Содержание

1 Bash

1.1 echo

Выведет на *stdout* текст

```
echo "Hello World"
```

В Python:

```
print('Hello World')
```

1.2 ls

```
ls -la
```

Listing 1: Выводит содержимое указанное директории и/или информацию о файлах. Ключи можно комбинировать. Поддерживает глобы, когда указывается не полное название файла, а используется $*$. Про глобы дальше.

Ключи:

-l выводит подробную информацию

-a выводит информацию о скрытых файлах тоже ($.$ означает текущую директорию, $..$ означает родительскую директорию)

В Python можно использовать

```
import os

os.listdir('.')  # Вывести всё содержимое текущей директории
```

1.3 pwd

Узнать текущую рабочую директорию

```
pwd
```

В Python

```
import os

os.getcwd()
```

1.4 cd

Change Directory. Перейти в другую директорию.

```
cd /tmp
```

Перейти на уровень выше

```
cd ..
```

Вернуться назад, откуда перешли в текущую директорию

```
cd -
```

В Python можно использовать

```
import os

os.chdir('..')
```

1.5 mkdir

Создать новую директорию.

```
mkdir /tmp/dirname
```

Создать директорию и все директории пути

```
mkdir -p /tmp/notexist/new_directory/target
# создаст /tmp/notexist/,
#         /tmp/notexist/new_directory/,
#         /tmp/notexist/new_directory/target
```

В Python

```
import os

os.mkdir('/tmp/notexist')
```

1.6 rm

Удалить файл или директорию.

В Bash нет понятия корзины — файлы удаляются насовсем!

```
rm /tmp/file.txt
```

Чтобы удалить директорию, надо использовать флаг $-r$ то есть удалить содержимое директории рекурсивно.

```
rm -r /tmp/dir
```

В Python

```
import os

os.unlink('/path/to/file')   # или os.remove
os.rmdir('/path/to/dir')
```

1.7 mv

Переместить (переименовать) файл или директорию

```
# переименует файл
mv homework.py homework.py.backup
# переместить все файлы с суффиксом .py в директорию
mv *.py /path/to/dir
```

В Python (для переноса множества файлов вам придётся написать цикл, который будет проходить по этим файлам)

```
import os

os.replace('/path/to/homework.py', '/path/to/homework.py.backup')
```

1.8 cp

Copy. Копировать файл или директорию.

```
# Создаст копию файла homework.py
cp homework.py homework.py.backup
```

Чтобы копировать директорию, нужен флаг $-r$ то есть копировать рекурсивно

```
cp -r /tmp/dir /tmp/new_dir
```

1.9 cat

Вывести содержимое файла/ов на stdout.

```
cat homework.py
```

В Python

```
with open('/path/to/homework.py', 'r') as fd:
    print(fd.read())
```

1.10 head

Вывести первые n (по умолчанию 10) строк из файла

```
head -n15 homework.py
```

В Python

```
with open('/path/to/file', 'r') as fd:
    fd.readlines(hint=15)
```

1.11 tail

Вывести последние n (по умолчанию 10) строк из файла

```
# Прочитать последние 15 строк
tail -n15 /path/to/homework.py
```

У tail есть очень полезная функция: читать изменения в файле. То есть, он будет обновлять файл если в него что-нибудь будет дозаписываться в этот момент.

```
tail -f /path/to/file
```

В Python (трудно реализовать не прочитав весь файл, здесь читается весь файл целиком)

```
with open('/path/to/homework.py', 'r') as fd:
    result = '\n'.join(fd.read().split('\n')[-15:])
```

1.12 less / more

"Читалка"текстовых файлов с возможностью листать текст взад-вепрёд стрелками клавиатуры

```
less homework.py
# может быть more а не less
more homework.py
```

1.13 grep

Поиск подстроки в файле

```
grep class homework.py
```

grep -n вернёт результат с указанием номера строки, где встретилась искомая подстрока

```
grep -n class homework.py
```

Можно искать по многим файлам

```
# ключ -l вернёт только название файла, где встретилась искомая по
grep -l class venv/lib/python3.8/site-packages/*/*.py
```

1.14 chmod

Команда для управления правами доступа к файлу. Может быть полезна для добавления файлу прав на запуск, делает файл исполняемым скриптом.

```
chmod +x homework.py
# Теперь homework.py можно запустить (если указан shebang)
./homework.py
```

1.15 which

Покажет путь, откуда будет вызываться исполняемый файл. Полезно, чтобы проверить будет ли вызываться Python-интерпретатор из глобально окружения или из вашего venv.

```
# вернёт один путь
which python
# активируем venv и получим другой путь
source venv/Scripts/activate
which python
```

1.16 date

Вывести текущую дату

```
date
```

```
date +"%d-%m-%Y"
```

1.17 history

Посмотреть историю команд, которые вы вызвали

```
history
```

```
1  sudo apt install cmake
2  pip3 install sklearn pandas matplotlib graphviz sympy scipy num
-user
3  apt search graphviz
4  sudo apt update
5  sudo apt install graphviz
6  pip3 install seaborn --user
7  pip3 install xgboost --user
```

Можно вызвать повторно команду из истории, указав номер команды

```
# Вызовет команду под номером 4 в истории: sudo apt update
!4
```

1.18 man

Документация по командам

```
# Выведет документацию на man
man man
# Выведет документацию на tail
man tail
```

В Python

```
help(help)
```

1.19 apropos

Поиск по документации. Находит статьи, которые можно открыть через *man*.

```
apropos python
```

1.20 vi

Стандартный в ∗NIX текстовый редактор Vi. Возможно, есть vim.

```
vi homework.py
# или
vim homework.py
```

1.21 nano

user-friendly текстовый редактор

```
nano -w homework.py
```

1.22 ssh

Позволяет подключиться к удалённой машине, используя защищённый протокол

```
# Подключиться, используя hostname
ssh username@hostname.domain
# Подключиться, используя IP-address
ssh username@192.168.0.1
```

В Python есть библиотека fabric

1.23 scp

Скопировать файлы с удалённой машины или отправить на удалённую машину, используя защищённый протокол

```
scp homework.py username@hostname.domain:/path/to/dir
```

2 Фишки bash (и других ∗NIX-шеллов)

Почитать про идеологию UNIX и освоить базовое программирование на Bash можно в книге Программное окружение UNIX.

2.1 перенаправление

В ∗NIX-системах есть понятие файловых дескрипторов. По умолчанию, каждый запущенный процесс получает от ядра операционной системы три обязательных дескриптора:

stdin с индексом 0. Это стандартный ввод приложения, отсюда он может читать то, что пользователь вводит с клавиатуры, например.

stdout с индексом 1. Это стандартный вывод приложения, сюда выводятся данные через echo или print в Python.

stderr с индексом 2. Это стандартный вывод для сообщений об ошибках или предупреждениях. При вызове программы по умолчанию stderr выводится туда же куда и stdout.

Эти дескрипторы можно перенаправлять. Например, сохранить сообщения об ошибках в отдельном log-файле

```
pytest 2> homework.log
# Затем мы можем прочитать вывод об ошибках,
# используя less или more
less homework.log
```

2.2 pipe

В ∗NIX-системах можно "склеивать"программы друг с другом. По умолчанию, на вход следующей программе подаётся stdout предыдущей.

```
# Вернуть 15-ую строчку файла
tail -n15 homework.py |head -n1
```

```
# посчитает количество строк в файле
cat homework.py | wc -l
```

2.3 glob

Позволяет указывать множество файлов и/или директорий не перечисляя каждый. Используется в правилах *.gitignore*.

***** ноль или сколько угодно символов

? ноль или один символ

[abc] один из этих трёх символов

[a-zA-Z] один из символов алфавита

[0-9] одна из цифр

Подробнее почитать про синтаксис glob можно вызвав команду $man7glob$

Перенести все .py файлы в нужную директорию

```
mv *.py /path/to/python_scripts/
```

Удалить все *.pyc и *.pyo файлы

```
rm *.py[co]
```

2.4 shebang

Указывает bash как именно надо запустить скрипт. Прописывается обязательно в самой первой строке файла. Например, чтобы сделать файл скриптом на Python и затем запускать его, надо добавить первой строкой такой shebang.

```
#!/usr/bin/python
# укажите там путь до вашего интерпретатора Python
# в Linux можно использовать #!/usr/bin/env python
```

Узанть путь до вашего интерпретатора можно, используя команду which

3 Vi

Vi является стандартным текстовым редактором в ∗NIX-системах, поэтому он гарантированно будет, например, на Linux-сервере.

3.1 запуск vi

```
vi /path/to/file
```

3.2 режимы в vi

В vi есть два режима:

- режим команд. изначально вы попадаете именно в него. всё, что вы будете печатать на клавиатуре, будет восприниматься как команды, а не текст.

- режим ввода. об этом режиме сигнализирует большая надпись **–INSERT–** в левом нижнем углу редактора.

Чтобы войти в режим ввода нажмите **i**. Чтобы выйти из режима ввода нажмите **Esc**.

3.3 редактирование текста в vi

можете передвигать курсор стрелочками клавиатуры и вводить текст в режиме ввода.

3.4 сохранить файл

Выйдите в режим команд, если вы были в режиме ввода текста, нажам клавишу **ESC**. Наберите команду (в нижнем левом углу будет печататься команда)

`:w`

vi сообщит, что файл был записан (если у вас есть права на запись файла).

3.5 выход из vi

Выйдите в режим команд, если вы были в режиме ввода текста, и наберите команду

`:q`

Если вы внесли изменения, но забыли сохранить файл перед выходом, то vi сообщит вам об этом и не закроется. Сохраните файл и повторите команду выхода;

или закройте файл принудительно, без сохранения

`:q!`

или можно совместить команды сохранения и выхода

`:wq`

3.6 поиск в vi

Для поиска подстроки необходимо перейти в режим команд и вызвать команду поиска, нажав

`/`

вы увидите в нижнем левом углу приглашение (символ **/**) на ввод подстроки, которую вы хотите найти в тексте. Вводите текст и нажимайте **Enter**. Для перехода к следующему совпадению, нажимайте **n**, для перехода к предыдущему совпадению **N**