных модулей системы, к структуризации системы в целом.

Определение файла из Оксфордского словаря:

Файл - информация хранимая во вторичной памяти или во вспомог уст с целью ее созранения после заверш отдел задания или преодал огр основного зап устройства (т.н. рабочие файлы). В файле может содержаться любая информация.

Современное определение файла:

Файл - это любая поименованная совокупность данных, которая хранится во вторичной памяти.

Определение файловой системы:

Файловая система - это порядок, определяющий способ организации хранения, именования и доступа к данным на вторичных носителях информации. (Определение состоит из указания задач файловой системы)

Это общие представления о файловой системе.

Рассмотрим обобщенную модель файловой системы. (Из иерархической модели Медника-Донована - на самом высоком уровне файловая система). Аналогично любая файловая система имеет иерархическую структуру. Это связано с разными уровнями этой файловой системы так как различные задачи, которые решает файловая система, выполняются на разных уровнях операционной системы.

Именование файлов (символьный уровень) - это самый высокий уровень файловой системы. Он позволяет пользователю в удобной форме задавать имена файлов и искать их в каталогах. Но файл хранится на физическом уровне (является внешним), значит на нижнем уровне доступ к нему осуществляется с подсистемой ввода-вывода, так как, чтобы считать файл необходимо обратиться к внеш устройству.

[рис 1] (./lec1-1.png)

В системе у файла существует полное имя и короткое имя. Система всегда оперирует полным именем. В различных системах имеются отличия в представлении этого имени (в Windows оно начинается с имени логического диска, в Linux - с корневого каталога).

В Unix имя файла не является его идентификатором. В системе файл идентифицируется номером inode'a. Inode - (фактически) дескриптор файла. Первые два уровня определяют именование файла в

системе (см рис 1).

Затем модуль проверки прав доступа. (Важнейшая задача ос - контроль прав доступа read/write/execute).

Логический уровень. В данном случае файл похож на программу. Любая программа считает, что она начинается с нулевого адреса. В программе наход смещение. Логическая организация файла начинается с нуля.

Физический уровень. Файл хранится на внешнем устройстве - это уже подсистема ввода-вывода - это уже прерывания. На этом уровне осуществляется учет особенностей организации внешнего устройства.

Дисковые устройства на сегодняшний день являются единственными блочными устройствами (и флэш-память). Все остальные устройства - символьные.

Это самое общее представление уровней файловой системы.

В UNIX Linux имеется существ отличия в системе для работы с файлами.

В UNIX для работы с файлами организовано через интерфейс, который называется VFS/vnode (Virtual file system/virtual node).

В Linux VFS не определена структура vnode. Это сделано для того, чтобы обеспечить широкую поддержку различных файловых систем без перекомпиляции ядра.

[рис 2] (./lec1-2.png)

[рис 3] (./lec1-3.png)

Характерное в UNIX/Linux дерево каталогов.

[рис 4] (./lec1-4.png)

Для Linux.

Важными являются аббревиатуры GNU C и SCI (System call interface / Standard system call interface).

Магнитные диски (рис 5)

Современные файловыми системами поддерживаются так называемые очень большие файлы. Проводя аналогию с управлением памятью (таблицами страниц), обеспечивается хранение файлов в разброс, они не занимают непрерывные адреса в памяти. Обеспечивается возможность хранения информации из одного и того же файла и это обеспечивается хранением соответствующей информации, а именно адресов блоков.

Такая возможность в ОС юникс линукс реализуется за счет того, что в состав системы входит "слой абстракции" над собственным низкоуровневым интерфейсом файловой системы. Для этого VFS

предоставляет общую файловую модель, которая способна отображать общие возможности и "поведение" любой возможной файловой системы. Такой уровень абстракции работает на основе базовых концептуальных интерфейсов и структур данных, которые поддерживаются конкретными файловыми системами. Фактический код любой файловой системы скрывает детали реализации непосредственной работы с данными, организованными в файлы. А именно, предоставл в распоряж польз как правило набор API, таких как открыть файл, прочитать файл, записать в файл, удалить файл, переименовать файл и т.д. Любая файловая система поддерживает такие понятия как файл, каталог и опять же, действия опред над файлами (уже перечисленные). Это можно представить себе следующей абстракцией. (рис 6)

VFS предоставляет общую файловую модель, которую наследуют низлежащие файловые системы, реализуя действия для различных Posix API.

Системный вызов write() сначала обрабатывается общим системным вызовом sys_write(), который определяет фактический способ записи, характерный той фс, на которой находится файл. Затем общий системный вызов sys_write() вызывает метод конкретной фс для того, чтобы выполнить запись данных на физический носитель. То есть, VFS (как любая фс) скрывает особенности работы с конкретным физическим устройством.

Организация VFS в Linux

Внутренняя организация VFS построена на 4-х структурах:

1. Суперблок (superblock)
2. Индексный узел (inode)
3. Запись каталога (dentry)
4. Файл (file)

Есть диаграмма связи этих структур (рис 7).

Система различает файл находящийся на диске (его описывает inode) и открытый файл (его описывает struct file) это и показано на рисунке. Показана связь всех 4 структур.

Показано, что 2 процесса открыли файл (один и тот же с разными именами). У него (файла) 1 inode. Очевидно, чтобы обесп доступ к этому файлу в системе имеется struct dentry (сокращение от directory entry) -- обеспечивает работу с каталогами.

Структура superblock в VFS описывает конкретную подмонтированную файловую систему. Т.е. файловую систему, которая находится во внешней памяти, которая располагается на внешнем носителе (подмонтированная).

f_dentry, dinode, i_sb - это не идентификаторы, это имена полей соответствующих структур. В struct file есть поле dentry, которая ссылается на структуру, описывающую соответствующий каталог и т.д.

Любой файл хранится в (=принадлежит) конкретной (рабочей, т.е. определенной на диске) файловой системе.

Суперблок

Суперблок - это контейнер для высокоуровневых метаданных о файловой системе.

Суперблок - структура, которая должна находится на диске, тк эта стр опис фс. Для надежности он хранится в нескольких местах диска. В этой структуре хранятся управляющ параметры фс, такие как суммарное число блоков, свободное число блоков, корневой inode и т.д.

В системе сущ суперблок на диске и суперблок в оперативной памяти. Суперблок на диске хранит информацию необходимую системе для доступа к физическому файлу, т.е. хранит информацию о структуре файловой системе. Суперблок в памяти предоставляет информацию, необходимую для управления смонтированной файловой системой. При этом в системе имеется связный список - struct list_head superblocks (хранит информацию обо всех подмонтированных файловых системах). Эта структура определена в `<linux/fs.h>`.

Монтирование

В дерево файлов и катагов входят отдельн ветви, которые формируются разными фс. Для Юних определено подключение различных файловых систем в дерево каталогов и это действие получило название монтирование.

монтирование - система действий, в результате которых файловая система устройства становится доступной т.е. можно получить доступ к информации, которая хранится на устройстве.

Базовая форма команды `mount` принимает на вход 2 параметра: имя устройства (или какого-то другого ресурса, которая содержит монтируемую фс) и точку монтирования. Могут присутствовать ключи - тип файловой системы, опции файловой системы.

```
mount ключи -t тип_фс -o опции_фс устройство
каталог(=точка_монтирования)
```