КОНСТРУИРОВАНИЕ ПРОГРАММ

Лекция № 08 — Файловая система (кратко)

Преподаватель: Поденок Леонид Петрович, 505а-5

+375 17 293 8039 (505а-5)

+375 17 320 7402 (ОИПИ НАНБ)

prep@lsi.bas-net.by

ftp://student:2ok*uK2@Rwox@lsi.bas-net.by/

Кафедра ЭВМ, 2021

2021.10.13

Оглавление

Общие понятия....3
Файловая система ОС UNIX....5
Ограничения на имя файла....7
Типы файлов. Символические ссылки....9
Права доступа к файлам....11
Базовые биты прав доступа....12
Структура файловой системы....15
Физическая файловая система UNIX....16
Суперблок....17
Индексные дескрипторы....19
Файлы устройств....21
Два типа устройств....22
Операции над устройствами....23

Общие понятия

Файл (Именованный файл) — некий набор хранимых на внешнем носителе данных, имеющий имя (набор данных — data set).

Это не строгое определение файла — оно лишь отражает те свойства, которые мы обычно используем на прикладном уровне.

Термин «файловая система» имеет 2 значения:

1) подсистема ОС, отвечающая за хранение информации на внешних запоминающих устройствах (ЗУ) в виде именованных файлов;

2) обозначение структур данных, создаваемых на внешнем ЗУ с целью организации хранения на этом устройстве данных в виде именованных файлов.

Имя файла — определяется алфавитом и длиной.

Каталог (Directiry) — особый тип файла, хранящий имена файлов, некоторые из которых, возможно, сами являются каталогами.

При создании на диске файловой системы создается один каталог — **корневой каталог**.

При необходимости в корневом каталоге можно создавать другие каталоги, а их имена записать в корневой — **каталоги первого уровня вложености**.

В них, в свою очередь, тоже можно создавать каталоги.

При использовании каталогов говорят о **кратком** или **локальном** имени файла (строка символов, уникальная в пределах каталога) и о **полном имени файла** или **полном пути к файлу** (строка символов, включающая имена всех каталогов, начиная с корневого и заканчивая локальным).

Имена каталогов разделяются символом «**/**».
(*URI — Uniform Resource Identificator*)

В каждом каталоге существует файл со специальным именем, обозначающим каталог более высокого уровня (родительский). Его имя «**..**».
Еще одно специальное имя — файл, описывающий сам каталог. Его имя «**.**».

При работе с файлами один из каталогов тем или иным способом объявляется **текущим (current directory)**.
К файлам из текущего каталога можно обращаться по короткому имени.
Для обращения к файлам вне текущего каталога можно использовать **полный (абсолютный)** или **относительный** путь.
Относительный путь начинается с текущего каталога.
Операционные системы различают полные и относительные пути по наличию символа-разделителя «**/**» перед первым именем в пути.

```
/home/user/music/ACDC/Back in Black/Back in Black.flac
music/ACDC/Back in Black/Back in Black.flac
```

Файловая система ОС UNIX

Единое дерево каталогов.

В имя файла ни в каком виде не входит имя устройства, на котором файл находится.

Если в системе присутствует несколько устройств прямого доступа (dasd), один их них объявляется **корневым (root)**, и все остальные **«монтируются» (mount)** в тот или иной каталог, называемый точкой **монтирования (mount point)**.

Для указания полного пути к файлу на примонтированном устройстве спереди к имени файла добавляется полный путь к точке монтирования. Например, если у нас есть дискета, смонтированная в **`/media/myfloppy`**, а на ней есть каталог **`work`**, а в нем файл fo**`o.c`**, то полный путь к этому файлу будет **`/media/myfloppy/work/foo.c`**.

В ОС UNIX каталоги хранят только имя файла и некоторый номер, позволяющий идентифицировать соответствующий файл.

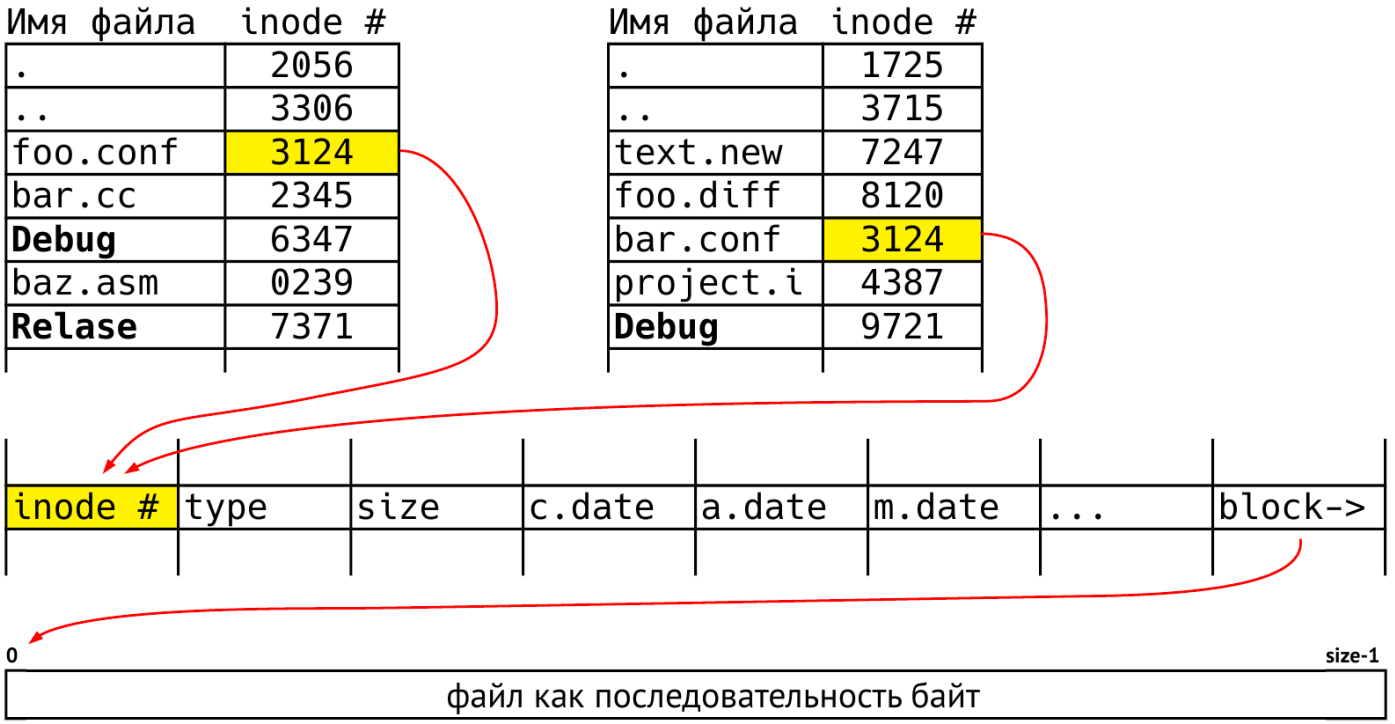
Вся остальная информация о файле, как то размер, расположение на диске, даты создания, модификации и последнего обращения, данные о владельце и о правах доступа к нему связываются не с именем файла, а с этим самым номером, который связан индексным дескриптором файла.

ИНДЕКСНЫЙ ДЕСКРИПТОР — хранимая на внешнем ЗУ (диске) структура данных, содержащая всю информацию о файле, исключая его имя.

Размер индексного дескриптора обычно составляет 128 байтов.

Допускается, чтобы несколько имен файлов, в том числе и в разных каталогах, ссылались на один и тот же номер индексного дескриптора.

Индексный дескриптор содержит:
- номер (уникальный в рамках файловой системы данного диска);
- тип файла;
- права доступа к файлу;
- количество связей (ссылок на файл в каталогах) файла;
- идентификатор пользователя и группы-владельца;
- размер файла в байтах;
- время последнего доступа к файлу;
- время последнего изменения файла;
- время последнего изменения индексного дескриптора файла;
- указатели на блоки данных файла (обычно 10);
- указатели на косвенные блоки (обычно 3).



Ограничения на имя файла

В имени файла допустимы любые символы ASCII за исключением разделителя каталогов «/» и нулевого символа '`\0`'.

Ограничения на длину имени существуют — обычно 255.

Современные ФС допускают использование UNICODE.

Единственная кодировка UNICODE, совместимая с ASCII — utf8.

Стандарт POSIX **(Portable Operating System Interface)** требует реализации поддержки двух типов связей имен с индексными дескрипторами

- жестких;

- символических.

Жесткой ссылкой (hard link) считается элемент каталога, указывающий непосредственно на некоторый индексный дескриптор.

Жесткие связи очень эффективны, но у них существуют определенные ограничения, так как они могут создаваться только в пределах одной физической файловой системы.

Когда создается такая связь, связываемый файл должен уже существовать.

Кроме того, каталоги не могут связываться жесткой связью.

Фактически, файл может иметь несколько совершенно равноправных имен.

Файл создается (**`open()`**, **`create()`**) всегда с одним именем. Дополнительные имена можно назначить файлу используя системный вызов **`link()`**. Этот вызов проецируется на команду оболочки **`ln`**.

В индексном дескрипторе содержится счетчик жестких ссылок на дескриптор.

Создание жестких ссылок на каталоги не допускается (на уровне ядра) из-за вероятности возникновения рекурсии и зацикливания.

Создание жестких ссылок на файлы, расположенные на других устройствах, не допускается (уникальность номеров индексных дескрипторов имеет место только в пределах устройства).

Файл удаляется функцией **`unlink()`**, которая удаляет одну из жестких ссылок. Когда будет удалена последняя, файл считается удаленным и ресурсы, которые он занимает на носителе освобождаются.

Типы файлов. Символические ссылки

Каталоги — это файлы специального типа. Они хранят имена и номера индексных дескрипторов. Все, чем отличается каталог от обычного файла на низком уровне — это значение признака типа в индексном дескрипторе.

В остальном хранение каталогов на диске не отличается от хранения обычных файлов.

Кроме обычных файлов и каталогов операционные системы поддерживают и другие специальные типы файлов. FAT (MS DOS, Windows), например, поддерживает тип «метка тома» (volume label).

UNIX поддерживает достаточно большое количество разновидностей файлов специального типа:

- файлы байт-ориентированных устройств;
- файлы блок-ориентированных устройств;
- имена сокетов;
- именованные каналы (FIFO);
- символические ссылки.

Файловая система обеспечивает перенаправление запросов, адресованных периферийным устройствам, к соответствующим модулям подсистемы ввода-вывода.

Символическая ссылка (Symbolic link) — файл специального типа, содержащий путь к другому файлу.

Указание на то, что данный элемент каталога является символической ссылкой, находится в индексном дескрипторе. О*бычные* команды доступа к файлу вместо получения данных из физического файла, берут их из файла, имя которого приведено в ссылке.

Этот путь может указывать на что угодно — это может быть каталог, он может даже находиться в другой физической файловой системе, более того, указанного файла может и вовсе не быть.

В описании пути можно использовать любое сочетание символических и жестких ссылок.

Операция открытия символической ссылки на чтение или запись приводит к открытию на чтение или запись того файла, на который она ссылается, а не ее самой.

Символическая ссылка в отличие от жесткой имеет свой собственный номер индексного дескриптора и имеет свой тип.

Создание и/или удаление символической ссылки никогда не затрагивают ни имени файла, на который она ссылается, ни его индексного дескриптора.

Файл может быть удален, а ссылка остается («висящая»).

Файл может не существовать в момент создания символической ссылки.

Права доступа к файлам

Файловая система контролирует права доступа к файлам, выполняет операции создания и удаления файлов, а также выполняет запись/чтение данных файла.

Каждый файл в ОС UNIX содержит набор прав доступа, по которому определяется, как пользователь взаимодействует с данным файлом. Этот набор хранится в индексном дескрипторе данного файла в виде 16-разрядного целого значения, из которого обычно используется 12 битов. Каждый бит используется как переключатель, разрешая (значение 1) или запрещая (значение 0) тот или иной доступ.

Три первых бита устанавливают различные виды поведения при выполнении.

Оставшиеся девять делятся на три группы по три, определяя права доступа для владельца, группы и остальных пользователей. Каждая группа задает права на чтение, запись и выполнение.

Базовые биты прав доступа

Восьмеричное значение	Вид в столбце прав доступа	Право или назначение бита
4000	`---s------`	Установленный эффективный идентификатор владельца (бит SUID)
2000	`------s---`	Установленный эффективный идентификатор группы (бит SGID)
1000	`---------t` `---------T`	*Клейкий* (sticky) бит. Вид для каталогов и выполняемых файлов, соответственно.
0400	`-r--------`	Право владельца на чтение
0200	`--w-------`	Право владельца на запись
0100	`---x------`	Право владельца на выполнение
0040	`----r-----`	Право группы на чтение
0020	`-----w----`	Право группы на запись
0010	`------x---`	Право группы на выполнение
0004	`-------r--`	Право всех прочих на чтение
0002	`--------w-`	Право всех прочих на запись
0001	`---------x`	Право всех прочих на выполнение

```
$ ls -l /etc
итого 2608
drwxr-xr-x. 3 root root   4096 дек 17 16:35 abrt
-rw-r--r--. 1 root root     18 фев 21  2020 adjtime
-rw-r--r--. 1 root root   1529 окт  9  2019 aliases

lrwxrwxrwx. 1 root root   30 сен 27  2019 extlinux.conf -> ../boot/extlinux/extlinux.conf
lrwxrwxrwx.  1 root root  21 ноя  4 22:13 os-release -> ../usr/lib/os-release
```

Бит чтения для всех типов файлов имеет одно и то же значение: он позволяет читать содержимое файла (получать листинг каталога командой **ls**).

Бит записи также имеет одно и то же значение — он позволяет писать в этот файл, включая и перезапись содержимого. Если у пользователя отсутствует право доступа на запись в каталоге, где находится данный файл, то пользователь не сможет его удалить. Аналогично, без этого же права пользователь не создаст новый файл в каталоге, хотя может сократить длину доступного на запись файла до нуля.

Бит выполнения. Если для некоторого файла установлен бит выполнения, то файл может выполняться как команда. В случае установки этого бита для каталога, этот каталог можно сделать текущим (перейти в него командой **cd**).

Бит SUID. Установленный бит SUID означает, что доступный пользователю на выполнение файл будет выполняться с правами (с эффективным идентификатором) владельца, а не пользователя, вызвавшего файл (как это обычно происходит).

Бит SGID. Установленный бит SGID означает, что доступный пользователю на выполнение файл будет выполняться с правами (с эффективным идентификатором) группы-владельца, а не пользователя, вызвавшего файл (как это обычно происходит).

Если бит SGID установлен для файла, не доступного для выполнения, он означает обязательное блокирование, т.е. неизменность прав доступа на чтение и запись пока файл открыт определенной программой.

Клейкий/липкий бит. Установленный клейкий бит для обычных файлов ранее (во времена PDP-11) означал необходимость сохранить образ программы в памяти после выполнения (для ускорения повторной загрузки). Сейчас при установке обычным пользователем он сбрасывается.

Значение этого бита при установке пользователем **root** зависит от версии ОС и иногда необходимо. Так, в ОС Solaris необходимо устанавливать клейкий бит для обычных файлов, используемых в качестве области подкачки.

Установка клейкого бита для каталога означает, что файл в этом каталоге может быть удален или переименован только пользователем-владельцем файла, пользователем-владельцем каталога, если файл доступен пользователю на запись и пользователем root.

Расчет прав. Для расчета прав доступа необходимо сложить восьмеричные значения всех необходимых установленных битов. В результате получится четырехзначное восьмеричное число. Например:

Чтение для владельца:	0400
Запись для владельца:	0200
Выполнение для владельца:	0100
Чтение для группы:	0040
Выполнение для группы:	0010
Выполнение для прочих:	0001
Сумма:	0751

Структура файловой системы

Каждый жесткий диск состоит из одной или нескольких логических частей (групп цилиндров), называемых разделами (partitions). Расположение и размер раздела определяется при форматировании диска.

В ОС UNIX разделы выступают в качестве независимых устройств, доступ к которым осуществляется как к различным носителям данных. Обычно в разделе может располагаться только одна физическая файловая система.

Имеется много типов физических файловых систем, например FAT16, VFAT, NTFS, HPFS, HFS с разной структурой.

Имеется множество типов физических файловых систем UNIX (ufs, s5fs, ext2, vxfs, jfs, ffs и т.д.).

Физическая файловая система UNIX

Физическая файловая система UNIX занимает раздел диска и состоит из следующих основных компонентов:

Суперблок (superblock).
Содержит общую информацию о файловой системе.

Массив индексных дескрипторов (ilist).
Содержит метаданные всех файлов файловой системы.
Индексный дескриптор (***inode***) содержит информацию о статусе файла и указывает на расположение данных этого файла. Ядро обращается к индексному дескриптору по индексу в массиве.

Один дескриптор является корневым для физической файловой системы, через него обеспечивается доступ к структуре каталогов и файлов после монтирования файловой системы.
Размер массива индексных дескрипторов является фиксированным и задается при создании физической файловой системы.

Блоки хранения данных.
Данные обычных файлов и каталогов хранятся в блоках. Обработка файла осуществляется через индексный дескриптор, содержащий ссылки на блоки данных.

Суперблок

Суперблок содержит информацию, необходимую для монтирования и управления файловой системой в целом. В каждой файловой системе существует только один суперблок, который располагается в начале раздела. Суперблок считывается в память ядра при монтировании файловой системы и находиться там до ее отключения - демонтирования.

Суперблок содержит:

- тип файловой системы;
- размер файловой системы в логических блоках, включая сам суперблок, массив индексных дескрипторов и блоки хранения данных;
- размер массива индексных дескрипторов;
- количество свободных блоков;
- количество свободных индексных дескрипторов;
- флаги;
- размер логического блока файловой системы (512, 1024, 2048, 4096, 8192).
- список номеров свободных индексных дескрипторов;
- список адресов свободных блоков.

Количество свободных индексных дескрипторов и блоков хранения данных может быть значительным, поэтому хранение списка номеров свободных индексных дескрипторов и списка адресов свободных блоков целиком в суперблоке непрактично.

Для индексных дескрипторов храниться только часть списка. Когда число свободных дескрипторов приближается к 0, ядро просматривает список и вновь формирует список свободных дескрипторов.

Такой подход неприемлем в отношении свободных блоков хранения данных, поскольку по содержимому блока нельзя определить, свободен он или нет. Поэтому необходимо хранить список адресов свободных блоков целиком.

(§) Список адресов свободных блоков может занимать несколько блоков хранения данных, но суперблок содержит только один блок этого списка.

Первый элемент этого блока указывает на блок, хранящий продолжение списка.

Выделение свободных блоков для размещения файла производиться с конца списка суперблока. Когда в списке остается единственный элемент, ядро интерпретирует его как указатель на блок, содержащий продолжение списка. В этом случае содержимое этого блока считывается в суперблок, и блок становится свободным. Такой подход позволяет использовать дисковое пространство под списки, пропорциональное свободному месту в файловой системе. Когда свободного места практически не остается, список адресов свободных блоков целиком помещается в суперблоке.

Индексные дескрипторы

Индексный дескриптор, или **inode**, содержит информацию о файле, необходимую для обработки данных, т.е. метаданные файла. Каждый файл ассоциирован с одним индексным дескриптором, хотя может иметь несколько имен (жестких ссылок) в файловой системе, каждое из которых указывает на один и тот же индексный дескриптор.

Индексный дескриптор не содержит:

- имени файла, которое содержится в блоках хранения данных каталога;
- содержимого файла, которое размещено в блоках хранения данных.

Индексный дескриптор содержит:

- номер;
- тип файла;
- права доступа к файлу;
- количество связей (ссылок на файл в каталогах) файла;
- идентификатор пользователя и группы-владельца;
- размер файла в байтах;
- время последнего доступа к файлу;
- время последнего изменения файла;
- время последнего изменения индексного дескриптора файла;
- указатели на блоки данных файла (обычно 10);
- указатели на косвенные блоки (обычно 3).

Размер индексного дескриптора обычно составляет 128 байтов.

Индексный дескриптор содержит информацию о расположении данных файла. Поскольку дисковые блоки хранения данных, в общем случае, располагаются не последовательно, индексный дескриптор должен хранить физические адреса всех блоков, принадлежащих данному файлу.

Каждый дескриптор содержит 13 указателей.

Первые 10 указателей непосредственно ссылаются на блоки данных файла.

Если файл большего размера — 11-ый указатель ссылается на первый косвенный блок (indirection block) из 128 (256) ссылок на блоки данных.

Если и этого недостаточно, 12-ый указатель ссылается на дважды косвенный блок, содержащий 128 (256) ссылок на косвенные блоки.

Наконец последний, 13-ый указатель ссылается на трижды косвенный блок из 128 (256) ссылок на дважды косвенные блоки. Количество элементов в косвенном блоке зависит от его размера.

Поддерживая множественные уровни косвенности, индексные дескрипторы позволяют отслеживать огромные файлы, не растрачивая дисковое пространство для небольших файлов.

Файлы устройств

ОС UNIX обобщает концепцию файла как универсальную абстракцию.

Практически любое внешнее устройство может быть представлено на пользовательском уровне, как файл специального типа. Это справедливо для жестких дисков, всевозможных параллельных и последовательных портов и виртуальных терминалов.

Например, для создания образов CD/DVD, резервных копий дисков и дисковых разделов в ОС Windows необходимо использовать специальное ПО.

В ОС UNIX этот вопрос решается крайне просто.

```
$ cat /dev/cdrom > image.iso
$ cat myfile.ps > /dev/lp0
$ dd if=/dev/sdb of=sdb.backup
$ dd if=sdb.backup of=/dev/sdb
```

Работа с такими устройствами осуществляется с помощью тех же системных вызовов, которые используются для работы с файлами.

Два типа устройств

Устройства, которые могут быть представлены в виде файлов, делятся на два типа — **символьные** (или потоковые) и **блочные**. Иногда используются термины «байт-ориентированные» и «блок-ориентированные» устройства.

Основными операциями над байт-ориентированными устройствами являются запись и чтение одного (очередного) символа (байта).

Блок-ориентированные устройства воспринимаются как хранилища данных, разделенных на блоки фиксированного размера. Основными операциями, соответственно, являются чтение и запись заданного блока.

Байт-ориентированные — терминал (`/dev/tty0`, `/dev/tty1`, `...`), клавиатура, мышь, принтер (`/dev/lp0`, `...`), звуковая карта, `/dev/null`, `/dev/zero`, `/dev/random`, ...

Блок-ориентированные — диски (/dev/sda,/dev/sdb,...), разделы дисков (/dev/sda1, /dev/sda2,...).

Операции над устройствами

Файлы устройств могут быть открыты на чтение и запись. В результате, используя полученный дескриптор, можно писать на устройство и читать с него.

Блок-ориентированные устройства поддерживают позиционирование с помощью вызова lseek().

Позиционироваться можно в любую точку, которая необязательно должна находиться точно на границе блока или сектора.

Прочитать и записать можно произвольное количество данных. ОС позаботится, чтобы прочитать соответсвующий блок, если начало и/или конец порции не находятся на границах, модифицировать их содержимое и записать обратно.

Байт-ориентированные устройства операцию позиционирования не поддерживают.

Это основное отличие устройств одного типа от другого.

Над устройствами определены дополнительные операции, специфические для конкретного устройства. Например, открытие/закрытие лотка привода, установку скорости обмена и канального протокола последовательного порта, низкоуровневую разметку гибких дисков, управление громкостью воспроизведения звука, и т.п.

Все операции данной категории выполняются с помощью системного вызова управления устройствами — ioctl()