Пиздец, я так не хочу этим заниматься....

Оглавление

[1. Inode 2](#_Toc14467)

[2. Ссылки 2](#_Toc5910)

[3. Потоки (не ввода - вывода, а потом вводы и вывода) 3](#_Toc10024)

[4. Экранирование 4](#_Toc8388)

[5. Команды фильтры 5](#_Toc28389)

[6. Права 5](#_Toc27218)

[7. Командная оболочка 5](#_Toc17039)

[8. Компилятор интерпретатор 6](#_Toc9848)

[9. Интерфейс командной строки 6](#_Toc4173)

[10. Терминал 6](#_Toc16335)

[11. Консоль 6](#_Toc8502)

[12. Пути 6](#_Toc18539)

[13. Переменные окружения 7](#_Toc24322)

[14. Переменные оболочки 7](#_Toc27469)

[15. Сигналы 7](#_Toc14380)

[16. More 8](#_Toc5755)

[17. Типы файлов 8](#_Toc7735)

[18. Upd командная оболочка (?) 9](#_Toc2749)

[19. Маски 9](#_Toc14061)

#### Inode

Inode - если очень кратко, то это хранлище метаданных(тип файла, права, счетчик ссылок, id пользователя, id группы, размер файла, времена доступа, модификации и создания, атрибуты, список контроля доступа(спец пользователи/группы), ссылка на расположение файла (блоки) и т.д.).

А вот теперь, если не кратко: inode(i-node) - индексный дескриптор - структура данных, хранящая метаданные файла(смотри выше),и в которой перечислены блоки с данными файла(вот про это дальше будет инфа). inode - уникальный индекс, присваемый каждому файлу файловой системы, для хранения свойств этого же файла. Inode хранит в себе так же и блоки(инфу, в каких блоках инфа, вот так), в которых хранится содержимое файла. Количество индексов ФС ограничено, как, например, любая память на любом устройстве. Если закончатся inod-ы, то файлы невозможно будет создать даже при наличии свободного места в файловой системе. По умолчанию размер одного блока ФС равен 4092 байта. В начале раздела расположен суперблок, в котором находятся метаданные всей файловой системы, за ним идут несколько зарезервированных блоков, а затем размещена таблица Inode и только после неё блоки с данными. Таким образом, все Inode размещены в начале раздела диска.  
  
 Подводя итог - инод связанное с именем файла число, связывающее непосредственно имя файла с таблицей метаданных и хранящее информацию о блоках с содержимым файла. Я написала много одной и той же информации разными словами... Но в моей голове мне всё понятно.

#### Ссылки

Счётчик ссылок - кол-во жестких ссылок. Ну и,собственно, ссылка - указатель на другой файл.

Символические(программные) ссылки - файл, содержимое которого указывает на связанный файл. А-ля ярлык. Если удалить или переместить исходник, то ссылка нарушится и файл (ссылка==файл) будет недоступен. Содержимое символической ссылки - только имя целевого файла. У символической ссылки по дефолту стоят максимальные права, ибо права не управляются. Исходный файл - имя, которое связано уже с инодом, а символическая ссылка ссылается на имя. Размер ссылки - кол-во байтов в имени файла, на которое она ссылается.

Жёсткие ссылки - имя, которое ссылается не на имя другого файла, а на инод. То есть создание жёсткой ссылки - добавление ещё одного имени в инод. Разницы между жёсткой ссылкой и исходником, по сути, нет - два имени связаны с одним индексным узлом. Жёсткая ссылка - не спец.файл, а просто файл, связанный с тем же инодом, что и родитель -> идентичный родителю.

Но

* -жесткие ссылки на директории создавать нельзя, так как структура каталогов в линукс - циклическая и по идее, каждая директория, кроме root, содержит жесткую ссылку на саму себя и на ссылку каталога выше (в root жесткая ссылка только на себя). Создав ещё одну жесткую ссылку, вы поломаете дерево каталогов ФС ибо оно станет ациклическим (пример потом вставлю как-нибудь).
* -жесткие ссылки невозможно создавать между ФС, ибо у каждой ФС своя база inod - > индексы могут повторяться.
* -удаление жесткой ссылки или родителя не повлечет за собой удаления информации, потому что информация хранится в блоках, инфа о блоках хранится в иноде, а файлы, связанные с инодом - жесткие ссылки -> пока существует хотя бы одна жесткая ссылка, данные будут существовать в ФС.

Пункты 1-2, но нормальным языком - [Inode-рассказываю и показываю на примерах в системе Линукс - Huawei ICT Club](https://forum.huawei.com/enterprise/ru/inode-%D1%80%D0%B0%D1%81%D1%81%D0%BA%D0%B0%D0%B7%D1%8B%D0%B2%D0%B0%D1%8E-%D0%B8-%D0%BF%D0%BE%D0%BA%D0%B0%D0%B7%D1%8B%D0%B2%D0%B0%D1%8E-%D0%BD%D0%B0-%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80%D0%B0%D1%85-%D0%B2-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B5-%D0%BB%D0%B8%D0%BD%D1%83%D0%BA%D1%81/thread/830419-100139)

#### Потоки (не ввода - вывода, а потом вводы и вывода)

Потоки - выполнение последовательности машинных инструкций/ последовательное выполнение цепочки инструкций/ нити исполнения в одном процессе - базовая единица загрузки ЦП(У), состоящая из идентификатора потока, счётчика, регистров и стека. В Linux каждый поток - процесс -> чтобы создать новый поток, нужно создать процесс (я хуй знает, так это или не так). Ремарка: процесс - хранилище потоков.

Факты? (слишком дохуя всего)

* Потоки - выполняемые задачи/функции
* Потоки делят между собой адресное пространство, в отличие от процессов, которые более изолированы друг от друга. Из-за единого адресного пространства потоки конкурируют между собой: стек и локальное хранилище потока захватывают часть виртуального адресного пространства , не позволяя занять его другим потокам соответственно
* Потоки быстрее обмениваются данными по сравнению с процессами, и для ОС проще создать поток, чем процесс

**Потоки ввода-вывода**

STDIN(0 - дескриптор потока) - standard input - поток с информацией, передаваемой в терминал (инструкции, переданные в оболочку для выполнения). Данные попадают в поток с клавиатуры обычно

STDOUT(1) - standard out - поток данных, которые [оболочка](#_Командная оболочка) выводит после выполнения каких-то инструкций. Инфа в итоге попадает в окно терминала.

STDERR(2) - standard error - поток ошибок, очень схожая с потоком вывода штука, ибо данные тоже оказываются на экране терминала, но этими потоками можно управлять раздельно (перенаправлять).

**< >** - знаки перенаправления.

**>** - перенаправление вывода|перезаписывает файл

**>>** - добавляет новые данные в файл после старых

**<** - перенаправление ввода

**<<** - я ничего не нашла, но в табличке по опд был этот символ, мб он опечатался, конечно, но пускай будет

$ comm <(sort list1.txt) <(sort list2.txt)

(перенаправление отсортированных потока данных файлов в команду comm)

1. - перенаправление потока ошибок (2 -дескриптор потока ошибок)

**&>**- перенаправление двух потоков (вывод и ошибки)

| - оператор потока PIPE - перенаправляет поток вывода одной команды в поток ввода для другой(?)/ выходные данные первой команды служат входными данными для второй команды

#### Экранирование

Экранирование - замена управляющих символов на соответствующие текстовые.

Одинарные кавычки - выводят заключённую в них инфу буквально (типа интерпретирует написанное как строку текста)

$ text='a $(echo b) c'

$ echo "${text}"

a $(echo b) c

Двойные кавычки - выводят данные, полученные в ходе выполненной инструкции, заключённой в двойных кавычках.

$ text="a $(echo "b") c"

$ echo "${text}"

a b c

**Экранирование спец символов в bash** - экранирование в двойных кавычках

\ - экранирует спец символы, превращая их в составляющие текста (?)

Символы, которые нужно экранировать

**$**

**`**

**“**

**\**

**!**

**<новая строка> - не \n**

$ - $() - выполняет команды в скобках, ${} - выводит значения переменных

Специальное назначение некоторых экранированных символов используемых совместно с echo и sed

\n - перевод строки (новая строка)

\r - перевод каретки

\t - табуляция

\v - вертикальная табуляция

\b - забой (backspace)

\a - "звонок" (сигнал)

\0xx - ASCII-символ с кодом 0xx в восьмеричном виде)

#### Команды фильтры

Команды из задания с grep, head и т.д. - это всё команды фильтры для сортировки и фильтрации информации.

#### Права

Три вида доступа - чтение, запись, выполнение - для трёх видов пользователей - владелец, группа, остальные

Про них всё понятно короче, ещё способы задания - цифры и буквы

#### Командная оболочка

Командная оболочка - интерпретатор командной строки/ программа, фактически обрабатывающая команды и выдающая результат/ пользовательский интерфейс для доступа к службам ОС и взаимодействия с ней

Командные оболочки Linux Unix

Bourne shell - исполняемый файл: sh

Bourne again shall - исполняемый файл: bash -командный интерпретатор, выполняющий команды, прочитанные со стандартного входного потока или из файла

C shell - исполняемый файл: csh

TENEX C shell - исполняемый файл: tsch

#### Компилятор интерпретатор

Компилятор - преобразует программу, написанную на высокоуровневом языке в машинный/какой-то другой (байт код, например) код (низкоуровневый язык), понимаемый для процессора.

Интерпретатор - выполняет код программы построчно, не используя машинный код, а работая непосредственно с языком программирования. Исполняемый файл, поэтапно читающий программу и выполняющий её, как инструкцию и часть собственного исполняемого файла, построчно.

#### Интерфейс командной строки

Управление программами с помощью командной строки :) . Собственно, основной инструмент - клавиатура, с которой вводятся команды, выполняемые построчно, при нажатии enter. Интерфейс встроен в ядро, поэтому запустится даже если графический интерфейс не запустится. Добраться можно через консоль (как добраться до консоли - хз, но виртуальных консолей 7 - переключаться можно ctrl+alt+Fn (n-номер консоли)- у 1-6 интерфейс командной строки - у 7-й графический режим) и терминал (графический эмулятор консоли)

Интерфейс командной строки - любой тип интерфейса, используемый для ввода команд. (терминал, например)/ программа, обрабатывающая команды, лл. Пользователь взаимодействует с командной оболочкой через интерфейс командной строки

#### Терминал

Среда ввода и вывода текста. По сути, программа, отображающая графический интерфейс, позволяющая запускать оболочку и работать с командами/взаимодействовать с OC.

#### Консоль

Физический терминал (хз, напрямую подключён к компу), который распознаётся ОС как терминал (реализованный в ядре).

#### Пути

Абсолютный - из корневой папки

Относительный - из фактической директории

#### Переменные окружения

Переменные окружения - переменные, определённые для используемой оболочки и наследуются любыми дочерними оболочками/процессами. Переменные окружения используются для передачи информации в процессы, порождаемые оболочкой

Окружение - область, автоматически создаваемая оболочкой при запуске сеанса, содержащего переменные, определяющие системные свойства

Локальные переменные окружения - эти переменные определены только для текущей сессии. Они будут безвозвратно стерты после завершения сессии, будь то удаленный доступ или эмулятор терминала. Они не хранятся ни в каких файлах, а создаются и удаляются с помощью специальных команд.

Системные переменные окружения - Эти переменные доступны во всей системе, для всех пользователей. Они загружаются при старте системы из системных файлов конфигурации:  /etc/environment, /etc/profile, /etc/profile.d/ /etc/bash.bashrc.

#### Переменные оболочки

Переменные, созданные и используемые в самой оболочке. Используются для отслеживания каких-то невидимых данных (текущий рабочий каталог, например).

Пользовательские переменные оболочки - эти переменные оболочки в Linux определяются для конкретного пользователя и загружаются каждый раз когда он входит в систему при помощи локального терминала, или же подключается удаленно. Такие переменные, как правило, хранятся в файлах конфигурации: .bashrc, .bash\_profile, .bash\_login, .profile или в других файлах, размещенных в директории пользователя.

# Сигналы

Сигнал - уведомление процесса о каком-либо событии, способ взаимодействия процессов между друг другом.

|  |  |  |
| --- | --- | --- |
| **Код сигнала** | **Название** | **Описание** |
| 1 | SIGHUP | Закрытие терминала |
| 2 | SIGINT | Сигнал остановки процесса пользователем с терминала (CTRL + C) |
| 3 | SIGQUIT | Сигнал остановки процесса пользователем с терминала (CTRL + \) с дампом памяти |
| 9 | SIGKILL | Безусловное завершение процесса |
| 15 | SIGTERM | Сигнал запроса завершения процесса |
| 17 | SIGSTOP | Принудительная приостановка выполнения процесса, но не завершение его работы |
| 18 | SIGTSTP | Приостановка процесса с терминала (CTRL + Z), но не завершение работы |
| 19 | SIGCONT | Продолжение выполнения ранее остановленного процесса |

1. **More**

[Команда more в Linux - Losst](https://losst.pro/komanda-more-v-linux)

1. **Типы файлов**

Обычные файлы

* Текстовые
* Исполняемые
* Изображения
* Файлы архивов
* Файлы библиотек программ
* И т.д.

Специальные файлы

* Блочные файлы - файлы, которые могут передать большой блок данных за небольшой один раз, так как собираемые данные собираются в буфере до момента их сохранения/передачи/использования.
* Символьные файлы - обеспечивают не буферизованный доступ к аппаратным компонентам и ядру. Передают по одному символу за раз, но в целом, устроены так же, как и блочные
* Символические ссылки
* Туннели и именованные туннели - файлы, позволяющие настроить взаимодействие между процессами, перенаправив вывод одного файла на вход другого. (pipe)
* Файлы сокетов - файлы, обеспечивающие связь между процессами в обе стороны

Каталоги - файла с именами других файлов

1. **Upd командная оболочка (?)**

Символ тильды ~ в Linux-системах – это сокращенное обозначение домашнего каталога пользователя (/home/user). Именно эту директорию Bash делает текущей по умолчанию.

Знак доллара $ обозначает, что мы работаем под непривилегированным пользователем. Если находиться в системе под администратором, вместо доллара будет знак решетки #.

1. **Маски**

Важное свойство большинства командных оболочек системы Linux — способность обращаться более, чем к одному файлу по имени с использованием специальных символов — масок

\* - любой символ/последовательность символов

? - один любой символ