

Администрирование операционной системы Linux

Лекция 3. Работа с файлами и каталогами.

Старший преподаватель, к.т.н. Митричев Иван Игоревич

Москва 2020

План лекции

- Система файлов и каталогов.
- Получение списков файлов и каталогов.
- Перемещение по дереву каталогов.
- Создание и удаление файлов и каталогов.
- Копирование, перемещение и переименование файлов.
- Перенаправление ввода-вывода. Конвейер.
- Поиск файлов.
- Определение типа файлов.
- Жесткие связи и символические ссылки.
- Команда `grep`. Регулярные выражения.

Получение списков файлов и каталогов

`pwd`



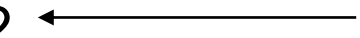
- вывести имя текущего каталога;

`ls -la`



- вывести подробную информацию и скрытые файлы;

итого 502



- число 512-байтных блоков, занятых файлами директории

drwxr-xr-x 2 user user 4096 ноя 28 2016 Музыка

<тип><права доступа> <владелец> <размер><дата изменения><имя>
<кол-во имен (жестких ссылок)> <первичная группа владельца>

`ls -F`



- выводит после имен каталогов /, после имен исполняемых файлов *, после символических ссылок @;

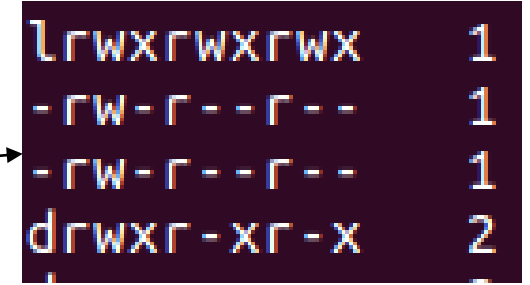
`ls -ld`



- вывести подробную информацию только о каталогах

Типы файлов в Linux

В выводе команды `ls -l` — первый столбец



```
l rwxrwxrwx 1  
- rw-r--r-- 1  
- rw-r--r-- 1  
drwxr-xr-x 2
```

Обозначение	Пояснение
—	обычные файлы
d	каталоги
l	символические ссылки (указатель на другой файл)
b	блочные устройства (специальные файлы для обращения к устройствам, например, жесткому диску - /dev/sda)
c	символьные устройства (специальные файлы для ввода/вывода с устройств, таких как терминал или мышь)
p	именованный канал (для взаимодействия между процессами)
s	сокеты (для организации межпроцессного взаимодействия по сети)

Создание файлов

> filename

← Создать пустой файл (путем переадресации стандартного потока вывода, в который ничего не подано, в файл)

>> filename

← Создать файл, но не перезаписывать, если файл существует

touch filename

← Создание файлов (можно перечислить несколько имен через пробел), или изменение даты для существующего файла.

Пример:

```
date > f1; cat f1; ls -l f*; sleep 60; touch f1 f2; date; cat f1; ls -l f*
```

date > f1 – создает файл f1, содержащий текущую дату;

cat f1 – выводит содержимое этого файла;

ls -l f* – выводит подробную информацию о файле f1;

sleep 60 – задерживает ход выполнения последующих команд на 60 секунд;

touch f1 f2 – изменяет дату модификации файла f1 и создает новый пустой файл f2;

date – выводит текущее время;

cat f1 – снова выводит содержимое файла f1, оно не изменилось;

ls -l – показывает, что даты модификации файлов f1 и f2 соответствуют времени выполнения команды touch.

Перемещение по дереву каталогов

cd ~



- переместиться в домашний каталог;

cd <каталог>



- переместиться в заданный каталог;

cd -



- переместиться в предыдущий каталог.

home/user – домашний каталог пользователя user

~ – короткое имя для домашнего каталога.

. – текущий каталог;

.. – родительский каталог.

Удаление файлов

`rm <filename(s)>` ←• удаление файлов и директорий.

Опции:

`f` – не спрашивать подтверждения; `i` – спрашивать подтверждения;

```
> rm -fi ghhg  
rm: удалить пустой обычный файл 'ghhg'? █
```

`r` – рекурсивно удаляет директории и их содержимое.

`rm -rf` — использовать осторожно, особенно- с удалением по шаблону.

Можно использовать перечисление при создании. Обязательно проверяйте с помощью `ls`, что будете удалять, перед удалением по шаблону (* и другие шаблоны):

```
touch f{1,2,3}  
ls f*[1,2]  
rm f*[1,2]
```

Создание и удаление каталогов

```
mkdir <dir>
```

```
mkdir -p dir1/dir2/dir3
```

• создание каталога;

• создание дерева (опция -p – ветвь вложенных каталогов);

```
rmdir <dir>
```

```
rmdir -p
```

• удаление пустого каталога;

• удаление дерева пустых каталогов.

Пример:

```
mkdir -p d1/d2/d3
```

```
$ ls -R d1
```

```
d1:
```

```
d2
```

```
d1/d2:
```

```
d3
```

```
d1/d2/d3:
```


Копирование, перемещение и переименование файлов

```
cp <source> <destination>
```

← • копирование;

```
cp <file1> <file2> ... <fileN> <dir>
```

← • копирование файлов в каталог dir;

```
cp dir/f{1,2}
```

← • копирование с перебором (плохой стиль), эквивалентно команде `cp dir/f1 dir/f2`, создается копия файла `f1` с именем `f2` в каталоге `dir`;

```
cp -R <dir1> <dir2>
```

← • рекурсивное копирование; аналог: `cp -r`

```
mv <source> <destination>
```

← • перемещение (переименование).

```
mv <file1> <file2> ... <fileN> <dir>
```

Работает и для файлов, и для директорий.

Команда mv для директорий

```
$ ls
mydir
$ cp -r mydir/ /tmp/
$ ls -R /tmp/mydir
/tmp/mydir:
f1 f2
```

Пример:

Скопировать каталог mydir со всем содержимым в каталог /tmp.

Команда mv используется для перемещения и переименования:

mv oldName newName – переименовывает oldName в newName;

mv file1 file2 fileN dir – перемещает заданные файлы в каталог dir;

mv file1 file2 fileN dir/ - лучше, наглядней (ошибка, если dir не существует)

mv oldName newName – переименовывает каталог oldName в newName.

Перенаправление ввода/вывода, передача результата выполнения одной команды другой

> - перенаправление вывода

< - перенаправление ввода

```
test@test:/tmp/t$ echo "77" > f4; cat f4  
77  
test@test:/tmp/t$
```

| - передача результата выполнения команды (pipe, или, конвейер)

```
test@test:/tmp/t$ echo "5.8-3.8" | bc  
2.0
```

Установка значений переменных

Как установить переменные? Используйте конвейер

```
test@test:/tmp/t$var1=`echo "5.8-3.8" | bc`  
test@test:/tmp/t$echo $var1  
2.0
```

```
test@test:/tmp/t$var1=$(echo "5.8-3.8" | bc)  
test@test:/tmp/t$echo $var1  
2.0
```

Конвейер приводит к запуску дополнительной дочерней оболочки (subshell). Избегайте частого создания оболочек в критичных к времени исполнения приложениях.

Использование cat и конвейера

```
cat 1.txt | grep -i '^Begin'
```

Используйте вместо этого

```
grep -i '^Begin' 1.txt
```

ИЛИ

```
grep -i '^Begin' < 1.txt
```

или, даже (нетрадиционный синтаксис)

```
< 1.txt grep -i '^Begin'
```

Особенности
переадресации
ввода/вывода:

```
echo "kl" > 1.txt 2.txt
```

```
test@test:/tmp/t$cat 1.txt  
kl 2.txt
```

Поиск файлов

```
find <места_поиска> <критерии> <модификаторы>
```

места_поиска – каталоги, начиная с которых будет осуществлен поиск (поиск также производится и в подкаталогах);

критерии – любые атрибуты файла (имя файла, размер, владелец файла, тип, даты доступа и изменения и пр.

модификаторы – как искать и выводить найденную информацию.

Поиск ведется в названиях файлов/директорий. Для поиска по файлам используйте `grep` (см. далее)

Пример: в домашней директории найти файлы `f1` и `f2`

```
$ find ~ -name f[1,2]
./d1/f1
./d1/f2
```

Пример: найти все файлы, начинающиеся на `f`, в домашней директории (кавычки обязательны)

```
find -name 'f*'
find -name "f*"
```

Экранирование (поиск файла «`f*`»)

```
find -name 'f\*'
find -name "f\*"
```

Критерии поиска

name – поиск по имени/шаблону;
iname – поиск по имени без учета регистра;
type – поиск по типу;
size – поиск по размеру;
empty – поиск только пустых файлов;
mtime – поиск по дате модификации;
atime — поиск по дате обращения;
ctime — поиск по дате создания;
perm – поиск по правам доступа;
user и group – поиск по принадлежности
пользователю и группе.

Пример: найти все файлы в ~/orca, которые
разрешено считывать, изменять и
исполнять всем пользователям и группам

```
find ~/orca -perm 777
```

Типы файлов:

d – каталог;
f – обычный файл;
l – символическая ссылка.

Пример: найти все символические ссылки в ~/orca

```
find ~/orca -type l
```

Модификаторы численных аргументов

К численным аргументам команды **find** применяются следующие модификаторы:

+n — более, чем n

-n — менее, чем n

n — ровно n

Пример: найти все файлы/директории в /tmp/t, доступ к которым последний раз осуществлялся

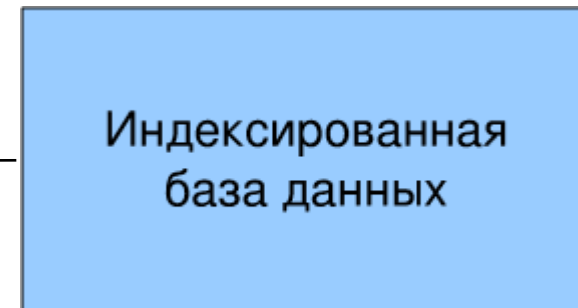
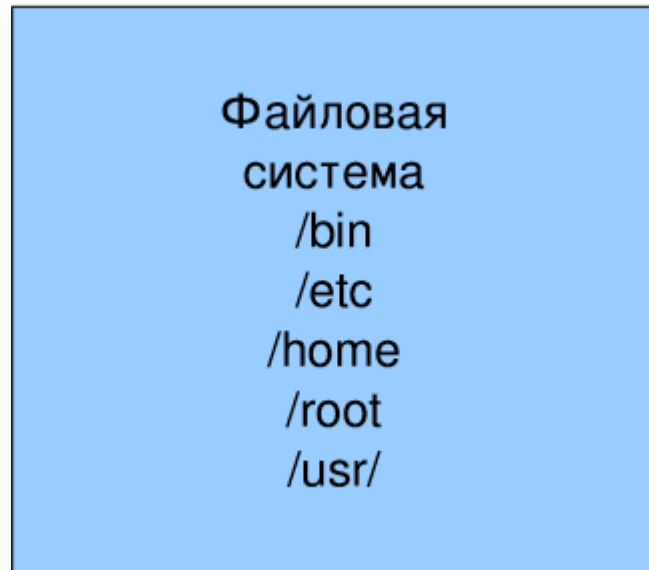
1) 17 минут назад, 2) 18 минут назад, 3) меньше 18 минут назад.

```
test@test:/tmp$find /tmp/t -amin 17
/tmp/t/g
/tmp/t/g/f45few
test@test:/tmp$find /tmp/t -amin 18
test@test:/tmp$find /tmp/t -amin -18
/tmp/t
/tmp/t/g
/tmp/t/g/f45few
```


Поиск файлов по индексу командой locate

```
locate <подстрока имени файла>
```

Поиск по индексу. Не воспринимает файловые шаблоны поиска, а только строки. Имеются версии команды, работающие с регулярными выражениями.



```
updatedb
```

Обычно выполняется ночью, автоматически.

Для любознательных: `cat /etc/crontab`,
`cat /etc/cron.daily/mlocate`, прочитать о службе cron
(будет в следующих лекциях)

Создайте новый файл, попытайтесь
найти его с помощью locate.
Получилось?

Некоторые опции поиска

```
find ~ -name "f*" -o -empty
```

- опция -o – два критерия, объединенные логическим ИЛИ.

Пример:

```
find ./d1 -name "f[1,2]" -o -empty
./d1/d2/d3
./d1/f1
./d1/f2
```

Пример: поиск и удаление всех файлов, имена которых содержат в себе строку "lost".

```
find ~ -name "*lost*" -exec mv {} ~ \;
```

- ← опция exec – выполнение команд с найденными файлами.

{ } \; – вместо фигурных скобок будут подставлены имена всех найденных файлов будут заменены именами найденных файлов, они будут переданы в качестве аргументов командам, вызванным опцией -exec. Конструкция \; применяется для указания конца опции.

Поиск информации о системных программах

```
which <что_ищем>
```

- выводит путь к заданной команде;

```
whereis <что_ищем>
```

- команда UNIX и UNIX-подобных операционных систем, которая выводит информацию о расположении файлов определенного приложения, указывает путь к документации и исходникам (поиск файла в системных каталогах).

Пример:

```
$which man  
/usr/bin/man
```

```
$whereis man  
man: /usr/bin/man /usr/local/man /usr/share/man /usr/share/man/man7/man.7.gz /usr/share/man/man1/man.1.gz
```

Определение типа файлов

file - определяет тип файла по содержимому. Использует магические числа (стандартные сигнатуры файлов).

```
file /etc/passwd  
file /bin/ls
```

```
file f3 ; file d1  
f3: empty  
d1: directory
```

/usr/share/file/magic.mgc – база сигнатур файлов

Список распространенных сигнатур с пояснениями есть на сайте https://en.wikipedia.org/wiki/List_of_file_signatures

Дескрипторы и жесткие связи

Индексные дескрипторы (inode) или метаданные – содержат данные о файлах.

По стандарту POSIX содержат

- размер файла в байтах;
- идентификатор владельца и группы владельцев файла;
- права доступа к файлу;
- идентификатор устройства, на котором хранится файл;
- счетчик имен (жестких ссылок) для файла;
- timestamps – даты доступа к файлу, его создания и изменения;
- указатели на блоки с данными файла.

```
test@test:/tmp/t$ls -li
итого 4
29394639 -rw-rw-r-- 2 root root 0 фев 18 22:43 f1
29394639 -rw-rw-r-- 2 root root 0 фев 18 22:43 f1000
```

Жесткая связь – это наличие у файла другого имени. Оба файла, указывают на те же самые метаданные (индексные дескрипторы, или, inode). Количество имен у файла отражается в третьем столбце в выводе команды `ls -l`.

Метаданные хранятся в таблицах inodes по всему диску (в каждой группе блоков — подробнее - в будущей лекции по файловым системам и дискам).

df --inodes — сколько свободных inodes осталось на в разделах.

Символические ссылки

`ln -s` ← создать символическую (мягкую) ссылку.

Ссылка мягкая, поскольку при удалении исходного файла ссылка более недействительна!

`ln` ← жесткая ссылка – это еще одно имя файла (ссылка на inode).

Пока остается хоть одно имя, файл доступен.

Удалить жесткую/мягкую ссылку можно:

- с помощью `rm`, как обычный файл;
- командой `unlink`.

```
$ln -s file file100
```

```
$ll file*
```

```
-rw-rw-r-- 1 user user 0 map 1 10:51 file
```

```
lrwxrwxrwx 1 user user 4 map 2 12:33 file100 -> file
```

```
$unlink file100
```

```
$ll file*
```

```
-rw-rw-r-- 1 user user 0 map 1 10:51 file
```

Определение свободного и занятого места на устройствах

df -h

– свободное место на блочном устройстве.

```
$df -h
Файл.система      Размер  Использовано  Дост  Использовано%  Смонтировано в
udev              1,9G    0             1,9G    0%             /dev
tmpfs             396M    6,4M         389M    2%             /run
/dev/sda5         459G    181G         255G    42%            /
```

`df /dev/sda` — информация о свободном месте на заданном устройстве

`df -h` — отображение в стандартных единицах (human readable format) — мегабайты, гигабайты, а не в 512-байтных блоках (по умолчанию, без опций).

`du -sh <имя_директории>` – определение размера директории.

`du -shc ~/*` - вывод информации по поддиректориям с общим итогом: `-s` — суммарный результат для каждая поддиректории, `-c` — общая сумма

К понятию о регулярных выражениях

Регулярные выражения – специализированная система поиска и обработки текста, основанная на сравнении части текста с образцами для поиска (шаблонами).



Шаблон – заменяет один или несколько символов.

Квантификатор – показывает **число вхождений символа** или группы символов, которая находится перед ним в регулярном выражении.

Таблица основных регулярных выражений

Метасимвол	Класс	Расширенный	Описание
^	Шаблон		Начало строки
\$	Шаблон		Конец строки
\<	Шаблон	*	Начало слова
\>	Шаблон	*	Конец слова
.	Шаблон		Любой символ, включая пробел
[]	Шаблон		Набор символов
()	Шаблон	*	Группировка символов
	Шаблон	*	Инфиксный оператор (или)
*	Квантификатор		Вхождение любое количество раз
+	Квантификатор	*	Вхождение не менее одного раза
?	Квантификатор	*	Вхождение не более одного раза
{n}	Квантификатор	*	Вхождение n раз
{n, }	Квантификатор	*	Вхождение не менее n раз
{n, m}	Квантификатор	*	Вхождение от n до m раз

Из книги Береснев А.Л. Администрирование GNU/Linux с нуля. — 2-е изд., перераб. и доп.
— СПб.: БХВ-Петербург, 2010. — 576 с

Команда `grep` — проверка текста на совпадение с образцом

Можно использовать стандартные регулярные выражения, команда `egrep` (`extended grep`) работает и с расширенными регулярными выражениями.

```
egrep -c '^.{1,3}: ' /etc/passwd
```

← Определить число пользователей, имя которых содержит не более 3 символов

Важные опции `grep`

- A *n* — вывести указанное число строк *n* после каждого совпадения с шаблоном поиска
- B *n* — вывести указанное число строк *n* до каждого совпадения с шаблоном поиска
- C *n* — вывести указанное число строк *n* до и после каждого совпадения с шаблоном поиска
- i — не учитывать регистр символов при поиске
- v — инвертировать результаты поиска — будут выведены все строки, НЕ содержащие совпадений
- c — считать только число совпадений, но не выводить совпадения
- include=`*.{c,h}` — включить в поиск только `.h` и `.c` файлы
- exclude=`*.{c,h}` — исключить из поиска файлы с расширением `.h` и `.c`
- r — рекурсивный поиск (используйте для поиска внутри файлов вместо `find`)
- l — показывать только имя файла, а не совпадающие строки

```
grep -rl "honey" ~
```

Шаблоны и квантификаторы

* – любое количество любых символов (кроме . в именах скрытых файлов).

Пример:

```
$ echo ??????  
Музыка Разное
```

– вывести список файлов (и директорий), в именах которых имеется ровно шесть СИМВОЛОВ.

[0-9] – шаблон для любой цифры

Перечисление вариантов:

```
$ echo .bash{rc,_profile}  
.bashrc .bash_profile
```