Урок 8

Linux — базовые навыки работы в серверной операционной системе

Основные команды для выполнения операций в ОС Linux.

Команда sudo и особенности ее использования

Создание нового пользователя

Создание новой группы и добавление в нее пользователя

Работа с директориями и файлами

Команды man, info, whatis, --help

Права доступа к файлам в Linux

Категории пользователей файлов в Linux

Специальные права доступа к файлам в Linux

Определение прав доступа к файлам в Linux

Условное обозначение прав

Изменение прав на доступ к файлу

Установка и удаление пакетов в CentOS/Ubuntu

Создание общей директории всех пользователей в Linux

Получение списка всех активных служб и перезапуск любой из них

Просмотр открытых портов

Просмотр информации о запущенных процессах

Настройка виртуальной среды

Практическое задание

Дополнительные материалы

Используемая литература

Команда sudo и особенности ее использования

- ***Что такое sudo и каковы особенности использования данной команды?***

Утилита **sudo** (и соответствующая команда) позволяют выполнять административные операции под root-пользователем (суперпользователем). Это привилегированный пользователь ОС Linux, имеющий права на выполнение любых действий, удаление любых файлов и изменение любых параметров в системе. По умолчанию при установке одной из ОС семейства Linux — например, Ubuntu — первый пользователь (который создается при установке) получает полные права на использование **sudo**.

Изначально учетная запись суперпользователя в Ubuntu отключена для предотвращения случайных ошибок и несанкционированного входа в систему. Согласно документации Ubuntu, разблокировку прав суперпользователя следует проводить в крайних случаях. Если же это необходимо сделать, используется команда:

```
sudo passwd root
```

Ее необходимо ввести в терминале, после чего установить пароль суперпользователя. Его обязательно нужно сохранить где-то или записать.

Включение для первого пользователя прав суперпользователя позволит выполнять административные команды без утилиты **sudo**. Достаточно идентифицировать подключение в качестве суперпользователя с помощью команды **su** и сохраненного пароля суперпользователя:

```
su
```

Далее можно выполнять любые команды — например, создание нового пользователя и соответствующего пароля:

```
useradd имя_пользователя
passwd пароль
```

Если прислушаться к рекомендациям документации Ubuntu и не включать привилегии суперпользователя, то команду создания пользователя можно выполнить следующим образом:

```
sudo useradd имя_пользователя
```

То есть общий формат таков:

```
sudo <команда>
```

Если же необходимо выполнить под суперпользователем несколько команд, указывать каждый раз утилиту **sudo** будет трудоемко. В этом случае необходимо выполнить специальную команду для

временного назначения прав суперпользователя. При выполнении первой команды каталог меняется на **/root**, при второй — остается тем, который был до выполнения команды.

```
sudo -i
sudo -s
```

Создание нового пользователя

- ***Как создать нового пользователя и изменить его параметры?***

В первом разделе урока приведен один из способов создания нового пользователя:

```
sudo useradd имя_пользователя
sudo passwd пароль
```

Существует также альтернативный способ добавления нового пользователя в систему — с указанием не только имени пользователя, но и пароля:

```
sudo useradd -p пароль имя_пользователя
```

Чтобы увидеть изменения в списке пользователей системы, необходимо указать команду:

```
sed 's/:.*//' /etc/passwd
```

У команды **adduser** существует целый набор полезных опций, получить который можно с помощью команды:

```
sudo adduser --help
```

Удаление пользователя осуществляется с помощью утилиты **userdel**:

```
sudo userdel -f имя_пользователя

sudo userdel -r имя_пользователя
```

В первом случае удаляется пользователь, даже если он работает в системе. Во втором — каталог пользователя.

Создание новой группы и добавление в нее пользователя

- ***Как создать новую группу, изменить ее параметры и добавить в нее пользователя?***

Для создания новой группы в ОС Linux используется утилита **groupadd**. Формат использования данной утилиты:

```
sudo groupadd имя_группы
```

Чтобы изменить свойства группы — в частности, имя и пароль — применяется команда **groupmod** с флагом: **-n** для изменения имени группы или **-p** для изменения пароля группы. Пример:

```
sudo groupmod -n новое_имя_группы имя_группы
```

Для удаления группы необходимо выполнить в терминале следующую команду:

```
sudo groupdel имя_группы
```

Чтобы добавить пользователя в созданную группу, следует выполнить инструкцию:

```
sudo usermod -a -G имя_группы имя_пользователя
```

Для вывода списка групп, в которые входит указанный пользователь, предназначена команда:

```
sudo groups имя_пользователя
```

Например:

```
sudo groups new_user
```

Результат:

```
new_user : new_user new_group
```

Работа с директориями и файлами

- ***Как осуществляется переход между директориями? Как получить список файлов для текущей директории? Как удалить файл? Как создать новый каталог?***

Переход по директориям

Переход в домашний каталог (**/home**) пользователя:

```
cd
```

Переход в любой каталог системы:

```
cd путь_до_нужного_каталога
```

Переход в каталог на один уровень вверх:

```
cd ..
```

Переход в каталог на два уровня вверх:

```
cd ../..
```

Переход в каталог, в котором вы находились до перехода в текущий:

```
cd -
```

Работа с содержимым каталога

Полный путь до текущего каталога:

```
pwd
```

Вывести содержимое текущего каталога:

```
ls
```

Вывести дерево текущего каталога:

```
tree
# Это специальная утилита, для установки которой следует выполнить команду
sudo apt install tree
```

Работа с каталогами и файлами

Создание каталога:

```
mkdir имя_каталога
```

Удаление каталога:

```
rmdir имя_каталога
```

Копирование файлов из каталога **test_dir** в текущий каталог:

```
cp test_dir/*
```

Удаление файла:

```
rm имя_файла
```

Команды man, info, whatis, --help

- ***Чем различаются команды whatis, info, man, --help?***

whatis — позволяет в одной строке отобразить краткое описание программы, например:

```
whatis python
```

info — отвечает за вывод исчерпывающей информации о программе:

```
info python
```

man — выводит основную информацию о программе, включая описание и некоторые параметры:

```
man python
```

--help — выводит одну строку ответа на каждую поддерживаемую командой опцию, например:

```
python --help
```

Права доступа к файлам в Linux

- ***Какие основные права доступа к файлам реализованы в Linux?***

Базовый список параметров доступа для файлов включает:

- **Чтение**. Возможность получать содержимое файла без доступа к функции записи. При работе с каталогом означает возможность получить список вложенных каталогов и файлов.
- **Запись**. Доступ к функции записи новых данных в файл, к функции изменения существующих данных, а также к функции создания и изменения не только файлов, но и каталогов.
- **Выполнение**. Для системы идентификатором работы с файлом, как с программой, является наличие у файла флага выполнения. Он означает, что данный файл необходимо запускать, как программу.

Категории пользователей файлов в Linux

- ***Какие категории пользователей имеет каждый файл?***

Каждый файл в ОС Linux имеет три категории пользователей, для каждой из которых можно установить определенные сочетания прав доступа:

- **Владелец**. Понятие, характеризующее набор прав для владельца файла. Это пользователь, который его создал или установлен в качестве владельца. Обычно владелец обладает всей совокупностью прав для работы с файлом: чтение, запись, выполнение.
- **Группа**. Любая группа пользователей, действующая в системе и привязанная к файлу.
- **Остальные**. Все остальные пользователи, кроме владельца и пользователей, относящихся к группе файла.

На основе данных наборов полномочий устанавливаются права файлов в Linux. Каждый пользователь может получить доступ только к тем файлам, где он значится в качестве владельца, или к которым ему разрешен доступ. Только для суперпользователя предусмотрена возможность работы с любыми файлами независимо от связанных с ними наборами полномочий.

Специальные права доступа к файлам в Linux

- ***Какие существуют специальные права доступа к файлам в Linux?***

Со временем стандартных прав доступа к файлам оказалось недостаточно. Добавили еще несколько флагов, позволяющих устанавливать файлы как неизменяемые или запускать их от имени корневого пользователя без знания его пароля.

SUID. Это флаг, наличие которого у исполняемого файла позволяет непривилегированному пользователю запускать файл с правами владельца. При этом ID пользователя автоматически меняется на идентификатор пользователя-владельца файла.

SGID. Наличие этого флага у исполняемого файла означает, что непривилегированный пользователь может запускать исполняемый файл с правами группы этого файла. Идентификатор группы непривилегированного пользователя меняется на идентификатор группы файла.

Наличие флага **SGID** у директории означает, что каждый создаваемый в ней файл при запуске будет принимать идентификатор группы директории — а не группы владельца, который создал файл в этой директории.

Например, команду **passwd** может выполнить и обычный пользователь. Но если бы у соответствующего исполняемого файла отсутствовали флаги **SUID** и **SGID**, то это было бы невозможно. Ведь для изменения пароля необходим root-доступ к файлу **/etc/shadow** — а по умолчанию этого доступа у непривилегированного пользователя нет. Флаги **SUID** и **SGID** как раз и предоставляют root-доступ.

Sticky-bit. Флаг, используемый для создания общих каталогов. При его наличии пользователь может только создавать, читать и выполнять файлы, но не удалять те, что принадлежат другим пользователям.

Определение прав доступа к файлам в Linux

- ***Как узнать права доступа к файлам в Linux?***

Проверить права доступа к Linux-файлам можно в файловом менеджере, но полученная информация будет неполной. Для доступа к максимально подробным сведениям обо всех флагах необходимо обратиться к команде **ls** и выполнить ее с параметром **-l**. В результате будет сформирован список файлов каталога со всеми флагами, определяющими права доступа к файлам.

```
ls -l
```

```
-rw-rw-r-- 1 dima dima    0 Aug 30 04:30 __init__.py
-rw-rw-r-- 1 dima dima  129 Aug 30 04:32 __init__.pyc
-rw-rw-r-- 1 dima dima 3277 Aug 30 06:08 settings.py
-rw-rw-r-- 1 dima dima 2658 Aug 30 06:08 settings.pyc
-rw-rw-r-- 1 dima dima  935 Aug 30 06:42 urls.py
-rw-rw-r-- 1 dima dima 1034 Aug 30 06:45 urls.pyc
-rw-rw-r-- 1 dima dima  388 Aug 30 04:30 wsgi.py
-rw-rw-r-- 1 dima dima  583 Aug 30 05:55 wsgi.pyc
```

Права доступа к файлу отображаются в начале каждой строки. Первый флаг указывает на тип файла. В данном случае это обычный файл. Следующие три группы флагов (по три на каждую категорию пользователей) указывают на права доступа для владельца, группы и остальных пользователей. В данном примере и владелец, и пользователи группы имеют права на чтение, запись файла, но не на

выполнение. Все остальные — только на чтение. Рассмотрим еще несколько примеров комбинаций параметров, определяющих права на файл.

```
-rwxrwxrwx # файл доступен всем

-rwxr-xr-x # владелец может все, группа - только чтение и выполнение, остальные пользователи тоже только чтение и выполнение

--------- # нет права ни у одного пользователя

-rwx-wx--- # владелец может все, а группа - только изменение и выполнение, без просмотра содержимого файла

drwxr-xr-t # d-указание на тип объекта - каталог. Владелец может все, группа просматривать и выполнять, а для остальных пользователей указано, что они не могут удалить файл (флаг -t).

-rwsr-sr-x # для владельца установлены флаги SUID и SGID
```

Условное обозначение прав

- ***Какие условные обозначения соответствуют различным правам?***

Флаг	Описание права
---	Нет прав
--x	Разрешен только запуск файла на выполнение, без отображения содержимого и перезаписи
-w-	Разрешена только перезапись файла
-wx	Разрешена перезапись файла и его выполнение, но просмотр содержимого каталога недоступен
r--	Разрешено только чтение файла
r-x	Доступно только отображение содержимого и запуск файла на выполнение
rw-	Доступно отображение содержимого файла, перезапись, но без выполнения
rwx	Все права
--s	Установлен флаг **SUID** или **SGID**. Первый отображается в группе параметров для владельца, второй — для группы
--t	Установлен флаг **sticky-bit**, означающий запрет на удаление файла

Изменение прав на доступ к файлу

- ***Как изменить права на доступ к файлу?***

Рассмотрим пример простейшего текстового файла и проверим привязанные к этому файлу права доступа:

Для изменения прав на файл в Linux необходимо использовать утилиту **chmod**. Она позволяет изменять все флаги, в том числе специальные. Синтаксис утилиты:

```
chmod опции категориядействиефлаг файл
```

Параметр опции в данном случае является необязательным. Параметр «категория» указывает группу пользователей, для которой необходимо применять права:

u — владелец файла;

g — группа файла;

o — другие пользователи;

Действие может быть одним из двух: добавить флаг (‘+’) или убрать флаг (‘-’)

Флаг может иметь одно из следующих значений:

- **r** — чтение,
- **w** — запись,
- **x** — выполнение,
- **s** — **suid/sgid**, в зависимости от наименования комбинации параметров (для владельца или группы),
- **t** — **sticky-bit**.

Например, необходимо установить для представленного выше файла все права доступа для всех пользователей:

```
chmod ugo+rwx test.txt
```

Результат:

Еще примеры изменения права доступа к файлу:

```
chmod g+rx test.txt # дадим группе право на чтение и выполнение
chmod o+r test.txt # остальным пользователям - только на чтение
chmod u+s test1 # для файла test1 устанавливаем SUID
chmod g+s test2 # для файла test2 устанавливаем SGID
```

Установка и удаление пакетов в CentOS/Ubuntu

- ***Как устанавливать и удалять пакеты в CentOS/Ubuntu?***

CentOS

В дистрибутиве **CentOS** ОС Linux для операций с пакетами используется пакетный менеджер **YUM**, который разработан компанией **RedHat** для операций с пакетами в RPM-формате. Формат использования утилиты:

```
sudo yum опции команда имя_пакета
```

Перед непосредственной установкой CentOS-пакетов необходимо выполнить обновление списка репозиториев:

```
sudo yum update
```

Чтобы установить пакеты из официальных репозиториев, следует воспользоваться командой **install** с указанием названия приложения:

```
sudo yum install имя_приложение

например,

sudo yum install firefox
```

В процессе установки приложения от пользователя требуется нажать клавишу “**y**”. Эту опцию можно указать непосредственно в инструкции установки:

```
sudo yum -y install firefox
```

Об установленном пакете можно получить информацию:

```
sudo yum info firefox
```

Удаление пакета осуществляется с помощью команды **remove**:

```
sudo yum remove firefox
```

Для поиска определенного пакета может применяться **search**:

```
sudo yum search firefox
```

Ubuntu

Перед установкой пакетов в Ubuntu надо предварительно устанавливать зависимости. Это дополнительные пакеты, от которых зависит программа. В Ubuntu зависимости разрешаются с помощью пакетного менеджера (**Synaptic**, **apt**, Центр приложений, **apt-get**, **aptitude**), который автоматически установит зависимости из репозитория. Если нужная программа отсутствует в основном репозитории или репозитории недоступны, программу можно установить на основе deb-пакета.

Рассмотрим установку из репозитория:

1. **С помощью графического интерфейса**. Это **Synaptic** — простое визуальное программное средство для выбора устанавливаемых программ. Для запуска приложения необходимо выбрать: ***Система —Администрирование — Менеджер пакетов Synaptic***.

2. **С помощью командной строки**. Это ручной метод, позволяющий получить больше информации о процессе установки.

   Обновляем данные о доступных в репозиториях программах:

   ```
   sudo apt-get update
   ```

   Устанавливаем необходимое программное приложение:

   ```
   sudo apt-get install имя_программы
   ```

   Удаляем программу:

   ```
   sudo apt-get remove имя_программы
   ```

Рассмотрим установку с помощью deb-пакета:

Deb-пакет можно скачать с **packages.ubuntu.com**, **getdeb.net** или с сайта самой программы.

1. **С помощью графического интерфейса**. Необходимо с помощью файлового менеджера **Nautilus** перейти в папку с deb-пакетом, открыть свойства файла и на вкладке «Права» разрешить выполнение файла (установить галочку у параметра «Разрешить исполнение файла как программы»). Далее закрыть свойства файла, и по двойному щелчку **Nautilus** предложит просмотреть код или запустить файл на выполнение. Установить deb-пакет можно с помощью специального установщика **GDebi** (для него инструкция установки имеет вид **sudo apt-get install GDebi**).

2. **С помощью командной строки**. При этом используется программа **dpkg**. Файл **.deb** предварительно должен быть загружен.

```
sudo dpkg -i путь_к_deb_файлу
например,
sudo dpkg -i /home/user/soft/ntlmaps_0.9.9.0.1-10_all.deb
```

Рассмотрим механизм установки приложений в Ubuntu на примере интерпретаторов Python 2.7 и Python 3. В ОС Windows для установки Python достаточно загрузить установщик с официального сайта языка и запустить двойным щелчком мыши. В OS Ubuntu установку выполняют так:

```
sudo apt-get update
sudo apt-get install python2.7

или

sudo apt-get install python3
```

Для запуска Python-скрипта через терминал достаточно перейти в директорию скрипта, ввести команду вызова интерпретатора и указать имя файла-скрипта с расширением:

```
python2.7 run.py

или

python3 run.py
```

Чтобы проверить наличие установленной версии интерпретатора, необходимо ввести следующую команду:

```
python2.7

или

python3
```

Создание общей директории всех пользователей в Linux

- ***Как создать общую директорию для всех пользователей Linux?***

Общие папки в системе предназначены для хранения данных, доступных для каждого Linux-пользователя определенного компьютера. Каталоги, имеющие общий доступ, предназначены для работы над совместными проектами и обмена данными.

Рассмотрим пошаговый процесс создания общей папки:

1. Создание общей папки. Откроем терминал и введем команду:

```
sudo mkdir -p common_dir
```

2. Создание общей группы. Используем команду **groupadd**:

```
sudo groupadd share_group
```

3. Добавление пользователей, которые должны иметь права записи в директорию **common_dir**, в группу **share_group**:

```
sudo usermod -a -G имя_группы имя_пользователя

например,

sudo usermod -a -G share_group root
```

Флаги **-a** и **-G** означают добавление пользователя в дополнительную группу и указание имени группы. Проверим, добавлен ли пользователь в группу:

```
sudo groups root
```

Результат:

```
root : root share_group
```

4. Настройка необходимых разрешений на директорию. Опция **-R** означает выполнение рекурсивных операций во вложенных директориях:

```
sudo chgrp -R share_group common_dir
sudo chmod -R 2775 test_dir
```

5. Число 2775 означает: 2 — новые файлы всегда будут получать ту группу, которая стоит у папки, в которой они находятся; 7 — выдает все права (**rwx**) владельцу папки; 7 — выдает все права (**rwx**) группе; 5 — выдает право на чтение (**r-x**) другим пользователям.

6. Создаем нового пользователя и добавляем его в группу:

```
sudo useradd -m -c "new_user" -s/bin/bash -G share_group new_user
```

7. Создаем подкаталог для нового пользователя:

```
sudo mkdir -p common_dir/new_user_dir
```

Получение списка всех активных служб и перезапуск любой из них

- ***Как получить список всех активных служб и выполнить перезапуск любой из них?***

Службы — это особые фоновые приложения, контролирующие состояние системы. Они обеспечивают автоматическое подключение внешних устройств и сети, позволяют процессам взаимодействовать с оборудованием . В виде служб реализованы веб-серверы и серверы баз данных. В отличие от пользовательских приложений, службы выполняются в фоновом режиме, и пользователь не имеет к ним прямого доступа. Они запускаются еще до входа пользователя в систему.

Для управления службами в Linux применяется специальная утилита **systemctl**, которая обеспечивает перезапуск служб, проверку их состояния и анализ эффективности их загрузки. Чтобы получить список всех активных служб, необходимо выполнить следующую команду:

```
systemctl list-units --type service
```

Следующая команда отвечает за формирование списка служб **linux**, включая даже не запущенные, но известные утилите:

```
systemctl list-units --type service -all
```

Команда для сортировки служб списка по состоянию — например, только выполняющиеся:

```
systemctl list-units --type service --state running
```

Для запуска службы нужно воспользоваться командой **start**:

```
sudo systemctl start имя_службы.service
```

Расширение **service** указывать необязательно, оно устанавливается по умолчанию.

Останов службы можно выполнить следующим образом:

```
sudo systemctl stop имя_службы
```

Для просмотра состояния службы предназначена команда **status**:

```
sudo systemctl status имя_службы
```

Перезапуск службы:

```
sudo systemctl restart имя_службы
```

Просмотр открытых портов

- ***Как существуют способы просмотра открытых портов в Linux?***

Состояние открытия для порта означает, что он используется программой (например, сервисом) для связи с другой программой по сети или в локальной системе. В основном для просмотра открытых портов используют команду **netstat**. В результатах вывода отображаются все сервисы, а также прослушиваемые ими порты и ip-адреса. Альтернативные способы: использование утилиты **lsof** и команды **nmap**.

Способ 1. NETSTAT

```
sudo netstat -ntulp
```

Параметры команды:

- **-n** — отображать ip-адреса в числовом виде;
- **-t** — отобразить tcp-порты;
- **-u** — отобразить udp-порты;
- **-l** — отобразить только прослушиваемые порты;
- **-p** — вывести имя программы, а также ее PID.

Способ 2. LSOF

Данная утилита используется для сбора информации обо всех открытых в системе соединениях, в том числе сетевых:

```
sudo lsof -i
```

Еще один пример, в котором выполняется сканирование порта 80 и определение процессов, работающих с этим портом:

```
sudo lsof -i | grep 80
```

Способ 3. NMAP

Данная утилита представляет мощный сетевой сканер для удаленных узлов. Также ее можно использовать и применительно к локальному компьютеру:

```
nmap localhost
```

Просмотр информации о запущенных процессах

- ***Как получить информацию о запущенных процессах в Linux?***

В Linux основной командой, отвечающей за получение информации о состоянии процессов, является **ps**. В качестве ее дополнительных параметров могут быть установлены:

- **-l** — генерация листинга запущенных процессов в длинном формате;
- **-a** — вывод информации о наиболее часто запрашиваемых процессах.

Общая команда вывода информации о запущенных процессах будет выглядеть так:

```
ps -la
```

Эта команда отвечает за вывод основных сведений о процессах, запущенных текущим пользователем.

```
ps -ela
```

Команда вывода информации о запущенных процессах для всех пользователей. Здесь в команду добавляется флаг **-a**, который определяет получение информации по всем пользователям.

```
pstree
```

Еще один вариант получения списка запущенных процессов в формате дерева. Позволяет увидеть родительские и дочерние процессы.

Настройка виртуальной среды

- ***Что такое виртуальная среда для проекта, и как ее настроить?***

Виртуальная среда позволяет создавать зависимости отдельно для каждого Python-проекта. Для работы с виртуальными средами применяется инструмент **virtualenv**, который устанавливается с помощью команды:

```
pip install virtualenv
```

Создание виртуального окружения:

```
virtualenv virtualenvs/new_venv
```

Активация виртуального окружения:

```
source ./virtualenvs/new_venv/bin/activate
```

Установка зависимостей для проекта:

```
pip install -r requirements.txt
```

Деактивация виртуального окружения:

```
source ./virtualenvs/new_venv/bin/deactivate
```

Практическое задание

1. Создать общую директорию и общую группу, добавить в эту группу себя (текущего пользователя). Выполнить настройку прав доступа к каталогу для пользователей из группы. Разрешить проведение рекурсивных операций в подкаталогах.

2. Создать трех пользователей (с логинами и паролями) и добавить их в созданную группу. Проверить их присутствие там.

3. Создать в общей директории подкаталоги для каждого пользователя. Войти в систему под каждым пользователем и проверить доступ к общей папке. Создать в подкаталогах вложенные файлы, посмотреть результат.

4. Создать виртуальное окружение для Django-проектов. Создать файл зависимостей **requirements.txt** (с единственной зависимостью — самим фреймворком Django). Установить зависимости и протестировать работу Django-проектов из уроков 4 и 5 под созданным виртуальным окружением.

5. Получить информацию об активных службах, открытых портах, запущенных процессах.

6. Проверить особенности установки приложений, выполнив установку, например, Google Chromium, а затем удалив его.

Каждое из заданий необходимо выполнять в отдельном окне терминала. Когда задание выполнено и проверено вами, необходимо сделать скриншот экрана и сохранить окно терминала с выполненными командами для соответствующего задания. Таким образом, для каждого из шести заданий надо создать отдельный файл-изображение терминала с командами и сохранить в каталоге задания для этого урока. Далее этот каталог необходимо пометить в архив и отправить на проверку преподавателю.

Дополнительные материалы

1. В чем разница между sudo и su.
2. root и sudo.
3. Linux: добавить пользователя в группу.
4. Пользователи и группы.
5. Администратор в Ubuntu, или Что такое sudo.

Используемая литература

1. Вопросы по базовым командам Linux, которые вам могут задать на собеседовании.
2. Как посмотреть пользователей Ubuntu.
3. Команды linux для работы с файловой системой.
4. Типы файлов в Linux.
5. Права доступа и файловые флаги.
6. Общие папки Linux.
7. Как посмотреть открытые порты в Linux.
8. 10 приемов работы в терминале Linux, о которых мало кто знает.
9. Русскоязычная документация по Ubuntu.