Задание №2

Шаг 1. Команды Linux

Введите логин/пароль, полученный у преподавателя, для входа на консоль вашей виртуальной машины. После входа на консоль сперва разберемся с навигацией по каталогам (аналог папок в Windows) операционной системы Linux.

Выполните команду перехода в корневой каталог (это что-то вроде C:/ в Windows):

```
cd /
```

После выполнения команды вы перешли в корневой каталог. Выполните команду pwd, отображающую текущий каталог:

```
pwd
```

Вы увидите, что текущий каталог это / Следующая команда (ls) отображает список файлов и каталогов в текущем каталоге.

```
ls
```

Нам необходимо попасть в личный каталог пользователя. Каталоги пользователей находятся в каталоге home. Для перехода в каталог home выполним команду cd home, затем проверим в каком каталоге мы находимся (pwd), затем посмотрим список файлов и каталогов в каталоге home:

```
cd home
pwd
ls
```

Перейдем в каталог пользователя. У меня это bsk23-01-student-1: В командной строке можно набрать «cd bsk» и нажать клавишу tab. Система автоматически заполнит оставшееся название каталога. Затем проверим в каком каталоге мы находимся и посмотрим список файлов в каталоге:

```
cd bsk23-01-student-1
pwd
ls
```

С помощью команды mkdir создадим в личном каталоге пользователя новый каталог с именем python:

```
mkdir python
```

Проверим, что создали каталог командой ls, перейдем в новый каталог cd python, проверим, что находимся в новом каталоге pwd:

```
ls
cd python
pwd
```

Стрелки вверх/вниз – перемещение по истории команд (чтобы не набирать заново).

To Know: Другие базовые команды в Linux (uname, man, mkdir, touch, cat, nano, cp, mv, rm, df).

Шаг 2. Установка пакетов

Программы в Linux называются пакетами. apt - менеджер пакетов. Используется для установки новых пакетов, обновления и удаления пакетов в системе. Если запустить apt без параметров, то увидим справку по использованию apt:

```
apt
```

Запуск apt с параметром list отображает список всех возможных пакетов (и установленных и тех, которые можно установить). Вывод результата выполнения команды займет несколько секунд:

```
apt list
```

Для того, чтобы apt list отобразил не весь список можно передать вывод команде grep. Для этого после apt list нужно поставить знак | и далее написать grep с текстом который нужно найти в результатах выполнения apt list. Например apt list | grep installed покажет только строки в которых есть слово installed, т.е. установленные пакеты:

```
apt list | grep installed
```

Чтобы грепнуть (grep) по нескольким условиям выполним команду с ключом -E. Команда отобразит установленный пакет gzip:

```
apt list | grep -E 'gzip.*installed'
```

Если вывод команды не пустой и отображает строку «gzip/stable …» значит пакет gzip установлен.

Установим пакет mc (Midnight Commander) для удобства работы с файлами системы. Для этого выполним команду apt с параметром install и именем пакета mc. Система позволяет устанавливать новые пакеты только администратору системы. Ваш пользователь обладает правами запуска команд от имени администратора. Для этого в начале команду необходимо написать sudo:

```
sudo apt install mc
подтверждаем установку «Y»
```

Проверим, что установили пакет mc

```
apt list | grep -E 'mc.*installed'
```

Шаг 3. Midnight Commander

mc – визуальный файловый менеджер. Позволяет:
— копировать, перемещать и удалять файлы и целые деревья каталогов;
— искать файлы и запускать команды в подкоманде;
— использовать внутренний просмотрщик и редактор.

Запустите mc:

```
mc
```

Горячие клавиши mc:
F3 – просмотр файла
F4 – редактирование файла
Shift+F4 – создание нового файла
F7 – создание нового каталога
F8 – удаление файла или каталога
F10 – выход из mc

To Know: Другие базовые команды MC

Шаг 4. Первый скрипт Python

Создадим новый файл нажатием **Shift+F4**.

При первом редактировании MC спросит какой редактор использовать. Выберем редактор Nano. Также файл можно создать из консоли выполнив команду nano с именем файла (предварительно выйти из mc нажав F10).

```
nano hello.py
```

В созданном файле пишем:

```
print(“Hello!”)
```

Ctrl+X (сохранить и выйти) затем подтверждаем нажатием **Y**

Если вы редактируете существующий файл или создали файл из командной строки сразу указав имя то nano попросит подтвердить имя файла. Если вы создали файл из MC с помощью Shift+F4 то нужно ввести имя файла. Указываем или подтверждаем название файла «hello.py»

Если мы были в MC то нажмём **F10** для выхода из mc.

Запустим Python Shell без параметров:

```
python3
```

Видим короткую справку и приглашение для ввода команд Python »>. Выполним команду вывода на экран текста print:

```
print("text")
```

увидим результат выполнения команды: Текст выведен на экран.
Выйдем из Python Shell:

```
exit()
```

*так же для выхода можно использовать сочетание клавиш **Ctrl+D***

Теперь выполним скрипт Python из созданного нами файла hello.py:

```
python3 hello.py
```

Скрипт выполнил команду и написал нам:

```
Hello!
```

Теперь вы можете написать в резюме, что имеете опыт написания скрипта Python на виртуальном сервере Linux Debian 😎

Шаг 5. Простой цикл

Сделаем скрипт с простым циклом for. Создайте новый файл **for.py** (либо из MC, либо из консоли с помощью nano). В файле напишите:

```
for i in range(1, 10):
    print(i)
```

Цикл for в этом скрипте будем изменять значение переменной i в диапазоне от 1 до 10 и для каждого значения выполнять команду вывода на экран этого значения print(i).

Команда print(i) находится в блоке, выполняемом внутри цикла. Команды внутри блока написаны не с начала строки, а со смещением (это может быть пробелы или tab). В блоке цикла может быть несколько команд. Все они должны быть написаны со смещением.

Шаг 6. Чтение файла

Создайте новый файл с именем **file.txt**. Напишите в нем любые 5 или больше строк. Это будет

файл с входными данными для следующего скрипта. Скрипт прочитает данные из этого файла и отобразит их на экран.

Создайте новый файл скрипта Python с именем **fileread.py**

В скриптах Python можно писать комментарии. Всё, что написано после знака «#» считается комментарием

```
# открываем файл на чтение
f = open('file.txt','r')
# выводим содержимое файла на экран
print(*f)
# закрываем открытый файл
f.close
```

В этом скрипте мы открыли файл **file.txt** (функция open) на чтение (параметр 'r' в функции open) и вывели на экран всё содержимое файла (*f). Затем закрыли файл (функция close).

Шаг 7. Чтение одной строки файла

Вместо вывода всего файла считаем одну строку (функция readline) запишем её в переменную l и выведем её на экран (print(l))

```
# открываем файл на чтение
f = open('file.txt','r')
# читаем строку файла записываем в переменную l
l = f.readline()
# выводим переменную l на экран
print(l)
# закрываем открытый файл
f.close
```

Шаг 8. Чтение 3х строк из файла

Теперь считаем из файла 3 строки с помощью цикла for

```
# открываем файл на чтение
f = open('file.txt','r')
# запускаем цикл 3 раза
for i in range(1,4):
    # читаем строку файла записываем в переменную l
    l = f.readline()
    # выводим переменную l на экран
    print(l)
# закрываем открытый файл
f.close
```

Шаг 9. Чтение и изменение 3х строк из файла

Считаем из файла 3 строки и изменим их, добавив в конце каждой строки !!!!

```
# открываем файл на чтение
f = open('file.txt','r')
# запускаем цикл 3 раза
for i in range(1,4):
    # читаем строку файла, добавляем «!» записываем в переменную l
    l = f.readline()[:-1]+ '!!!!'
    # выводим переменную l на экран
 print(l)
# закрываем открытый файл
f.close
```

[:-1] – метод удаляет последний символ в строке (у нас это перевод строки)

Шаг 10. Чтение 3х строк из файла, изменение и запись в другой файл

Выведем результат не на экран, а в другой файл

```
# открываем файл на чтение
f = open('file.txt','r')
# открываем файл на запись
fw = open('newfile.txt','w')
# запускаем цикл 3 раза
for i in range(1,4):
    # читаем строку файла, добавляем «!» записываем в переменную l
    l = f.readline()[:-1] + '!!!!'
    # выводим переменную l, но не на экран, а в файл fw
    print(l, file = fw)
# закрываем открытые файлы
f.close
fw.close
```

Note: При открытии файла на запись, если файла с таким именем не было, то python его создаст, Т.е. этот файл нет необходимости создавать заранее

После выполнения этого скрипта вы не увидите результата на экране. Но можете увидеть новый файл «newfile.txt», в котором будут 3 измененные строки входного файла

Шаг 11. Чтение файла полностью

Теперь считаем и обработаем не 3 строки, а все строки входного файла **file.txt**
Сделаем это с помощью цикла while. Цикл while используется в Python для многократного исполнения блока команд до тех пор, пока заданное условие остается истинным:

```
# открываем файл на чтение
f = open(‘file.txt’,’r’)
# открываем второй файл на запись
fw = open(‘newfile.txt’,’w’)
# выполняем пока есть возможность считать строку из файла
while True:
    # считываем строку
    l = f.readline()
    # прерываем цикл, если строка пустая
    if not l:
        break
    # добавляем к строке «!»
    ll = l[:-1] + ‘!!!!’
    # выводим строку на экран
    print(ll)
    # пишем строку в файл
    print(ll, file = fw)
# закрываем открытые файлы
f.close
fw.close
```

Измененные строки должны отобраться на экране и записаться в файл. Проверьте, записались ли строки в файл.

To Know: Тут более подробное описание цикла While в Python

From:
https://sibgu-itlab-wiki.data-pool.ru/ - **SIBGU-ITLAB-WIKI**

Permanent link:
https://sibgu-itlab-wiki.data-pool.ru/zadanie_2._znakomstvo_s_linux_i_python?rev=1711033410

Last update: **2024/03/21 15:03**