

Шпаргалка по Bash и Vi

Вавилин Павел

Содержание

1	Bash	2
1.1	echo	2
1.2	ls	2
1.3	pwd	3
1.4	cd	3
1.5	mkdir	3
1.6	rm	4
1.7	mv	4
1.8	cp	4
1.9	cat	5
1.10	head	5
1.11	tail	5
1.12	less / more	6
1.13	grep	6
1.14	chmod	7
1.15	which	7
1.16	date	7
1.17	history	7
1.18	man	8
1.19	apropos	8
1.20	vi	9
1.21	nano	9
1.22	ssh	9
1.23	scp	9
2	Фишки bash (и других *NIX-шеллов)	9
2.1	перенаправление	9
2.2	pipe	10
2.3	glob	10
2.4	shebang	11
3	Vi	11
3.1	запуск vi	11
3.2	режимы в vi	11
3.3	редактирование текста в vi	12
3.4	сохранить файл	12
3.5	выход из vi	12
3.6	поиск в vi	12

1 Bash

1.1 echo

Выведет на *stdout* текст

```
|echo "Hello World"
Hello World
```

1.1.1 В Python

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

1.2 ls

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

```
|ls -la
total 40
drwxrwxr-x 9 pimienta pimienta 4096 Jun 13 21:41 .
drwxrwxr-x 3 pimienta pimienta 4096 Jun 13 21:41 ..
-rw-rw-r-- 1 pimienta pimienta 126 Jun 13 21:41 easy_install.py
drwxrwxr-x 5 pimienta pimienta 4096 Jun 13 21:41 pip
drwxrwxr-x 2 pimienta pimienta 4096 Jun 13 21:41 pip-20.0.2.dist-
info
drwxrwxr-x 5 pimienta pimienta 4096 Jun 13 21:41 pkg_resources
drwxrwxr-x 2 pimienta pimienta 4096 Jun 13 21:41 pkg_resources-
0.0.0.dist-info
drwxrwxr-x 2 pimienta pimienta 4096 Jun 13 21:41 __pycache__
drwxrwxr-x 6 pimienta pimienta 4096 Jun 13 21:41 setuptools
drwxrwxr-x 2 pimienta pimienta 4096 Jun 13 21:41 setuptools-44.0.0.
info
```

Ключи:

-l выводит подробную информацию

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

1.2.1 В Python

```
|import os
os.walk('.') # Вывести всё содержимое родительской директории
```

1.3 pwd

Узнать текущую рабочую директорию

```
|pwd  
/home/pimiento/yap
```

1.3.1 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 -
```

1.4.1 В Python

```
|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
```

1.5.1 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
```

1.6.1 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
```

1.7.1 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

Сору. 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
```

1.9.1 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
```

1.10.1 В Python

```
| with open('/path/to/file', 'r') as fd:  
|     print(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
```

1.11.1 В Python

трудно реализовать не прочитав весь файл. здесь сначала читается весь файл целиком

```
| with open('/path/to/homework.py', 'r') as fd:  
|     # сначала прочитаем весь файл целиком  
|     # разобьём его на строки  
|     # и затем уже вернём последние 15 строк  
|     print('\n'.join(fd.read().split('\n')[-15:]))
```

1.12 less / more

"Читалка" текстовых файлов с возможностью листать текст взад-вперёд 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

class Calculator:
class CaloriesCalculator(Calculator):
class CashCalculator(Calculator):
class Record:
```

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

```
| grep -n class homework.py

5:class Calculator:
36:class CaloriesCalculator(Calculator):
48:class CashCalculator(Calculator):
78:class Record:
```

Можно искать по многим файлам

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

venv/lib/python3.8/site-packages/pkg_resources/__init__.py
venv/lib/python3.8/site-packages/setuptools/__init__.py
venv/lib/python3.8/site-packages/setuptools/_deprecation_warning.py
venv/lib/python3.8/site-packages/setuptools/_imp.py
venv/lib/python3.8/site-packages/setuptools/archive_util.py
venv/lib/python3.8/site-packages/setuptools/build_meta.py
venv/lib/python3.8/site-packages/setuptools/config.py
venv/lib/python3.8/site-packages/setuptools/depends.py
venv/lib/python3.8/site-packages/setuptools/dist.py
venv/lib/python3.8/site-packages/setuptools/errors.py
venv/lib/python3.8/site-packages/setuptools/extension.py
venv/lib/python3.8/site-packages/setuptools/installer.py
venv/lib/python3.8/site-packages/setuptools/lib2to3_ex.py
venv/lib/python3.8/site-packages/setuptools/monkey.py
venv/lib/python3.8/site-packages/setuptools/msvc.py
```

```
venv/lib/python3.8/site-packages/setuptools/namespaces.py
venv/lib/python3.8/site-packages/setuptools/package_index.py
venv/lib/python3.8/site-packages/setuptools/py31compat.py
venv/lib/python3.8/site-packages/setuptools/py33compat.py
venv/lib/python3.8/site-packages/setuptools/sandbox.py
venv/lib/python3.8/site-packages/setuptools/ssl_support.py
venv/lib/python3.8/site-packages/setuptools/wheel.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
Mon 14 Jun 2021 09:10:11 PM +05

| date +"%d-%m-%Y"
14-06-2021
```

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
```

1.18.1 В Python

```
| help(help)
```

1.19 apropos

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

```
| apropos python3

2to3-2.7 (1) - Python2 to Python3 converter
ipython3 (1) - Tools for Interactive Computing in
Python.
py3versions (1) - print python3 version information
python3 (1) - an interpreted, interactive,
object-oriented programmi...
python3-config (1) - output build options for python C/C++
extensions or em...
python3.8 (1) - an interpreted, interactive,
object-oriented programmi...
python3.8-config (1) - output build options for python C/C++
extensions or em...
x86_64-linux-gnu-python3-config (1) - output build options
for python C/C++ e...
x86_64-linux-gnu-python3.8-config (1) - output build options
for python C/C++....
```


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
```

1.22.1 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 строку файла  
|tail -n15 homework.py |head -n1
```

```
return CREDIT
```

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

```
89
```

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 **man 7 glob**

Перенести все .py файлы в нужную директорию

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

Удалить все *.рус и *.руо 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
```

Узнуть 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, нажав 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 вводите 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**