нетология

Программирование на Bash: **Полезные утилиты**

Данияр
Калиев

Данияр Калиев

Тимлид команды DevOps

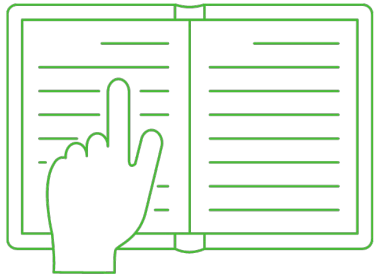
ТОО “Kazdream Technologies”

Данияр Калиев

Предисловие

На этом занятии мы рассмотрим основные **инструменты для работы с текстом и администрирования ОС Linux:**

- cat, more и less;
- head и tail;
- sort;
- split;
- wc;
- grep;
- locate;
- cut;
- find;
- sed;
- awk;
- vi/vim;
- nano.



План занятия

1. Предисловие
2. Работа с текстом
3. Текстовые редакторы
4. Итоги
5. Домашнее задание

Работа с текстом

cat, more и less

cat (concatenate) — команда cat читает содержимое одного или нескольких файлов и копирует его в стандартный вывод.

```
#  cat [файл...]
```

more и **less** — большой текстовый файл намного удобнее просматривать с помощью команд less или more.

Программа less удобнее, чем more , если она есть в вашей системе.

```
#  cat /var/log/messages | less
```

head и tail

head и **tail** — команда head выводит первые десять строк файла, а tail — последние десять.

Количество строк может регулироваться с помощью параметра -n.

```
#  head -n 10 /var/log/messages
```

```
#  tail -n 15 /var/log/messages
```

sort

sort — сортирует все указанные файлы, результат сортировки всех указанных файлов и перенаправленного отправляется на стандартный вывод.

```
#  sort [параметр]... [файлы]
```

sort

Параметр	Описание
-b	Пробелы в начале сортируемых полей или начале ключей будут игнорироваться.
-d	При сортировке будут игнорироваться все символы, кроме букв, цифр и пробельных символов.
-f	Игнорировать регистр букв.
-r	Сортировка в обратном порядке.
-o файл	Вывод результатов сортировки в файл.
-t символ	Использование указанного символа в качестве разделителя полей.

split

split — используется для разделения файлов на части.

По умолчанию создаются части размером в 1000 строк.

Изменить размер можно, указав количество строк, например:

```
#  split -500 файл1
```

wc

wc — подсчет слов в файле.

- для подсчета слов в текстовом файле:

```
#  wc /var/log/messages
```

- для подсчета слов в текстовом файле:

```
#  wc -l /var/log/messages
```

- для подсчета слов в текстовом файле:

```
#  wc -c /var/log/messages
```

grep

grep — текстовый фильтр.

Производит поиск строки в одном или нескольких файлах.

Если файлы не заданы, то программа читает текст из стандартного ввода.

locate

locate — простой способ поиска файлов

Программа locate выполняет быстрый поиск в базе данных имен файлов и выводит все имена, соответствующие искомой строке.

Допустим, например, что нужно найти все программы с именами, начинающимися с zip:

```
#  locate zip | grep bin
/bin/bunzip2
/bin/bzip2
/bin/bzip2recover
/bin/gunzip
/bin/gzip
/usr/bin/funzip
/usr/bin/gpg-zip
```

cut

cut — удаление фрагментов из всех строк в файлах.

cut используется для извлечения фрагментов текста из строк и вывода их в стандартный вывод.

Она может принимать имена файлов в аргументах или данные со стандартного ввода.

cut

Параметры команды cut для выбора фрагментов:

Параметр	Описание
-c	список символов: --characters= Извлекает фрагмент строки, определяемый списком символов. Список может включать один или несколько числовых диапазонов, разделенных запятыми.
-f	список полей: --fields= Извлекает одно или несколько полей из строки, как определено аргументом список символов. Список может включать одно или несколько полей или диапазонов полей, разделенных запятыми.

cut

Параметры команды cut для выбора фрагментов:

Параметр	Описание
-d	символ разделитель: --delimiter= В присутствии параметра -f, в качестве разделителя полей используется символ_разделитель. По умолчанию поля должны отделяться друг от друга одним символом табуляции --complement. Извлекает строку текста целиком, кроме фрагментов, определяемых параметром -c и/или -f.

cut

Если воспользоваться программой cat с параметром -A, можно увидеть, отвечает ли файл требованию в отношении использования символа табуляции в качестве разделителя полей.

```
#   cat -A distros.txt
Debian^I10.5^I01/08/2020$
Ubuntu^I20.04^I23/04/2020$
RHEL^I8.3^I03/11/2020^I$
CentOS^I8.3^I07/12/2020$
Fedora^I33^I10/27/2007$
openSUSE^I15.3^I10/04/2020$
```

cut

С помощью cut мы можем извлечь третий столбец:

```
#  cut -f 3 distros.txt
01/08/2020
23/04/2020
03/11/2020
07/12/2020
10/27/2007
10/04/2020
```

find

find — сложный способ поиска файлов.

В отличие от программы locate, выполняющей поиск файлов по именам, программа find ищет файлы согласно заданным атрибутам в указанном каталоге и во вложенных подкаталогах.

В простейшем случае программе find можно передать одно или несколько имен каталогов для поиска.

Например, с ее помощью можно получить список содержимого домашнего каталога:

```
#  find ~
```

find

Для большинства активных пользователей она выдаст длинный список. Так как список выводится в стандартный вывод, его можно передать по конвейеру другим программам.

Воспользуемся программой wc , чтобы подсчитать число файлов:

```
#  find ~ | wc -l
363168
```

find

Допустим, мы хотим получить список каталогов. Для этого добавим в команду следующую проверку:

```
#  find ~ -type d | wc -l
```

```
33440
```

Добавив проверку -type d, мы ограничились поиском только каталогов. Но точно так же можно ограничить поиск только обычными файлами:

```
#  find ~ -type f | wc -l
```

```
322504
```

find

Параметр	Описание
b	Специальный файл блочного устройства.
c	Специальный файл символьного устройства.
d	Каталог.
f	Обычный файл.
l	Символическая ссылка.

find

Добавив дополнительные проверки, можно выполнять поиск файлов по размеру и имени.

Давайте найдем все обычные файлы с именами, соответствующими шаблону *.JPG, и имеющие размер больше 1 мегабайта:

```
#  find ~ -type f -name "*.JPG" -size +1M | wc -l
15
```

find

Единицы измерения, поддерживаемые командой find:

b	Блоки размером по 512 байт (используется по умолчанию, если иное не указано явно).
c	Байты.
w	2-байтные слова.
k	Килобайты (Kilobytes, блоки по 1024 байт).
M	Мегабайты (Megabytes, блоки по 1 048 576 байт).
G	Гигабайты (Gigabytes, блоки по 1 073 741 824 байт).

sed

sed — потоковый текстовый редактор. Позволяет редактировать потоки данных на основе заданных правил.

С помощью sed можно провести простые операции по поиску и замене слов в тексте.

В общем случае синтаксис выглядит следующим образом:

sed ‘s/шаблон/замена/g’ file

Где:

- s — искать;
- шаблон — то, что ищем;
- замена — то, на что меняем текст;
- g — глобально, то есть во всём файле с именем file.

sed

Простой пример:

sed ‘s/test/text/g’ file найдёт все вхождения слова test в файле file и заменит на слово text, при этом результат работы выведет на экран, не изменяя основного файла.

```
#  cat file
test
hello test
#  sed 's/test/text/g' file
text
hello text
```

awk

awk — более мощная, чем sed, утилита для обработки потока данных.

С точки зрения awk данные разбиваются на наборы полей, то есть наборы символов, разделённых разделителем.

В awk используются переменные трёх типов:

- числовые (x=5);
- строковые (x=abc);
- переменные поля, которые обозначаются $1, $2 и т. д.

В отличие от скрипта bash, они означают номера полей, на которые разбита строка.

awk

awk можно использовать как самостоятельный язык для написания сценариев или вызывать его из командной строки для обработки потока данных.

Вызов происходит следующим образом:

поток_данных | awk '{скрипт_обработки_данных}'.

Здесь поток_данных — любая команда ОС или скрипт, результат работы которого будем передавать через pipe (|) на обработку awk.

```
#  ls -la | awk '{print $1 }'
```

Текстовые редакторы

vi/vim

vi/vim — текстовый редактор.

Редактор vi может работать в трех режимах:

- основной (визуальный) режим — в нем и осуществляется редактирование текста;
- командный режим — в нем выполняется ввод специальных команд для работы с текстом (если сравнивать vi с нормальным редактором, то этот режим ассоциируется с меню редактора, где есть команды вроде «сохранить», «выйти» и т. д.);
- режим просмотра — предназначен только для просмотра файла (если надумаете использовать этот режим, лучше вспомните про команду less).

vi/vim

После запуска редактора вы можете переключать режимы (как — будет сказано позже), но выбрать режим можно и при запуске редактора:

```
#  vi файл
```

```
#  vi -e файл
```

```
#  vi -R файл
```

Команды vi/vim

:q!	Выход без сохранения.
:w	Сохранить изменения.
:w <файл>	Сохранить изменения под именем <файл>.
:wq	Сохранить и выйти.
:q	Выйти, если нет изменений.
i	Перейти в режим вставки символов в позицию курсора.
a	Перейти в режим вставки символов в позицию после курсора.

Команды vi/vim

o	Вставить строку после текущей.
O	Вставить строку над текущей.
x	Удалить символ в позицию курсора.
dd	Удалить текущую строку.
u	Отменить последнее действие.

nano

nano — текстовый редактор

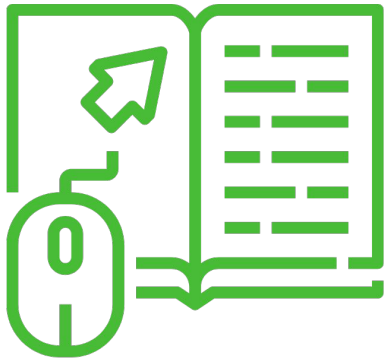
nano запускается аналогично:

```
#  nano <имя файла>
```

Итоги

Итоги

Сегодня мы познакомились с базовыми командами и текстовыми редакторами в ОС Linux.



Домашнее задание

Домашнее задание

Давайте посмотрим ваше домашнее задание.

Настоятельно рекомендуем вам выполнять ДЗ в том же ритме, что и просмотр лекций.

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Данияр Калиев

Данияр Калиев