Оглавление

[Часть 1 3](#_Toc132567312)

[1. Напишите сценарий, который выводит дату, время, список зарегистрировавшихся пользователей, и uptime системы. 3](#_Toc132567313)

[2. Напишите сценарий, который выводит содержимое любого каталога или сообщение о том, что его не существует. 3](#_Toc132567314)

[3. Напишите сценарий, который с помощью цикла прочитает файл и выведет его содержимое. 4](#_Toc132567315)

[4. Напишите сценарий, который с помощью цикла выведет список файлов и директорий. 4](#_Toc132567316)

[5. Напишите сценарий, который подсчитает объем диска, занимаемого директорией. В качестве директории можно выбрать любую 5](#_Toc132567317)

[6. Напишите сценарий, который выведет список всех исполняемых файлов в директории, для которых у текущего пользователя есть права на исполнение. 5](#_Toc132567318)

[Часть 2 6](#_Toc132567319)

[1. Определение зависимостей проекта. 6](#_Toc132567320)

[2. Создание виртуального окружения 6](#_Toc132567321)

[3. Написание скрипта запуска приложения на новой системе 6](#_Toc132567322)

# Часть 1

## **Напишите сценарий, который выводит дату, время, список зарегистрировавшихся пользователей, и uptime системы.**

Данный скрипт на языке bash выводит на экран текущую дату, список зарегистрировавшихся пользователей и время работы системы.

Листинг 1 – Фрагмент кода #1:

1. #!/bin/bash
2. # Сценарий для вывода даты, времени, списка зарегистрированных пользователей и uptime системы,
3. #  а также сохранения этой информации в файл:
4. # Определение переменных
5. date=$(date)
6. uptime=$(uptime)
7. users=$(who | awk '{print $1}')
8. # Вывод информации в консоль
9. echo "Дата и время: $date"
10. echo "Зарегистрированные пользователи: $users"
11. echo "Uptime системы: $uptime"
12. # Сохранение информации в файл
13. echo "Дата и время: $date" > system\_info.txt
14. echo "Зарегистрированные пользователи: $users" >> system\_info.txt
15. echo "Uptime системы: $uptime" >> system\_info.txt

## **Напишите сценарий, который выводит содержимое любого каталога или сообщение о том, что его не существует.**

Этот скрипт на языке bash проверяет существование директории и выводит список её содержимого. Если директория не найдена, то выводится сообщение об этом.

Листинг 2 – Фрагмент кода #2:

1. #!/bin/bash
2. # Сценарий для вывода содержимого каталога или сообщения о его отсутствии:
3. # Определение переменных
4. dir="/path/to/directory"
5. # Проверка существования каталога
6. if [ -d "$dir" ]
7. then
8. # Вывод содержимого каталога
9. ls "$dir"
10. else
11. echo "Каталог не существует"
12. fi

## **Напишите сценарий, который с помощью цикла прочитает файл и выведет его содержимое.**

Данный скрипт на языке bash считывает содержимое файла file.txt построчно и выводит каждую строку на экран.

Листинг 3 – Фрагмент кода #3:

1. #!/bin/bash
2. # Сценарий для чтения и вывода содержимого файла:
3. # Определение переменных
4. file="/path/to/file"
5. # Проверка существования файла
6. if [ -f "$file" ]
7. then
8. # Цикл чтения файла
9. while read line
10. do
11. echo "$line"
12. done < "$file"
13. else
14. echo "Файл не существует"
15. fi

## **Напишите сценарий, который с помощью цикла выведет список файлов и директорий.**

Данный скрипт на языке bash выводит список файлов и директорий из текущего каталога. Для каждого элемента (файла или директории) выводится информация о типе этого элемента: файл или директория.

Листинг 4 – Фрагмент кода #4:

1. #!/bin/bash
2. # Сценарий для вывода списка файлов и директорий из текущего каталога:
3. # Цикл чтения содержимого текущего каталога
4. for item in \*
5. do
6. # Проверка типа элемента (файл или директория)
7. if [ -f "$item" ]
8. then
9. echo "$item - файл"
10. elif [ -d "$item" ]
11. then
12. echo "$item - директория"
13. fi
14. done

## **Напишите сценарий, который подсчитает объем диска, занимаемого директорией. В качестве директории можно выбрать любую**

**директорию в системе.**

Этот скрипт на языке bash подсчитывает объем дискового пространства, занимаемого указанной директорией, и выводит его на экран. Команда du используется для подсчета размера файлов и директорий, а флаг -sh используется для вывода размера в удобочитаемом формате.

Листинг 5 – Фрагмент кода #5:

1. #!/bin/bash
2. # Сценарий для подсчета объема диска, занимаемого директорией:
3. # Определение переменных
4. dir="/path/to/directory"
5. # Проверка существования директории
6. if [ -d "$dir" ]
7. then
8. # Подсчет объема диска
9. du -sh "$dir"
10. else
11. echo "Директория не существует"
12. fi

## **Напишите сценарий, который выведет список всех исполняемых файлов в директории, для которых у текущего пользователя есть права на исполнение.**

Данный скрипт проходится циклом по текущему каталогу и проверяет файл на то является ли он исполняемым.

Листинг 6 – Фрагмент кода #6:

1. #!/bin/bash
2. # Сценарий для вывода списка всех исполняемых файлов в директории,
3. #  для которых у текущего пользователя есть права на исполнение:
4. # Цикл чтения содержимого текущего каталога
5. for item in \*
6. do
7. # Проверка типа элемента (файл) и наличия прав на исполнение
8. if [ -x "$item" ]
9. then
10. echo "$item"
11. fi
12. done

# Часть 2

1. Определение зависимостей проекта.

Для начала нужно скачать и распаковать архив с проектом из задания данной командой: tar -xvf blocknote-master.tar.gz. Для сбора всех не стандартных библиотек из файлов был написан код на языке python его листинг будет в конце данного отчёта. Основная суть кода заключается в сборе всех импортов и удалении из них стандартных библиотек: sys.modules.keys().

1. Создание виртуального окружения

Для создания виртуального окружения данным способом требуется, чтобы python был в path. Данный набор команд создаст и активирует виртуальное окружение (для линукс систем при установке требуется указывать версию python. обычно это python3, в моём случае – python3.10):

# Для Windows

> python -m venv Venv

> Venv\Scripts\Activate.ps1

> pip install -r requrements.txt

# Для Linux

> python3.10 -m venv Venv

> source Venv/bin/activate

> pip install -r requrements.txt

1. Написание скрипта запуска приложения на новой системе

Состав скрипта зависит от того, как именно проект будет распространяться. Если учитывать, что он будет распространяться как пакет, загруженный в PyPi будет достаточно использовать pip install. Eсли данный проект будет распространяться через github также в виде пакета, то будет достаточно выполнить команду git clone, и python install.py или следовать инструкции. Для своего примера я выбрал метод распространения через файлообменинк.

Листинг 7 – Часть 2 3 задание

1. #!/bin/bash
2. mkdir ProjectDjango
3. cd ProjectDjango
4. wget https://www.dropbox.com/s/ija7ax3sj6ysb0p/blocknote-master.tar.gz
5. tar -xvf blocknote-master.tar.gz
6. wget  https://github.com/Zhuchka\_03/TRPP/pract3(bash)/part2/new\_requirements.py
7. python3 new\_requirements.py
8. pip install -r requirements.txt
9. python manage.py makemigrations
10. python manage.py migrate
11. python manage.py runserver
12. echo "All Done!"

Листинг 8 – python код для генерации requirements.txt

1. import os
2. import sys
3. import chardet
4. import importlib.util
5. # список стандартных библиотек Python
6. std\_lib = sys.modules.keys()
7. # список уникальных имен библиотек
8. unique\_libs = set()
9. def process\_file(file\_path):
10. *"""*
11. Обрабатывает файл с заданным путем.
12. Ищет все имена библиотек в файле и добавляет их в список unique\_libs.
13. """
14. encoding = None
15. with open(file\_path, 'rb') as f:
16. data = f.read()
17. encoding = chardet.detect(data)['encoding']
18. with open(file\_path, "r", encoding=encoding) as f:
19. code = f.read()
20. for line in code.split("\n"):
21. line = line.strip()
22. if line.startswith("import") or line.startswith("from"):
23. tokens = line.split()
24. if len(tokens) > 1:
25. lib\_name = tokens[1]
26. if "." in lib\_name:
27. lib\_name = lib\_name.split(".")[0]
28. unique\_libs.add(lib\_name)
29. def process\_directory(directory):
30. *"""*
31. Обрабатывает директорию и вызывает функцию process\_file для каждого файла .py.
32. """
33. for dirpath, dirnames, filenames in os.walk(directory):
34. for filename in filenames:
35. if filename.endswith(".py"):
36. file\_path = os.path.join(dirpath, filename)
37. process\_file(file\_path)
38. def write\_to\_file(libs, filename):
39. *"""*
40. Записывает список имен библиотек в файл с заданным именем.
41. """
42. with open(filename, "w") as f:
43. for lib in libs:
44. f.write(lib + "\n")
45. # получаем относительный путь к папке blocknote-master
46. blocknote\_dir = os.path.join(os.path.dirname(\_\_file\_\_), "blocknote-master")
47. result\_file = os.path.join(os.path.dirname(\_\_file\_\_), "requirements.txt")
48. # вызов функций
49. process\_directory(blocknote\_dir)
50. unique\_libs = [lib for lib in unique\_libs if lib not in std\_lib]
51. write\_to\_file(unique\_libs, result\_file)