


Обзор IT-системы и Linux: Основы работы с командной строкой



План занятия

1. [Виртуальная файловая система Linux и структура каталогов](#)
2. [Немного о пользователях и доступах](#)
3. [Интерфейс командной строки](#)
4. [Итоги](#)
5. [Домашнее задание](#)



Виртуальная файловая система Linux и структура каталогов



VFS: Виртуальная файловая система

Виртуальная файловая система — это уровень абстракции над реализациями файловых систем, удобное представление их для процессов.

По сути это **дерево каталогов**, единая иерархия, в которой процессы могут обращаться к данным, хранящимся на диске, в памяти, на удалённых серверах и т.д.



VFS: Виртуальная файловая система

В отличие от Windows, где каждое устройство представлено независимой иерархией (диск C:\, D:\ ...), VFS предполагает наличие **единой иерархии, начинающейся с корневого каталога (root, /)** и подключение (**монтирование**) файловых систем в каталогах внутри этой иерархии.

Вспомним один из принципов POSIX и LSB: **всё в системе представлено в виде файла**. Поэтому монтируются не только каталоги с данными, но и виртуальные представления устройств, процессов, свойств системы.

Иерархия каталогов Linux

```
/ ..... корневой каталог
|-- bin ..... исполняемые файлы
|-- lib ..... библиотеки
|-- etc ..... конфигурационные файлы
|-- root ..... домашний каталог пользователя root
|-- home ..... домашние каталоги пользователей
|   |-- linus
|   |-- patrick
|   `-- ian
|-- dev ..... представление в фс устройств
|-- proc ..... представление в фс процессов
|-- sys ..... представление в фс прочих системных
|               параметров и состояний
--.
```

Иерархия каталогов Linux

```
-- .
|-- tmp . . . . . временные файлы и директории
|-- usr . . . . . дополнительная иерархия
|   |-- bin . . . . . - исполняемые файлы
|   |-- lib . . . . . - библиотеки
|   `-- share . . . . . - ресурсы (локализация, документация и т.д.)
|-- var . . . . . различные изменяющиеся файлы
|   |-- log . . . . . - логи (журналы событий)
|   `-- cache . . . . . - кэш (временное хранилище данных)
`-- opt . . . . . самостоятельные иерархии каталогов
    `-- jdk-16.0.2 . . . . . - приложение с самостоятельной
                               иерархией каталогов
```



Относительные и абсолютные пути

У любого процесса (включая процесс командной строки) есть текущая (рабочая) директория. Путь до файла может быть абсолютным и относительным (относительно рабочего каталога). **Абсолютный путь всегда начинается с /.**



Особые директории и синонимы

Любая директория (даже корень /) всегда содержит две особые директории:

- текущая директория
- • директория уровнем выше

Кроме того, абсолютный путь домашних каталогов доступен по алиасам:

- ~ домашняя директория текущего пользователя
- ~user домашняя директория пользователя user

Особые директории и синонимы

Например, для рабочей директории `/var/tmp` и пользователя `ian`:

```
./opt    -> /var/tmp/opt  
-./opt   -> /var/opt  
~/opt    -> /home/ian/opt
```

Использование директорий `.` и `..` (но не `~`) допустимо и в абсолютных путях:

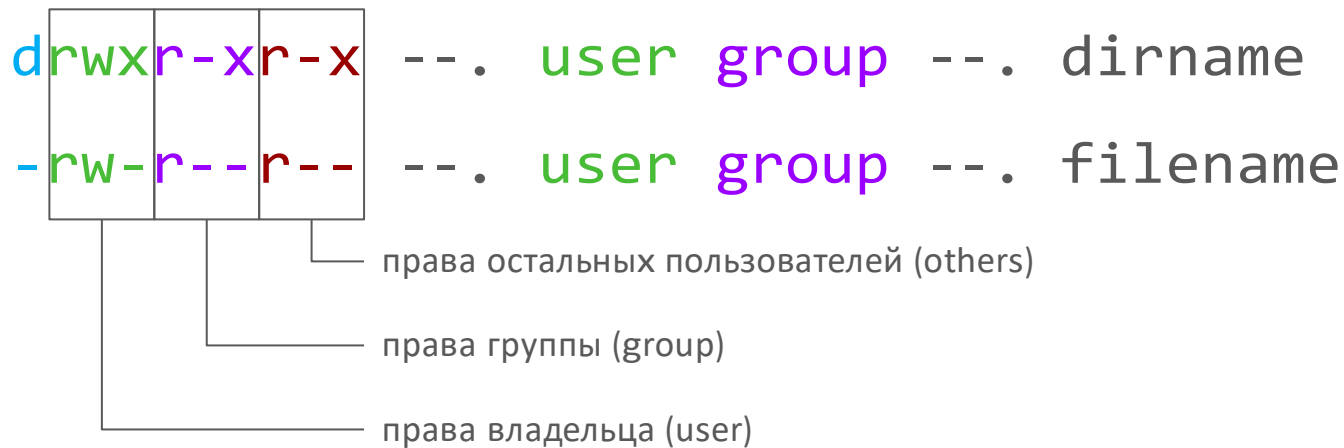
```
/var/tmp/./opt    -> /var/tmp/opt  
/var/tmp/-./opt   -> /var/opt  
/var/tmp/~opt    -> /var/tmp/~opt
```



Немного о пользователях и доступах

Права доступа

Файлы и директории принадлежат пользователю и группе, а права на доступ к файлам устанавливаются для владельца, членов группы и всех остальных пользователей:



r - Read	чтение файлов, листинг содержимого каталогов.
w - Write	запись в файлов, создание файлов в каталогах.
x - eXecute	исполнение файлов, переход в каталоги.

Пользователь root и повышение привилегий

Все современные операционные системы работают в многопользовательском режиме, и Linux не исключение. В Linux существует специальный пользователь-администратор (суперпользователь) — **root**.

Ряд вещей (но **далеко не все!**) необходимо осуществлять от имени суперпользователя. Для этого применяются два инструмента:

sudo более современный и гибкий, но по умолчанию установлен не везде.

su - более универсально доступный, но менее удобный.



Интерфейс командной строки

CLI: интерфейс командной строки

Существует два вида интерфейсов:

- Graphic User Interface (GUI), графический интерфейс
- Command Line Interface (CLI), **командная строка** (или **консоль**).

Преимущества командной строки:

- информативность,
- гибкость,
- примитивность,
- надёжность,
- экономия системных ресурсов.

Синтаксис команды

Синтаксис большинства команд **включает** в себя саму **команду** и **аргументы**, обычно аргументы делят на две категории: **опции** (их ещё называют **ключами**) и собственно **аргументы**. Опции модифицируют поведение команды, являются своего рода настройками и начинаются с **-** (короткой черты). Аргументы обычно указывают на цели, с которыми команда оперирует. Например:

```
ls -lh -d --all . /tmp /opt
```

ls	команда
-lh -d --all	опции (обратите внимание, что они могут «склеиваться» или быть «длинными»)
. /tmp /opt	аргументы (цели)

Автодополнение

Набирать текст команд и ошибок полностью - прямой путь к подобным ошибкам:

```
command not found  
no such file or directory
```

Достаточно ввести несколько символов и пару раз нажать клавишу “Tab”, чтобы произошло автодополнение **команды** или **имени директории или файла**:

```
$ ec <Tab><Tab>  
$ echo
```

Если вхождений несколько, вместо полного дополнения будут предложены варианты:

```
$ cd /etc/sysc <Tab><Tab>  
$ cd /etc/sysctl.  
sysctl.conf  sysctl.d/
```

Встроенная документация

- `man команда` — открыть документацию по команде
 - как читать:
 - [аргументы] в квадратных скобках **опциональны**,
 - аргументы без квадратных скобок **обязательны**,
 - `-o`, `--option` — взаимозаменяемые короткие / длинные опции;
- `команда --help`;
- `команда -h`
 - есть не для всех команд (а у некоторых вызывается иначе)
 - нет единого стандарта оформления.

Работа с текстом

- **echo** *текст* — вывести текст в консоль;
- **cat** *[цель]* — вывести содержимое файла (цели) в консоль;
- **head** *[цель]* — показать **первые** 10 строк файла
 - -15 — в качестве опции можно передать количество строк;
- **tail** *[цель]* — показать **последние** 10 строк файла
 - -15 — в качестве опции можно передать количество строк;
- **grep** *условие [цель]* — вывести строки, содержащие *условие*;
- **less** *[цель]* — интерактивный просмотр текста.

Стандартные потоки ввода-вывода

Поток номер 0 (STDIN) зарезервирован для *чтения команд* пользователя или *входных данных*.

```
command < filename
```

Поток номер 1 (STDOUT) зарезервирован для *вывода данных*, как правило текстовых.

```
command > filename    или  command >> filename
```

Поток номер 2 (STDERR) зарезервирован для вывода *диагностических и отладочных сообщений* в текстовом виде.

```
command 2> filename    или  command 2>> filename
```

В случае с потоками 1 и 2 один символ **>** означает **перезапись**, удвоенный **>>** — **дозапись**.

Конвейер (пайп)

Поскольку у любого процесса есть стандартные потоки ввода и вывода, их можно **перенаправлять** один в другой (вывод одной команды на ввод другой). Делается это с помощью, так называемого, **пайпа** или **конвейера**, в синтаксисе командной строки обычно представленного в виде **одной вертикальной черты** |:

```
command1 | command2
```

На слайдах с примерами команд в ряде инструментов цель помечена как необязательная, это потому, что многие команды для работы с текстом способны принимать его на стандартный ввод, например:

```
ls -lh /var/log | grep root
```

Работа с файлами и каталогами

- **pwd** — вывести рабочую директорию;
- **cd [цель]** — перейти в каталог
 - если не указывать цель— в домашний каталог пользователя;
- **mkdir цель** — создать каталог;
- **ls [цель]** — просмотр содержимого каталога
 - если не указывать цель — просмотр содержимого рабочего каталога
 - **-l** — длинный (подробный) вывод
 - **-a** — отображение скрытых (начинающихся с точки) файлов и каталогов
 - **-h** — «человекочитаемый» вывод размеров файлов;

Работа с файлами и каталогами

- **cp** *ист[ист] цель* — копирует *источник* в *цель* (если источников несколько, то *цель* должна быть каталогом)
 - -R — рекурсивно (если в качестве источника каталог — копировать его и его содержимое)
 - -v — вывести подробности о выполняемых операциях;
- **mv** *ист[ист] цель* — перемещает исходный файл в целевой
 - -v — вывести подробности о выполняемых операциях;

Работа с файлами и каталогами

- **rm** *цель* — удалить файл
 - -R — рекурсивно (если нужно удалить каталог вместе с содержимым)
 - -v — вывести подробности о выполняемых операциях;
- **rmdir** *цель* — удалить пустую директорию
 - -v — вывести подробности о выполняемых операциях;

Работа с пакетным менеджером

- **yum** — пакетный менеджер для RPM (RedHat, Oracle Linux и т.д.)

Стандартные команды:

- **yum clean all** — очистка данных о репо
- **yum list available <packet>** — проверка доступных пакетов
- **yum install *цель*** — установить пакет
- **yum update *цель* [*цель*]** — обновить пакет
- **yum list installed | grep *цель* [*цель*]** — вывести список установленных пакетов с grep по конкретному пакету
- **yum downgrade *цель* [*цель*]** — даун грейд пакета
- **yum remove *цель* [*цель*]** — удалить пакет

Переменная окружения \$PATH

Любой запускаемый процесс в Linux имеет доступ к, так называемым, переменным окружения. Подробнее о них в последующих лекциях, но о \$PATH стоит узнать уже сейчас.

Она содержит все директории, где интерпретатор командной строки ищет команды. Типичное значение \$PATH:

```
PATH=/usr/local/bin:/usr/bin:/bin
```

Поэтому `/usr/bin/ls` можно запустить как `ls`, а `mycmd` из `/opt/mycmd/bin/mycmd` нужно либо запускать с указанием полного или относительного пути, либо добавлять в \$PATH:

```
PATH=/usr/local/bin:/usr/bin:/bin:/opt/mycmd/bin
```



Итоги

Итоги

Сегодня мы узнали:

- что такое виртуальная файловая система и как выглядит структура директорий в linux;
- в чём различия между абсолютным и относительным путём;
- как обращаться к текущей директории и переходить на уровень выше;
- чем хороша командная строка и какой функционал она предоставляет системному администратору
- какие инструменты у нас есть для манипуляций файлами и текстом в командной строке;
- кто такой root, и что он может.

