

Файловая подсистема.
С ней начали писать UNIX (показатель);
(важность);

Файл - важнейшее понятие в файловой подсистеме;

Файл - информация, хранимая во вто-
(из Оксфорд-
Ского словаря) ричной памяти или во вто-
ричном ЗУ с целью её сохранения
после завершения отдельного задания или
преодоления ограничений, связанных с объёмом
основного ЗУ;

Задача ФС - обеспечивать сохранение
данных и доступ к сохранённым данным.

Файл - помеченная сов-ть данных,
хранимая во вторичной памяти
(возможно даже целая);

Доступ к файлу осущ. по его имени;

Чтобы обеспечить хранение файла и последующий
доступ к нему, Ф.С. должна обеспечить изоли-
рованность файлов;

Файл должен быть изолирован, т.е.
занимать некоторое адр. пр-во, и это
адр. пр-во должно быть защищено;

Более того, необходимо обеспечить доступ
к файлу, и он обеспечивается по тому, как
файл идентифицируется в системе;

Файловая система — порядок, определяющий способ организации хранения, именования и доступа к данным на вторичных носителях информации;

Вторичная память
(стор.)

Дисковая память

долговременное
хранение
данных

Вся жизнь наша построена на кассете, и это кассеты мы должны понимать

В наших компьютерах все 3У — блочные;

В системе 2 типа устройств:
символьные и блочные;

Цитата из Оксфордского словаря:

File management (управление файлами) — программные процессы, связанные с общими управлением файлами, т.е. с размещением во вторичной памяти, контролем доступа к файлам, записью резервных копий, ведением справочников (directory);

Осн. ф-ции управ-я файлами обычно выполняются ОС, а дополнительные — системами управления файлами;

Доступ к файлам: open, read, write, rename, delete, remove и т.д.

В UNIX всё файл. Если что-то не файл, то это процесс

→ Справедливо, т.к. в системе имеются спец. файлы, про которые говорят, что они больше, чем файлы: программные каналы, сокеты, внеш. устр-ва;

Файловая система работает с регулярными (обычными) файлами и директориями;

При этом UNIX и Linux не делают различий между файлами и директориями (каталогами);

Директория - файл, который содержит имена других файлов;

7 типов файлов в UNIX:

- "-" - обычный файл (regular);
- "d" - directory;
- "l" - soft link;
- "c" - special character device;
- "b" - block device;
- "s" - socket;
- "p" - named pipe;

Стандарт File System Hierarchy Standard (FHS), который определяет структуру и содержание каталогов в Linux distribution (Ubuntu поддерживает этот стандарт)

По этому стандарту корень ф.с. обозначается как "/" (корневой каталог) и его ветви обязательно должны составлять единую ф.с., расположенную на одном носителе (диске или дисковом разделе). В ней должны располагаться все компоненты, необходимые для старта системы;

Другое определ-е файла:

Файл - каждая индивидуально идентифицированная единица информации;

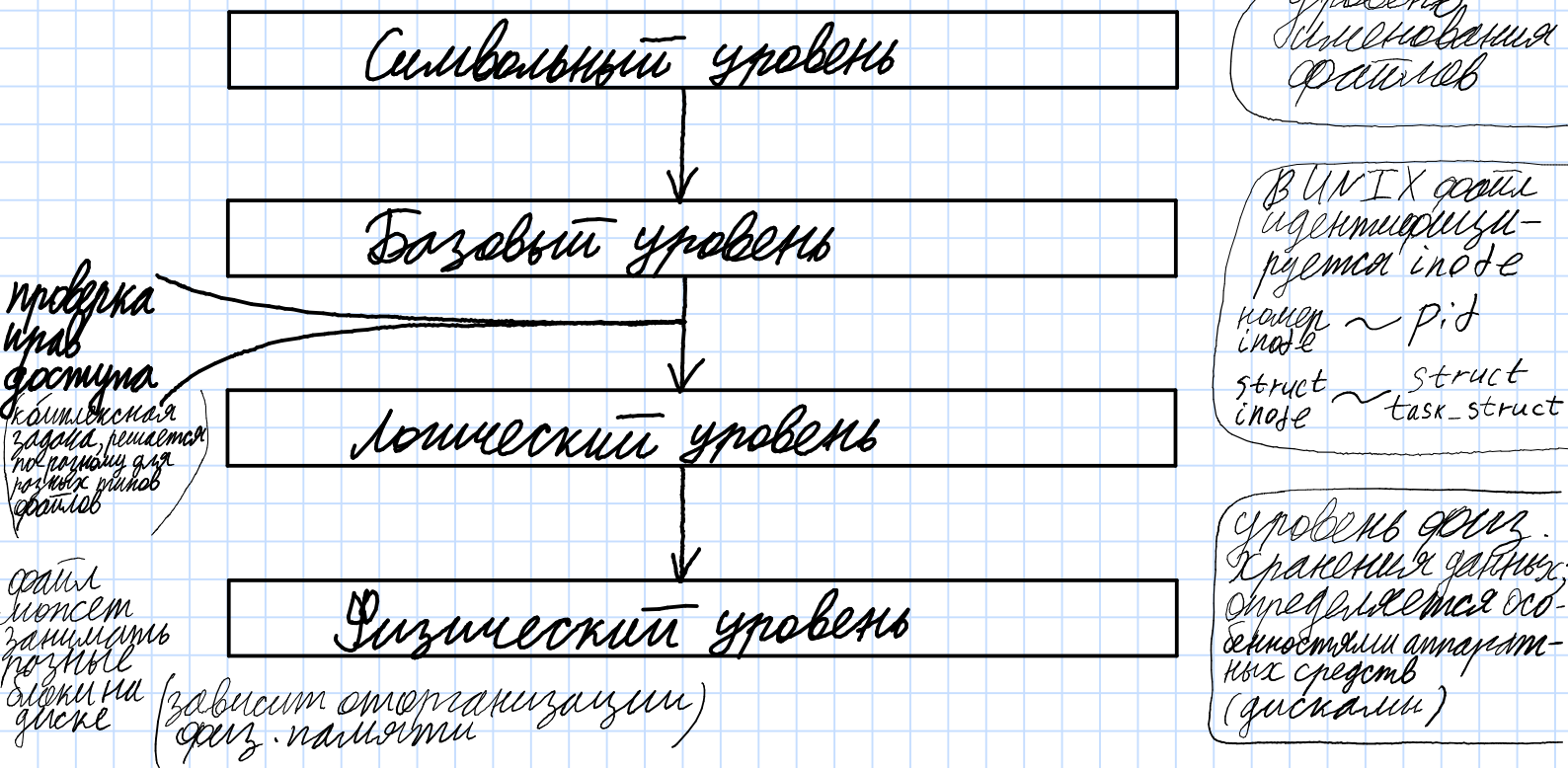
Задачи Ф.С.:

- 1) Именованье файлов;
- 2) Обеспечение программного интерфейса для работы с файлами пользо-вателей или с приложениями;

Ни ОС, ни ФС никак не интерпретирует содержимое файла;

- 3) Обратное логическое моделирование (логического представления файлов на физическую организацию хранения данных на соотв. носителе);
 - 4) Обеспечивает надёжное хранение файлов, доступ к ним и защиту от несанкционированного доступа (права доступа в Linux);
 - 5) Обеспечение совм. использования файлов;
- (и.б. не в той мере задача ф.с., но задача ОС)

Иерархическая структура ФС



В UNIX/Linux файл может иметь много имен (hard link);

В системе поддерживаются байт-ориентированные (direct-oriented) и блок-ориентированные (block-oriented) файлы;

Символьные файлы ~ байт-ориентированные файлы; (1 символ определяется 1 байтом)

Блочные файлы ~ блок-ориентированные файлы;

Уровень иерархии памяти определяется близостью к процессору;

Файл \neq место на диске

В мире совр. вычисл. техники файлы имеют настолько большие размеры, что не могут храниться

наться в неупр. физ. адр. пр-ве, они хранятся
вразброс;

Файл может занимать разные блоки/секто-
ра/дорожки на диске аналогично тому,
как память поделена на страницы; в лю-
бой момент м.б. загружена новая страница
(как и файл);

Также важно понимать, как это всё адресу-
ется;

Для файлов характерно несвязанное распреде-
ление: файлу выделяются участки памяти
вразброс;

Соответственно, система должна обеспе-
чить адресацию каждого такого участка;

Файловая система Linux.

Отличается от Ф.С. UNIX представ-
ляемым интерфейсом: ОС UNIX и Linux
позволяют работать с файлами как с Ф.С.;

Для этого UNIX предоставляет интерфейс
VFS/vnode, а в Linux — VFS (от vnode);

↳ virtual

и там, и там файл идентифицируется
номером inode

/root

/bin /boot /etc /usr /var /dev ... /home

„Родные” ФС для Linux — ext (ext2, ext4, ...)

ext2, ext4, XFS, UFS, NTFS, MS-DOS, FAT, FAT32, MPFS, NFS;

Фактически VFS — интерфейс, с помощью которого ОС может работать с самыми разными файловыми системами;

Основой такой работы (базовым действием) явл. монтирование: прежде чем ф.с. станет доступна (мы сможем увидеть её каталоги и файлы) она должна быть смонтирована;

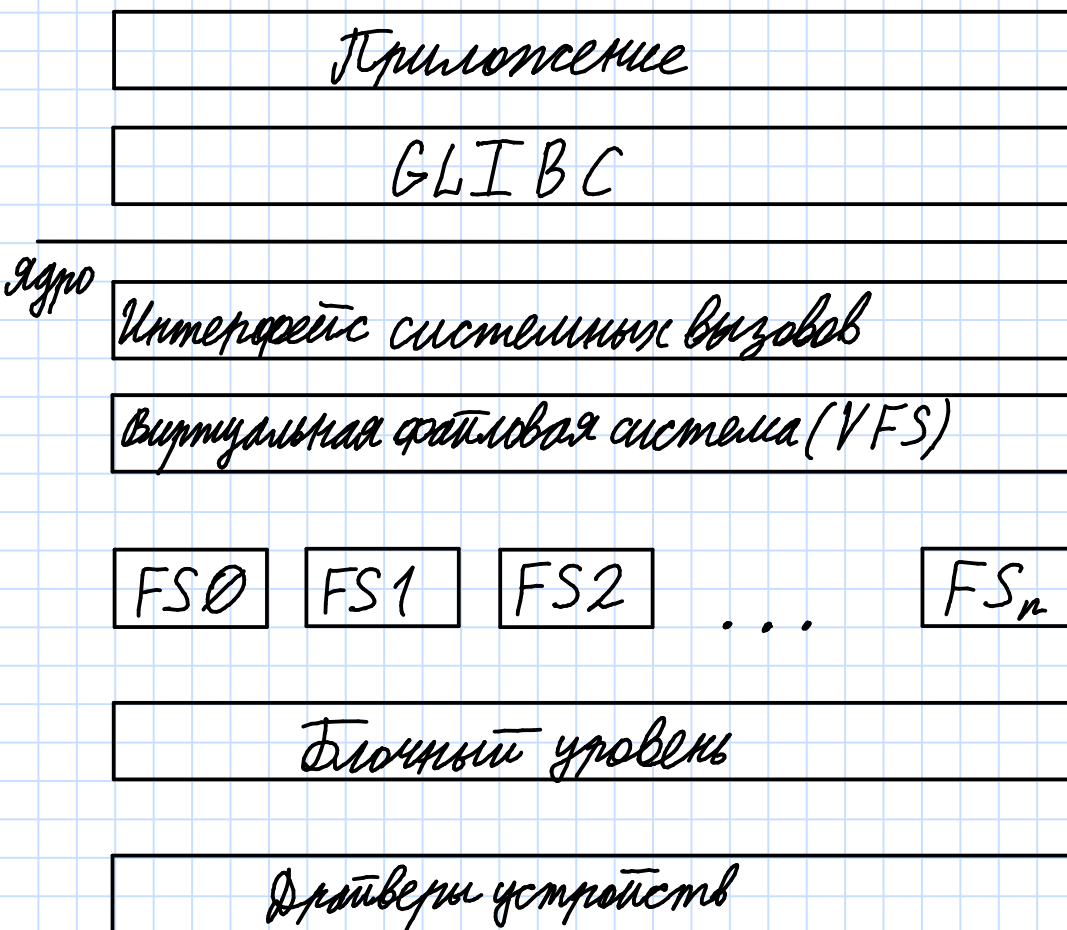
Команда mount;

DOS (и впоследствии Windows) освободил пользователя от этих действий;

Это обеспечило её популярность;

Спустя много десятилетий в Windows это исправили: когда мы подключаем, например, флешку, она монтируется автоматически (даже несмотря на то, что на ней м.б. другая ф.с.);

Структура слоев VFS:



(GNU C)
(glibc)

SCI (standard kernel system-call interface)

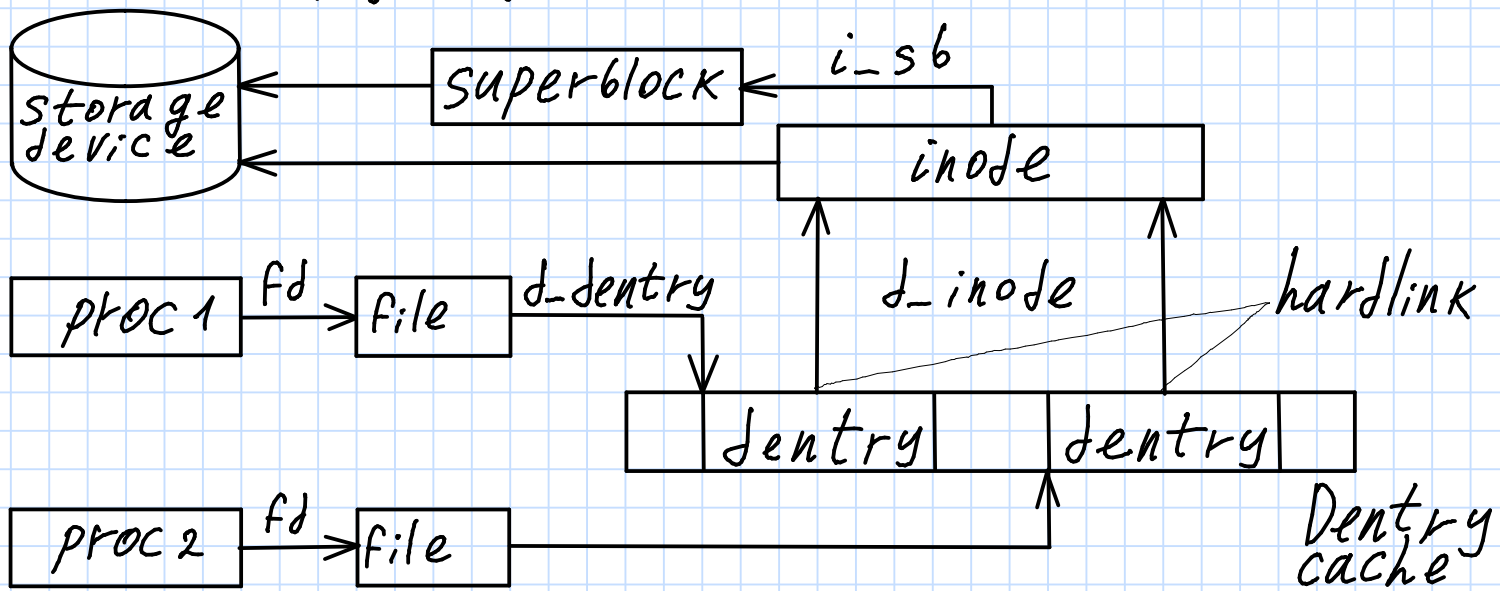
Все операции с файлами на подмонтированных ф.с. осуществляются через интерфейс VFS;

Внутренняя организация VFS базируется на 4 структурах

Struct

- 1) superblock;
- 2) dentry;
- 3) inode;
- 4) file;

Связь структур:



Все эти структуры взаимодействуют между собой, для каждого файла создаются их экземпляры (наз. объектами, хотя ядро UNIX и Linux написано на языке C без использования объектно-ориентированного подхода);

- 1) struct superblock описывает подмонтированную ф.с. на диске;
Именно он обеспечивает возможность работы с ф.с.;
Содержит информацию для обеспечения доступа к файлам, которые хранятся на дисках вразброс, т.е. адресацию соотв. участков диска;
- 2) struct dentry (Directory entry) описывает экземпляр директории;
Нигде не хранится, создается "на лету" на основе информации, которая хранится в директориях на диске;

3) struct inode - дескриптор файла;

Существует 2 типа inode:

- 1) Дисктовый inode описывает физический файл;
- 2) inode в ядре („ядреный“ inode), который позволяет предварительно контролировать доступ к файлу;

regular file → обращение к дисковому inode

pipe, socket и др. спец. файлы должны существовать не в superblock (доступ к ним должен осуществляться не через superblock);

Существует 2 инстансы файла:

- 1) файл, который лежит на диске;
- 2) открытый файл (с которым работает процесс);

4) struct file описывает открытый файл;

В ядре имеется сист. табл. открытых файлов.

Каждый процесс имеет собственную таблицу открытых файлов, дескрипторы которой ссылаются на дескрипторы в таблице открытых файлов;

Каждый процесс до того, как он был запущен, был файлом и принадлежал некоторой ф.с., поэтому в `struct task_struct` имеется указатель на ф.с., которой принадлежит файл программы, и указатель на таблицу открытых файлов процесса;

Стрелки на схеме = „имеет указатель на“;

Любой файл открывается с помощью процесса (примонсениа);

{ struct superblock рассматривается для ext 2 }

Вся ф.с. должна занимать либо диск, либо раздел диска и начинаться с корневого каталога;

Любая ф.с. монтируется к общему дереву каталогов (монтируется в поддиректорию);

И эта подмонтированная ф.с. описывается суперблоком и должна занимать некоторый раздел жесткого диска („это делается в процессе монтирования“)

Разделы жесткого диска ф.с. ext 2:
... не стали рисовать ...