Шпаргалка к Лабораторной работа №1

7 апреля 2021 г.

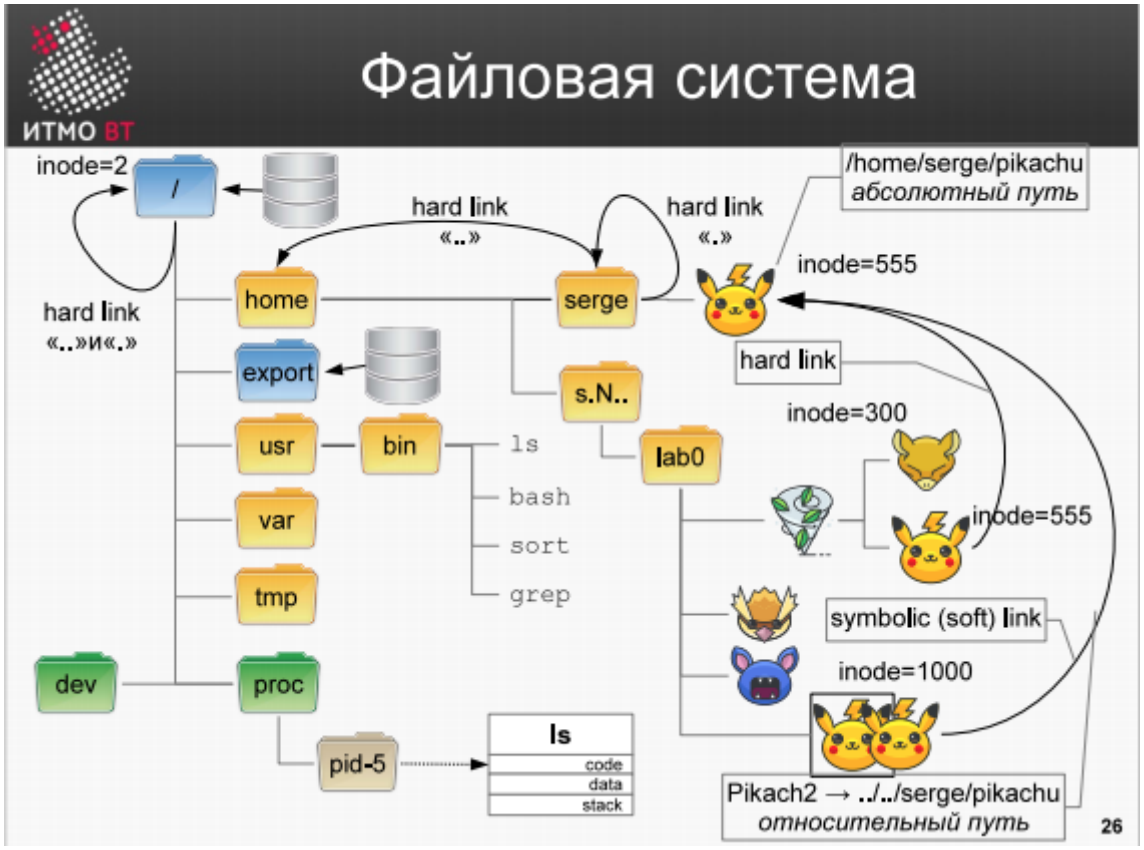


Рис. 1: Caption

Те, кто работает в Windows привыкли к такой абстракции как диск.

В UNIX принято, что у нас есть единая файловая система состоящая из разных дисков(на слайде синим цветом указаны точки монтирования).

Корнем иерархии является директория которая называется корень и обознаечается "/". Это всегда самая верхняя директория в иерархии.

Каждая директория содержит назначение собственное (home — папки пользователей).

Что такое файл?

Файл — некоторая абстракция которая позволяет логически сгруппировать однородную информацию.

Если мы взяли несколько файлов и скрепили, то это тоже файл только с упорядоченной информацией.
Скрепка тут — прообраз метаинформации(информация о файле)

То есть у нас сама информация, а есть **некоторое описание этого файла** которое в UNIX представлено в виде абстракции **inode**(содержит часть данных файла, общую информацию, права, нумеруется в виде последовательности цифр)

Обычно под метаинформацию отведено 1% от кол-ва памяти на диске.

Файлы в ОС UNIX — номер файла, метаинформация и данные(имени по умолчанию у файла нет).

Имя — запись в директории, которая определяет соответствие набора символов и номера файла.

Номер содержится внутри файловой системы и им управляет ОС. А когда создается файл, то **ОС выделяет пустой файл с номером** и мы его подключаем к директории и **даем имя**.

Директория — совокупность файлов.

Важный момент

В UNIX это имён для одного файла не ограничено. **Записей о файле может быть сколько хотим.** Мы можем создать файл с одним именем, потом сделать на него ссылку которая будет иметь другое имя, но она будет такая же(важен номер файла номер файла).

Жесткая ссылка — один и тот же файл. Файл находится на диске, а информация о этом файле находится в каталоге.

- Каждому жестко связанному файлу присваивается то же значение Inode, что и в оригинале, поэтому они ссылаются на одно и то же физическое местоположение файла. Жесткие ссылки более гибкие и остаются связанными, даже если исходные или связанные файлы перемещаются по всей файловой системе, хотя жесткие ссылки не могут пересекать разные файловые системы.
- Мы не можем создать жесткую ссылку на файл, который находится на другом физическом диске, тк список inode на каждом диске свой и номера уникальны на каждом жестком диске.
- Команда ls -l показывает все ссылки с колонкой ссылок, показывает количество ссылок.
- Ссылки содержат фактическое содержимое файла
- Удаление любой ссылки просто уменьшает количество ссылок, но не влияет на другие ссылки.
- Мы не можем создать жесткую ссылку на каталог, чтобы избежать рекурсивных циклов.
- Если оригинальный файл удален, то ссылка все равно покажет содержимое файла.
- Команда для создания жесткой ссылки:

```
$ ln [original filename] [link name]
```

Когда мы создаем директорию, то в ней всегда обычно создаются **две служебных записи о директории**: файл "."и файл "..".

При этом файл "."это жесткая ссылка на ту же самую директорию которую мы создаем.

На пустые файлы нельзя создавать жесткие ссылки.

То есть, при создании директории у нас будут созданы две жесткие ссылки: запись директории в которой находится созданная директория(родительской директории), а вторая это запись созданной директории.

Руками мы эти жесткие ссылки создать не можем, этим занимается ОС.(Нам запрещено создавать жесткую ссылку на директорию, а на файлы мы можем руками создать)

Кроме всего прочего, в директории могут существовать также символические ссылки(мягкие) — просто ссылка по имени(при создании такой ссылки будет имя с которой мы ее создаем и имя на которое она будет указывать).

- Мягкая ссылка похожа на функцию ярлыка файла, которая используется в операционных системах Windows. Каждый мягкий связанный файл содержит отдельное значение Inode, которое указывает на исходный файл. Подобно жестким ссылкам, любые изменения данных в одном файле отражаются в другом. Программные ссылки могут быть связаны между собой в разных файловых системах, хотя, если исходный файл удален или перемещен, файл с мягкими ссылками не будет работать правильно (это называется зависшей ссылкой).
- Жесткие ссылки могут работать в пределах одного физического диска, несмотря на то, что они находятся в общей иерархии, а символические ссылки могут тк это просто путь
- Команда ls -l показывает все ссылки со значением первого столбца l? и ссылка указывает на оригинальный файл.
- Soft Link содержит путь к исходному файлу, а не его содержимое.
- Удаление программной ссылки ни на что не влияет, кроме удаления исходного файла, ссылка становится «висячей» ссылкой, которая указывает на несуществующий файл.
- Мягкая ссылка может ссылаться на каталог.
- Связь между файловыми системами. Если вы хотите связать файлы между файловыми системами, вы можете использовать только символические / программные ссылки.
- Команда для создания Soft ссылки:

```
$ ln -s [original filename] [link name]
```

В ней будет записан путь к файлу, на который мы ссылаемся(абсолютный или относительный)

Мы всегда находимся в какой-то директории(текущая директория), поэтому ссылки могут указываться по разному:

- Относительный путь — Последовательность нашего пути до коечного файла от начального файла.
- Абсолютный путь — путь, начинающийся от корня.

В UNIX нету папок и каталогов, есть только директории.
Домашний каталог пределяется при создании пользователя.

Директория **в реализации** это тот же файл.(имя файла — номер)

Тильда просто удобное сокращение /home/<имя пользователя>(не все shell поддерживают сокращение)

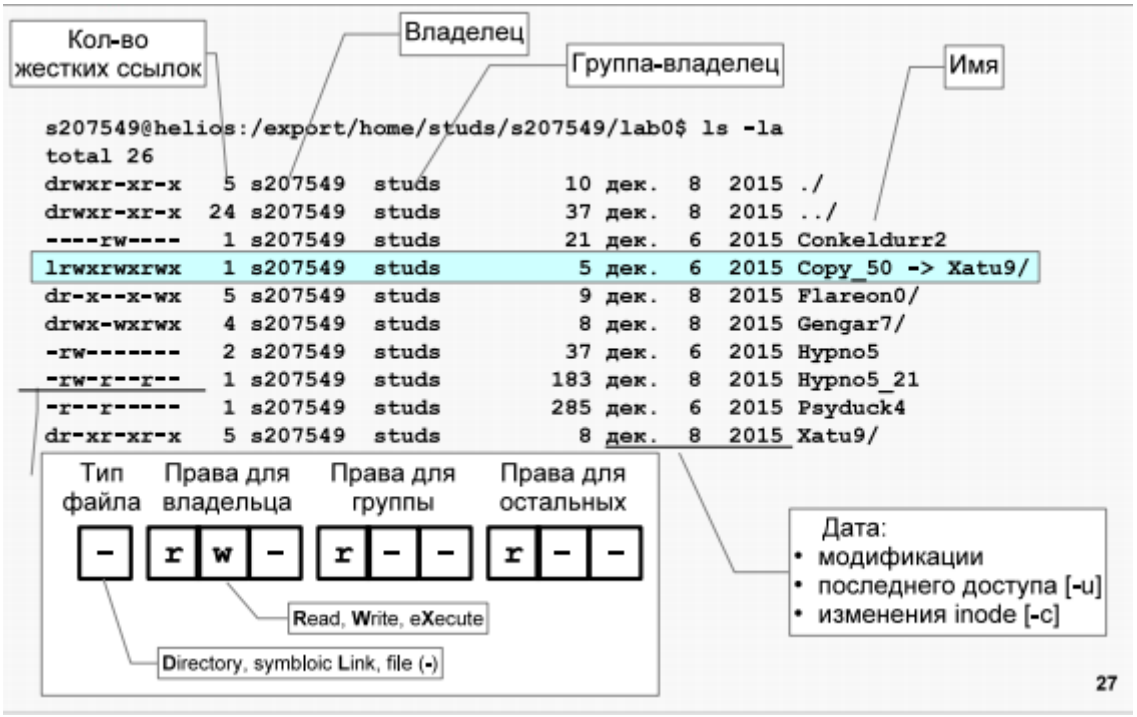
Служебные директории:

- \usr\bin — содержаться пользовательские команды(утилиты).
- \var — переменная(variable) конфигурационная информация.(там лежат логи)
- \tmp — временные файлы.

- \dev — специальные файлы драйверов устройств.
- \dev\ttv — отображение текущего терминала
- \dev\null — всё, записанное в этот файл исчезает
- \dev\zero — служебный файл, предназначенный для генерации нулей.
- \dev\random — служебный файл, предназначенный для генерации псевдослучайных чисел.
- \dev\null — любая информация записанная в этот файл удаляется
- \proc — входит в группу виртуальных директорий, которые представляют различные системные части в виде файлов. proc представляет работающие процессы в виде файлов, которые его заражтеризуют.

Корневой inode всегда имеет номер 2.

Права доступа:



Метаинформация — служеюная информация о том, что такое файл.

Она хранит:

- Имя файла(последовательность символов, любых)
- Дата последней модификации(по умолчанию)
- Дата последнего доступа к файлу[-u]
- Дата изменения inode[-c]
- Длина файла в байтах Длина символической ссылки — длина пути
- Права
- Владелец файла
- Группа—владелец файла
- Кол-во жестких ссылок(кол-ва записей в директориях о файлах с одним и тем же номером этого файла(пятая жесткая ссылка расположена в директории s207549 и называется она lab0))
- Метаинформация Жесткие ссылки обозначаюится [-] тк это файл

Для того чтобы вывести список файлов директории используется команда **ls(акроним от list)** с ключами -a(выводит скрытые файлы) и -l(выводит метаинформацию о файле, файл будет длинным)

В UNIX принято что файл является скрытым и невидимым по умолчанию если он начинается **с точки**.(по умолчанию такими являются файлы "."и "..")

Каждый файл когда он создается по умолчанию у пользователя обычно есть эффективные права(Когда вы вошли в систему вам ассоциирована реальная эффективная группа и пользователь) файл по умолчанию создасться с именем под которым мы вошли в систему с той группой под которой мы вошли в систему.(у файла есть всегда два владельца — пользователь и группа(каждый

пользователь группы) которой он принадлежал)

Группы можно менять(есть первичная группа — запись в файле passwd и вторичная группа к которой он может принадлежать — регулирует тот набор прав который пользователь получает при доступе к файловым объектам)

Выделяют **три категории пользователей**, которым могут предоставляться права на файл:

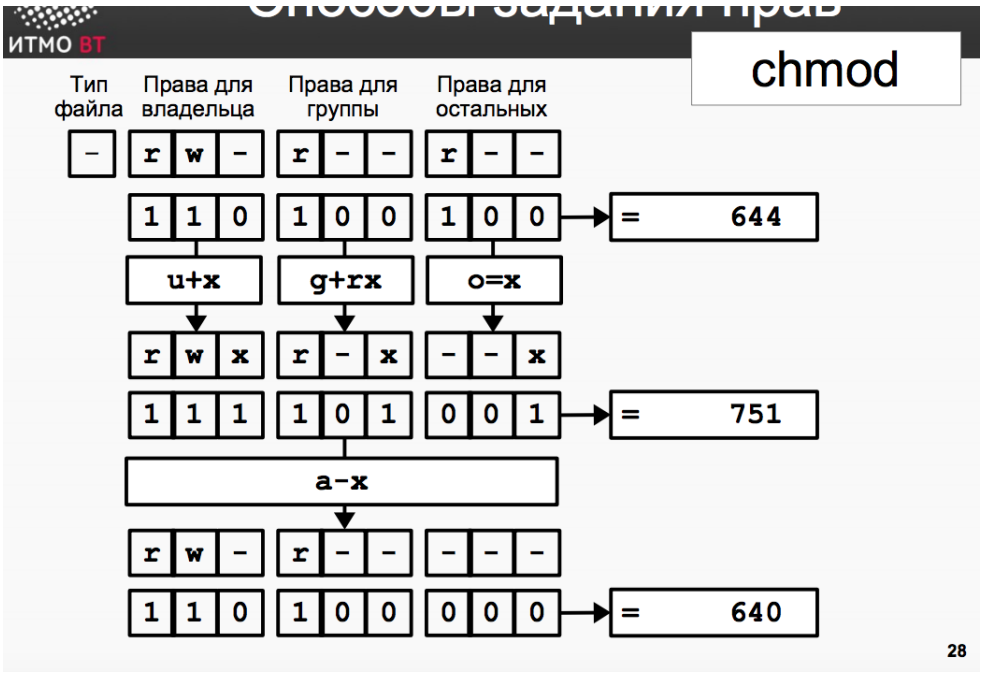
- **Сам владелец (u – user)** объекта — конкретный пользователь, чье имя **числится в атрибутах файла как имя владельца** этого файла.
- **Группа (g – group)**, к которой принадлежит владелец файла. Когда в UNIX создается пользователь, то для него **создается одноименная группа**.

  Однако **средствами администрирования системы можно объединять пользователей** в различные группы. При этом конкретный пользователь может входить **в состав нескольких групп**.
- **Все остальные (o – other)** — это все те, кто **не является владельцем** файла и **не принадлежит к группе** владельца файла.

!!Права на файл могут изменять **только владелец этого файла и суперпользователь**.*(команда chmod)*

Чтение, запись, выполнение — это то, что можно **делать с существующим файлом**. У каждой категории пользователей (владельца, группы, остальных) **должны быть свои права** на каждое вышеупомянутое действие.

Права доступа к файлам:



Буквенная запись прав доступа

Поскольку есть три категории пользователей и три действия над файлом, то получается, что в атрибутах файла должно быть девять записей о правах, **указывающих на то, кто и что может делать с файлом**.

Первые три записи – это **права владельца**
Вторые три записи – **права группы**
Последняя тройка – **права на файл для всех остальных**

Если **обозначить каждое право соответствующей буквой**, и все права всем предоставляются, то получится такая запись:

drwxrwxrwx - для директорий
-rwxrwxrwx - для остальных файлов

В **выводе команды ls -l**. В первом столбце, после символа типа файла указываются права доступа к файлу в буквенной нотации.

Если какое-либо право не предоставляется той или иной категории, то вместо буквы **ставится знак минус**.

Запись прав доступа с помощью чисел

Если положительное значение права (когда доступ есть) обозначать **единицей**, а отрицательное (доступа нет) – **нулем**, то получим такую картину для файла, который могут все читать, но изменять только владелец:

110100100

Однако запись из девяти символов **достаточно длинная**. Поэтому права доступа трех категорий (владельца, группы, остальных) можно выразить как **три независимых друг от друга числа от 0 до 7-ми**.

Часто используемые числа, которые полезно запомнить:
0: $---$
1: $--x$
2: $-w-$
3: $-wx$
4: $r--$
5: $r-x$
6: $rw-$
7: rwx

Права доступа к каталогам:

Каталог в Linux – это особый тип файла, смысловым содержанием которого является список других файлов.

Каталоги **имеют те же биты прав**, что и остальные файлы. Однако значение битов **отличается**:

- Право на чтение в случае каталога означает лишь возможность узнавать список содержащихся в нем файлов. При этом что-то узнать о свойствах файлов (размере, правах) не получится.
- Если каталог можно изменять, то можно изменять состав файлов в нем, то есть можно удалять, переименовывать файлы, создавать новые. Причем все это возможно даже с файлами, доступ к которым запрещен.
- Право на исполнение в случае каталога означает, что в него можно заходить, просматривать и изменять содержимое файлов*(если на это есть разрешения)*, узнавать свойства файлов.

  Однако нет права изменять имена файлов в этом каталоге.

  Если для каталога установлено только право на исполнение, то его содержимое просматривать нельзя. Можно лишь выполнять поиск файлов и обращаться к известным файлам

Отметим пару вытекающих из вышесказанного особенностей:

- Доступ к конкретному файлу зависит от наличия доступа на исполнение к каталогам на протяжении всего пути.
- Изменять существующие файлы можно, не имея доступа на запись в каталог, достаточно иметь доступ на запись самого файла.

Нужно дописать про sticky bit, suid, sgid

Что касается процессов, то с ними связано не два идентификатора, а 4-е: реальный и эффективный пользовательский (UID), а также реальный и эффективный групповой (GID). Реальные номера применяются для учета использования системных ресурсов, а эффективные для определения прав доступа к процессам. Как правило, реальные и эффективные идентификаторы совпадают. Владелец процесса может посылать ему сигналы, а также изменять приоритет.

Процесс не может явно изменить ни одного из своих четырех идентификаторов, но есть ситуации когда происходит косвенная установка новых эффективных идентификаторов процесса. Дело в том, что существуют два специальных бита: SUID (Set User ID - бит смены идентификатора пользователя) и SGID (Set Group ID - бит смены идентификатора группы). Когда пользователь или процесс запускает исполняемый файл с установленным одним из этих битов, файлу временно назначаются права его (файла) владельца или группы (в зависимости от того, какой бит задан). Таким образом, пользователь может даже запускать файлы от имени суперпользователя.

Например, как говорилось выше, команду chmod по умолчанию может запускать только root. Если мы установим SUID на исполняемый файл /bin/chmod, то обычный пользователь сможет использовать эту команду без использования sudo, так, что она будет выполнятся от имени пользователя root. В некоторых случаях очень удобное решение. Кстати по такому принципу работает команда passwd, с помощью которой пользователь может изменить свой пароль.

- Восьмеричные значения для SUID и SGID - 4000 и 2000.
- Символьные: u+s и g+s.

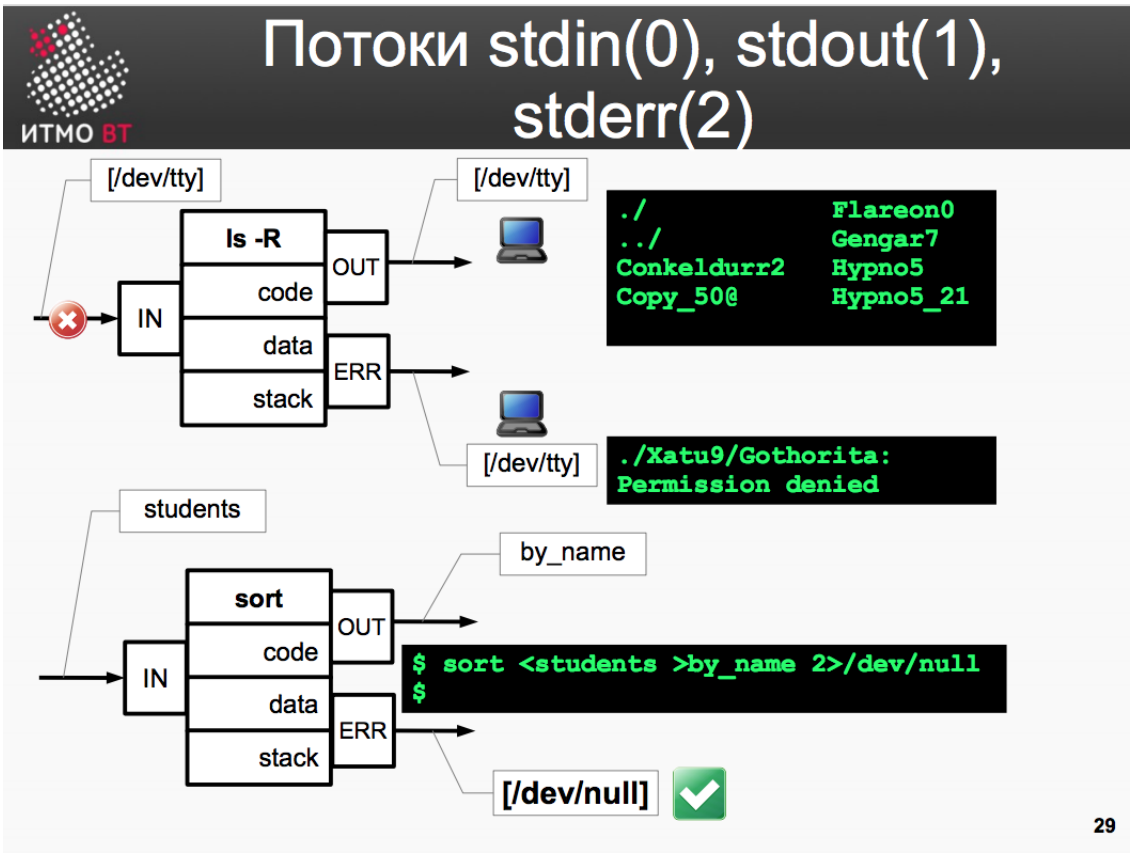
Видно, что для файла qwert установлен SGID, о чем свидетельствует символ «s» (-rwxrwsrwx). Символ «s» может быть как строчная буква (s), так и прописная (S). Регистр символа только лишь дает дополнительную информацию об исходных установках, т.е. был ли до установки SGID установлен бит, в данном случае на выполнение (rwxrwsrwx). Если s строчная, то права на выполнение у группы этого файла были до установки SGID. Если S прописная, то группа для этого файла ранее не имела прав на выполнение до установки SGID.)

Еще одно важное усовершенствование касается использования sticky-бита в каталогах.10) Каталог с установленным sticky-битом означает, что удалить файл из этого каталога может только владелец файла или суперпользователь. Другие пользователи лишаются права удалять файлы.11) Установить sticky-бит в каталоге может только суперпользователь. Sticky-бит каталога, в отличие от sticky-бита файла, остается в каталоге до тех пор, пока владелец каталога или суперпользователь не удалит каталог явно или не применит к нему chmod. Заметьте, что владелец может удалить sticky-бит, но не может его установить.

- Восьмеричное значение stiky-бита: 1000
- Символьное: +t

0 — setuid, setgid, sticky bits не установлены.
1 — Установить sticky bit.
2 — Установить setgid bit.
3 — Установить setgid и sticky bits.
4 — Установить setuid bit.
5 — Установить setuid и sticky bits.
6 — Установить setuid и setgid bit-ы.
7 — setuid, setgid, sticky bits активированы.

Перенаправление потоков:



Потоки — 3 файловых дескриптора. Когда мы запускаем программу они создаются средой исполнения программы для любого файла. Когда мы запускаем утилита с помощью shell, то он связывает нашу утилиту со стандартными терминалами.

>> file или 1>> file перенаправляет стандартный поток вывода в файл. Если файл не существует, он будет создан, если существует — данные будут дописаны к нему в конец. (Например, "ls >> newfile")

> file или 1> file перенаправляет стандартный поток вывода в файл. Если файл не существует, он будет создан, если существует — перезаписан сверху.

2> file перенаправляет стандартный поток ошибок в файл. Если файл не существует, он будет создан, если существует — перезаписан сверху.

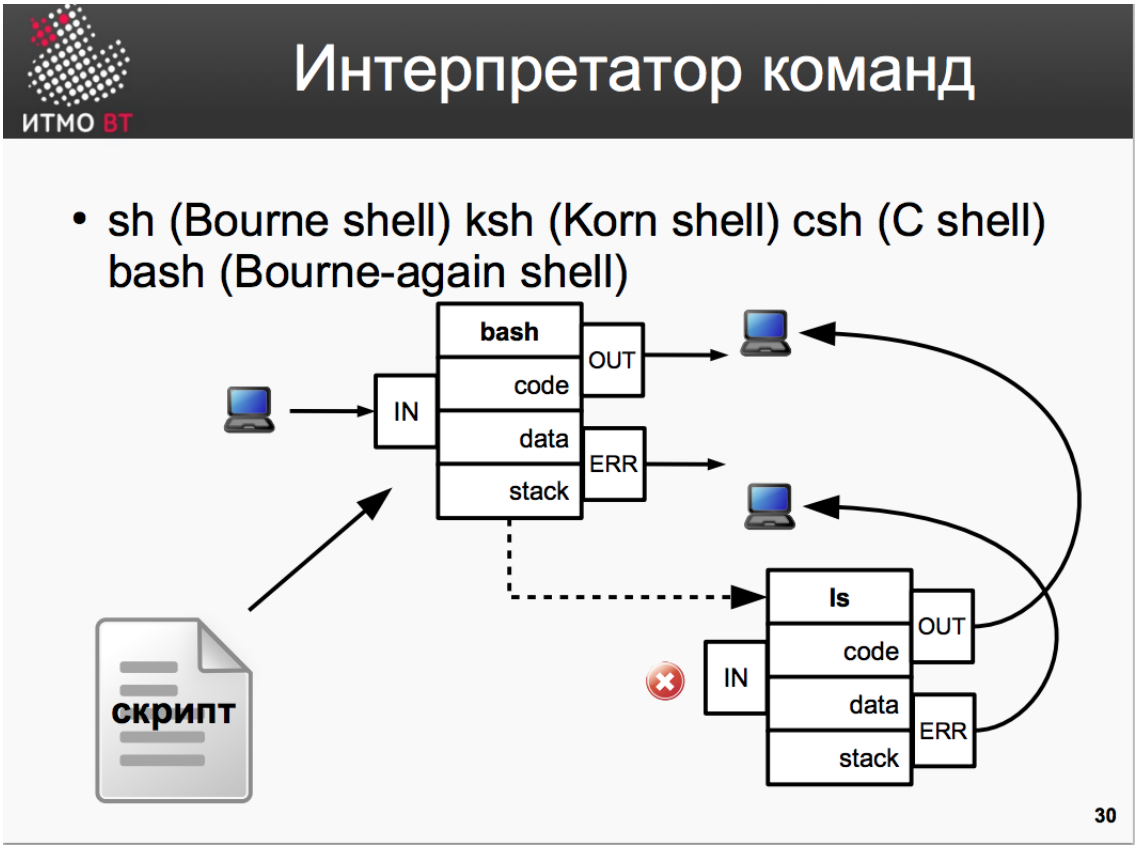
2>> file перенаправляет стандартный поток ошибок в файл. Если файл не существует, он будет создан, если существует — данные будут дописаны к нему в конец

Пример: ls -l myfile xfile 2> errorcontent 1> correctcontent
Файл errorcontent будет содержать сведения об ошибках выполнения команды ls -l myfile xfile.
Файл correctcontent будет содержать вывод команды ls -l myfile xfile.

&> file перенаправляет и stderr, и stdout в один файл.

1>2 и 1>>2 перенаправляют поток вывода в файл, в который направлен поток ошибок.

Интерпретатор команд:

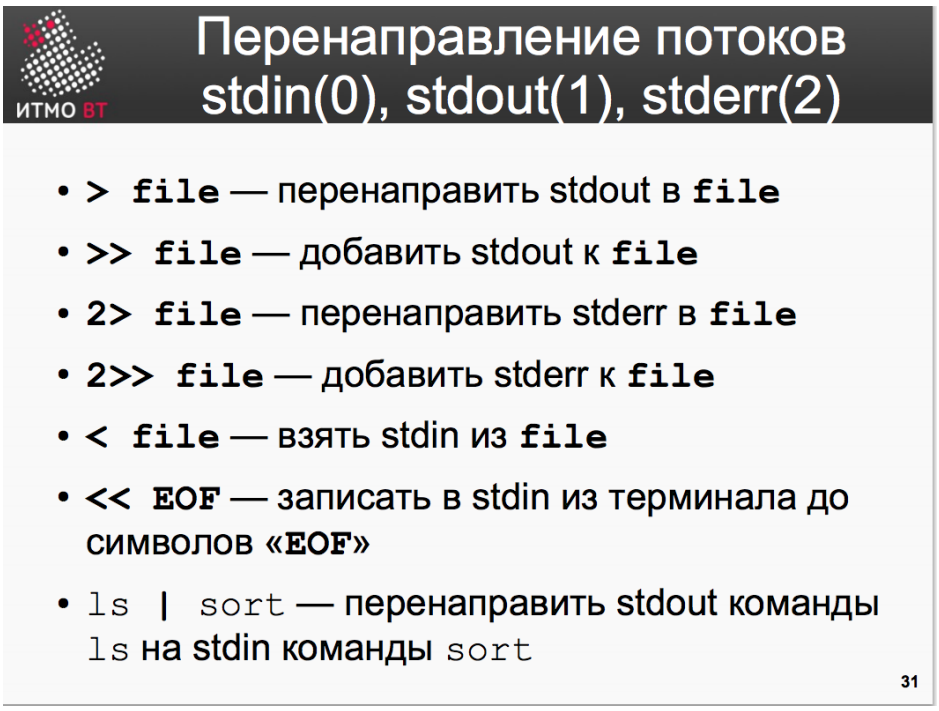


Shell на основании данных из команды или скрипта(содержит перечень команд которые нам нужно выполнить и также есть инструкции ветвления вначале стоит shebang который указывает какой shell мы используем) осуществляет связывание стандартных потоков с тем что требуется. На helios стоит ksh(korn shell).

Shell мы можем вывести выведя переменную среды SHELL с помощью команды:

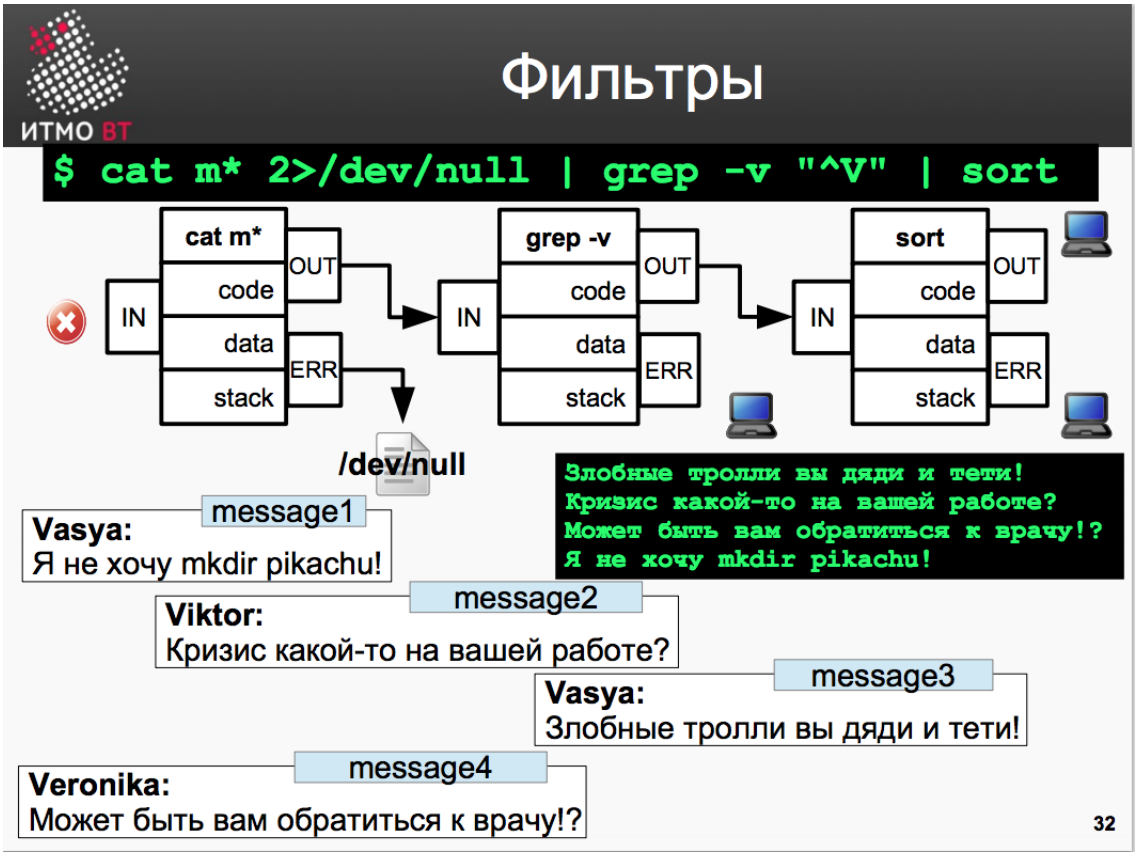
```
$ printenv SHELL
```

(Нам выведет путь до интерпретатора)
В shell'ах перенаправление потоков мы можем видеть на слайде:



Если одна галка то файл учечётся до нуля и потом запишеться вывод.

Фильтры:



Команды, которые были реализованы для использования совместно с программными каналами, называются фильтрами. Эти фильтры реализуются в виде простейших программ, которые крайне эффективно выполняют одну определенную задачу. Исходя из всего вышесказанного, они могут использоваться в качестве строительных блоков при создании сложных конструкций.

В данной главе представлена информация о наиболее часто используемых фильтрах. В результате комбинирования простых команд и фильтров с использованием программных каналов могут быть созданы элегантные решения.

Фильтрация осуществляется командами:

- **cat** — При размещении фильтра cat между двумя программными каналами не будет осуществляться какой-либо обработки передающихся через них данных (за исключением передачи этих данных из стандартного потока ввода stdin в стандартный поток вывода stdout фильтра).
- **tee** — Фильтр tee перемещает данные из стандартного потока ввода stdin в стандартный поток вывода stdout, а также записывает их в файл. Исходя из вышесказанного, фильтр tee функционирует аналогично фильтру cat, за исключением того, что он имеет два идентичных вывода.
- **grep** — Наиболее простым сценарием использования фильтра grep является фильтрация строк текста, содержащих (или не содержащих) определенную подстроку.
  - grep -i, который позволяет производить фильтрацию строк без учета регистра.
  - grep -v, который позволяет осуществлять вывод строк, не содержащих заданную строку.
  - grep -A1 в вывод также будет добавлена одна строка, располагающаяся после обнаруженной строки.
  - grep -B1 в вывод будет добавлена одна строка, располагающаяся до обнаруженной строки.
  - grep -C1 (контекст) в вывод может быть добавлена одна строка, находящейся до обнаруженной строки, и одна строка, находящаяся после нее. Все три параметра (A, B и C)

могут быть использованы для вывода произвольного количества дополнительных строк (например, могут быть использованы параметры A2, B4 или C20).

- **cut** — Фильтр cut может использоваться для извлечения данных из столбцов расположенных в файлах таблиц с указанием разделителя столбцов или количества байт данных в столбцах.

  В случае использования фильтра cut с символом пробела в качестве разделителя вам придется экранировать этот символ пробела.

- **tr** — Вы можете преобразовывать символы с помощью фильтра tr.

  Параметр tr -s также может использоваться для преобразования последовательностей из множества заданных символов в один символ.

  мы используем параметр tr -d для удаления заданного символа.

- **wc** — Подсчет слов, строк и символов в файле осуществляется достаточно просто в случае использования фильтра wc.

- **sort** — Фильтр sort по умолчанию сортирует строки файла в алфавитном порядке.

- **uniq** — С помощью фильтра uniq вы можете удалить повторяющиеся строки из отсортированного списка строк.

- **comm** — Сравнение потоков данных (или файлов) может быть осуществлено с помощью фильтра comm. По умолчанию фильтр comm будет выводить данные в трех столбцах.

- **od** — создается простой файл, после чего для показа его содержимого в форме шестнадцатеричных значений байт используется фильтр od.

- **sed** — Фильтр sed (расшифровывается как stream editor - редактор потока данных) позволяет выполнять различные операции редактирования при обработке потока данных с задействованием регулярных выражений.

Подробнее про cat:

По сути, задача команды cat очень проста - она читает данные из файла или стандартного ввода и выводит их на экран.

$ cat опции файл1 файл2 ...

Опции:

- -b — нумеровать только непустые строки;
- -E — показывать символ $ в конце каждой строки;
- -n — нумеровать все строки;
- -s — удалять пустые повторяющиеся строки;
- -T — отображать табуляции в виде Î;
- -h — отобразить справку;
- -v — версия утилиты.
- Если вы не передадите никакого файла в параметрах утилите, то она будет пытаться читать данные из стандартного ввода:

Подробнее про grep:

Для поиска заданного шаблона в файле или группе файлов в системах UNIX и Linux используется инструмент командной строки grep.

Если команде grep не заданы никакие дополнительные опции, она осуществляет поиск заданного шаблона в файле (или файлах).

(Обратите внимание: когда в вашем шаблоне больше одного слова, его следует взять в одинарные или двойные кавычки.)

Опции:

- **-n (—line-number)** — вывод строк с их номерами
- **-c (—count) — вывод числа строк, в которых найдено соответствие с шаблоном**

  Обратите внимание: если бы в одной строке было найдено два совпадения с шаблоном, в выводе все равно было бы 2. Команда подсчитывает число строк, а не число вхождений.
- **-v (—invert-match) — выводит строки, в которых не найден искомый шаблон**
- **-i (—ignore-case) — инструкция игнорировать регистр**
- **-l (—files-with-matches) — вывод не строк, содержащих искомый шаблон, а имен файлов, в которых есть такие строки**
- **-w (—word-regexp) — поиск предполагает, что заданный шаблон должен быть целым отдельным словом**
- **-o (—only-matching) — вывод только найденного совпадения с шаблоном**
- **-A (—after-context) and -B (—before-context) — вывод строк, идущих перед или после строки с искомым словом**

- **Есть еще опия -C (–context), эквивалентная одновременному использованию опций -A и -B. Число, указанное при опции -C, будет касаться количества строк и до, и после вхождения. (T.e., если вы укажете grep you grep.txt -C 5, это будет эквивалентно grep you grep.txt -A 5 -B 5. — Прим. ред. Techrocks).**

- **-R (—dereference-recursive) — рекурсивный поиск**

  По умолчанию grep не ищет в шаблон в директориях. При попытке сделать это вы получите ошибку «Is a directory». Опция -R делает возможным поиск в директориях и вложенных директориях.

Надо расписать утилиты cat и grep.

Команда cat >> test << EOF

Символы конца в файл не запишуться.(EOF)

Переписать из билетов про регулярки и особенно про регулярки команды man

Для лабы ls *_*/*_*/*

Команды по которым будут спрашивать:

Команда	Назначение и синтаксис
mkdir	mkdir [-m mode] [-p] dir...
echo	echo [string]...
cat	cat [-n] [file...] [-]
touch	touch [-am]... file...
ls	ls [options] [file/dir]...
pwd	pwd
cd	cd [argument]
more	more [file...]
cp	cp [options] SOURCE ... DEST
rm	rm [options] [file/dir]
rmdir	rmdir [dir]
mv	mv [-fi] SOURCE ... DEST
head	head [-num] [file...]
tail	tail [-/+num] [-bcl] [file...]
sort	sort [-unr] [-k num] [file...]
grep	grep [-v] regexp [file...]
wc	wc [-c | -m] [-lw] [file...]

Если ты написал команду, то будь готов отвечать и по команде и по ключам.

Расписать ключи у команд выше + команда yes

Про YES: если нас просят много соглашаться, то через pipe запускаем эту команду и на выводит то что мы указали в аргументе подряд или у без аргумента если.

Команды:

0.0.1 mkdir

Назначение: создание новых директорий

Синтаксис: mkdir опции директория(одна или несколько директорий через пробел, которые требуется создать.)

Опции:

- -m или -−m — Устанавливает права доступа для создаваемой директории.
- -p or –parents — Создать все директории, которые указаны внутри пути. Если какая-либо директория существует, то предупреждение об этом не выводится.
- -v or –verbose — Выводить сообщение о каждой создаваемой директории.
- -Z — Установить контекст SELinux для создаваемой директории по умолчанию.
- –contex[=CTX] — Установить контекст SELinux для создаваемой директории в значение CTX
- –help — Показать справку по команде mkdir
- –version — Показать версию утилиты mkdir

0.0.2 echo

Назначение: Это не системная утилита, у нее нет исполняемого файла.

Синтаксис: $ echo опции строка

Опции:

- -n — не выводить перевод строки
- -e — включить поддержку вывода Escape последовательностей
- -E — отключить интерпретацию Escape последовательностей

Это все опции, если включена опция -e, то вы можете использовать такие Escape последовательности для вставки специальных символов:

- /c — удалить перевод строки
- /t — горизонтальная табуляция
- /v — вертикальная табуляция
- /b — удалить предыдущий символ
- /n — перевод строки
- /r — символ возврата каретки в начало строки

  С помощью последовательности /r можно удалить все символы до начала строки!

echo можно использовать для редактирования конфигурационных файлов. Вы можете использовать запись echo в файл linux, если он пуст:

$$\text{echo } 1 > /\text{proc/sys/net/ipv4/ip\_forward}$$

Если строка содержит какие-либо специальные символы или может быть понята интерпретатором неоднозначно, следует заключить ее в кавычки.

0.0.3 cat

Назначение: можно очень просто посмотреть содержимое небольшого файла, склеить несколько файлов и многое другое.

Синтаксис: $ cat опции файл1 файл2 ...

Опции:

- **-b** — нумеровать только непустые строки;
- **-E** — показывать символ $ в конце каждой строки;
- **-n** — нумеровать все строки;
- **-s** — удалять пустые повторяющиеся строки;
- **-T** — отображать табуляции в виде Î;
- **-h** — отобразить справку;
- **-v** - версия утилиты.