

Оглавление

[1. Введение 3](#_Toc194150723)

[1.1 Цели работы 3](#_Toc194150724)

[2. Ход работы. Базовые Bash скрипты 4](#_Toc194150725)

[2.1 Задание 1 4](#_Toc194150726)

[2.2 Задание 2 5](#_Toc194150727)

[2.3 Задание 3 6](#_Toc194150728)

[2.4 Задание 4 6](#_Toc194150729)

[2.5 Задание 5 7](#_Toc194150730)

[2.6 Задание 6 8](#_Toc194150731)

[3. Ход работы. Развертка и запуск проекта при помощи Bash Script 10](#_Toc194150732)

[3.1 Определение зависимостей проекта 10](#_Toc194150733)

[3.2 Создание виртуального окружения 11](#_Toc194150734)

[3.3 Написание скрипта запуска приложения на новой системе 13](#_Toc194150735)

[4. Выводы 15](#_Toc194150736)

[4.1 Выводы 15](#_Toc194150737)

1. Введение
   1. Цели работы

Данная работа посвящена основам написания Bash-скриптов. Bash-скрипты являются мощным инструментом автоматизации в Unix-подобных системах, позволяя объединять последовательности команд в единые исполняемые файлы для эффективного решения разнообразных задач. В основе любого скрипта лежит шебанг-строка #!/bin/bash, которая указывает системе на необходимость использования интерпретатора Bash для обработки содержащихся в файле инструкций. После указания интерпретатора следует основной код, состоящий из набора команд, выполняемых последовательно сверху вниз.

Язык Bash предоставляет разработчикам богатый набор возможностей, включая работу с переменными для хранения данных, использование арифметических операций для вычислений, подстановку результатов выполнения команд, а также сложные управляющие конструкции.

Освоение базовых элементов, таких как объявление переменных, выполнение системных команд, организация условий и циклов, а также проверка состояния файловой системы, формирует необходимую основу для дальнейшего углубленного изучения Bash-программирования.

1. Ход работы. Базовые Bash скрипты
   1. Задание 1

Напишите сценарий, который выводит дату, время, список зарегистрировавшихся пользователей, и uptime системы и сохраняет эту информацию в файл. Программная реализация данного задания представлена на рисунке 1, а результат скрипта на рисунке 2.

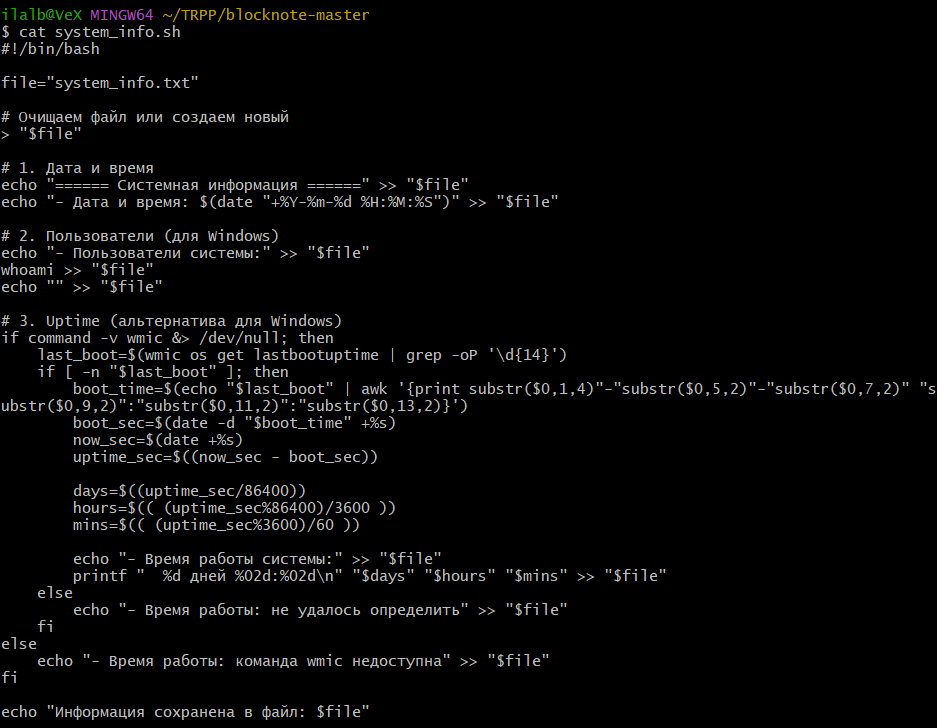


Рисунок – Программная реализация задания 1

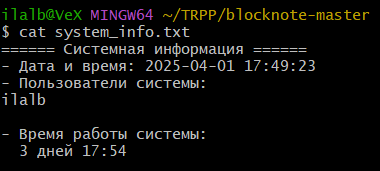


Рисунок – Дата, время, список пользователей и uptime

* 1. Задание 2

Напишите сценарий, который выводит содержимое любого каталога или сообщение о том, что его не существует. Программная реализация скрипта представлена на рисунке 3, а тестирование на рисунке 4.

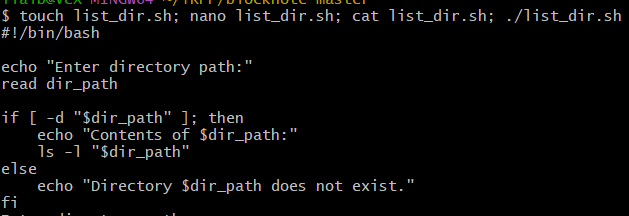


Рисунок 3 – Программная реализация задания 2

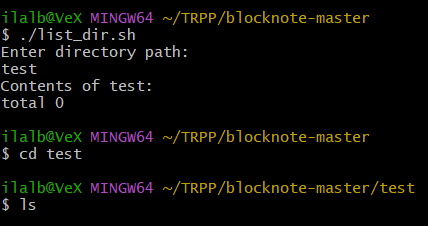


Рисунок 4 – Тестирование скрипта для задания 2

* 1. Задание 3

Напишите сценарий, который с помощью цикла прочитает файл и выведет его содержимое. Программная реализация скрипта и результат представлены на рисунках 5-6 соответственно.

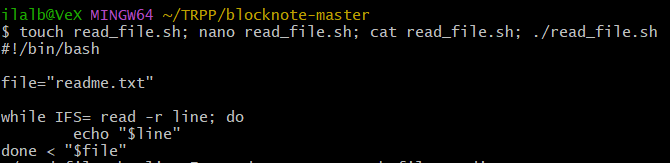


Рисунок 5 – Программная реализация задания 3

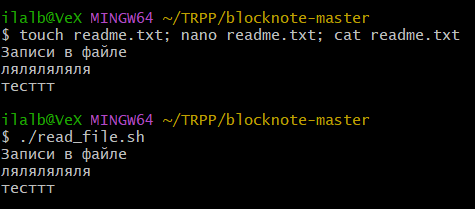


Рисунок 6 – Построчное чтение файла

* 1. Задание 4

Напишите сценарий, который с помощью цикла выведет список файлов и директорий из текущего каталога, укажет, что есть файл, а что директория. Программная реализация скрипта и результат представлены на рисунках 7-8 соответственно.

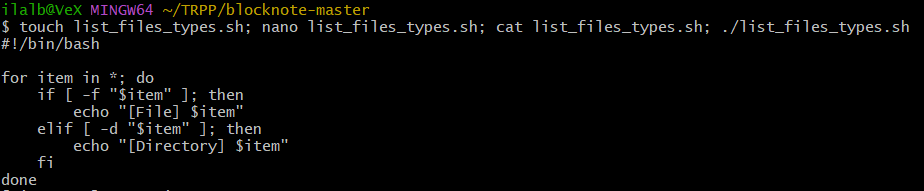


Рисунок 7 – Программная реализация задания 4

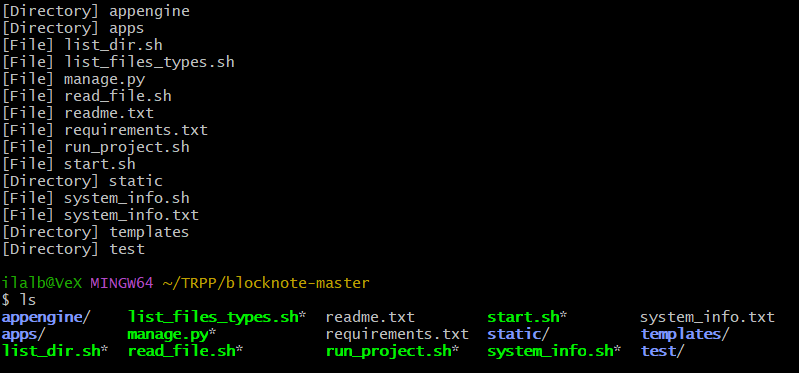


Рисунок 8 – Содержимое каталога

* 1. Задание 5

Напишите сценарий, который подсчитает объем диска, занимаемого директорией. В качестве директории можно выбрать любую директорию в системе. Программная реализация скрипта и результат представлены на рисунках 9-10 соответственно.

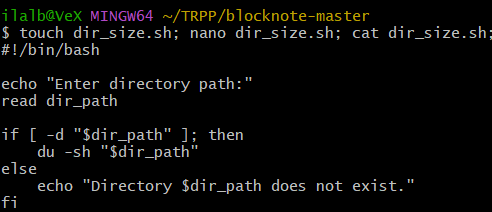


Рисунок 9 – Программная реализация задания 5

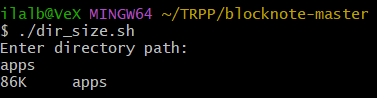


Рисунок 10 – Объем диска, занимаемый текущей директорией

* 1. Задание 6

Напишите сценарий, который выведет список всех исполняемых файлов в директории, для которых у текущего пользователя есть права на исполнение. Программная реализация скрипта и результат представлены на рисунках 11-12 соответственно.

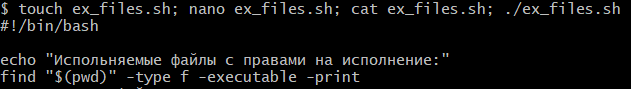


Рисунок 11 – Программная реализация задания 6

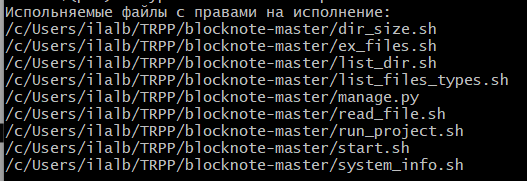


Рисунок 12 – Все исполняемые файлы в текущей директории

1. Ход работы. Развертка и запуск проекта при помощи Bash Script
   1. Определение зависимостей проекта

Файл «requirements.txt» играет важную роль в управлении зависимостями Python-проекта. Он содержит список всех необходимых библиотек и их версий, что позволяет быстро воспроизвести рабочую среду на любом компьютере или сервере. Это обеспечивает воспроизводимость, автоматизацию установки зависимостей и контроль версий пакетов, предотвращая конфликты и ошибки.

В ходе данной работы необходимо составить файл зависимостей для [следующего проекта](https://www.dropbox.com/scl/fi/voifzl1zetz99d1u51pcj/blocknote-master.tar.gz?rlkey=mhzggcu1ym3unnr3cbdojbs9u) (доступен по ссылке). Генерация файла и его содержимого представлена на рисунках 13-14.

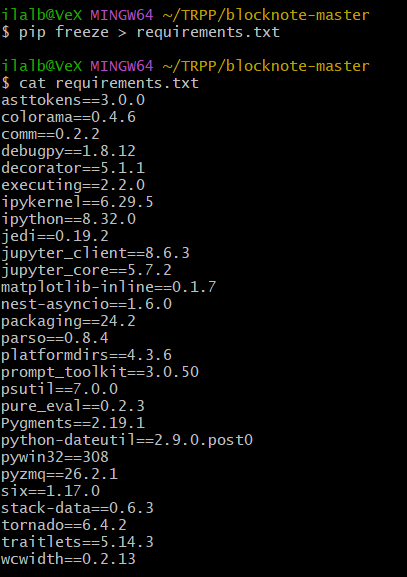


Рисунок 13 – Автоматическая генерация файла requirements.txt

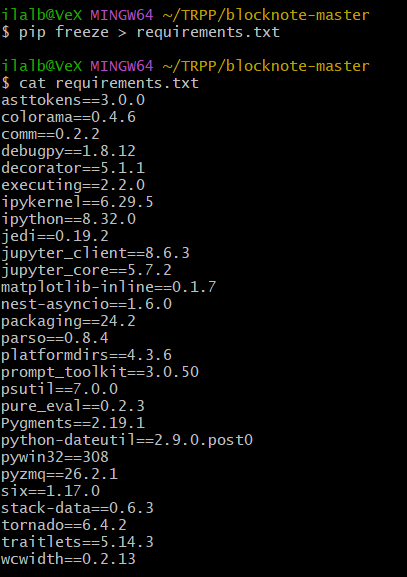


Рисунок 14 – Список зависимостей для проекта

* 1. Создание виртуального окружения

Необходимо на основании составленного в прошлом шаге списка команд написать скрипт скачивания указанного в прошлом шаге проекта с последующим созданием виртуального окружения и настройкой его под проект, то есть установкой всех необходимых библиотек. Скрипт для создания виртуального окружения представлен на рисунке 15, процесс работы скрипта виден на рисунках 16-17.

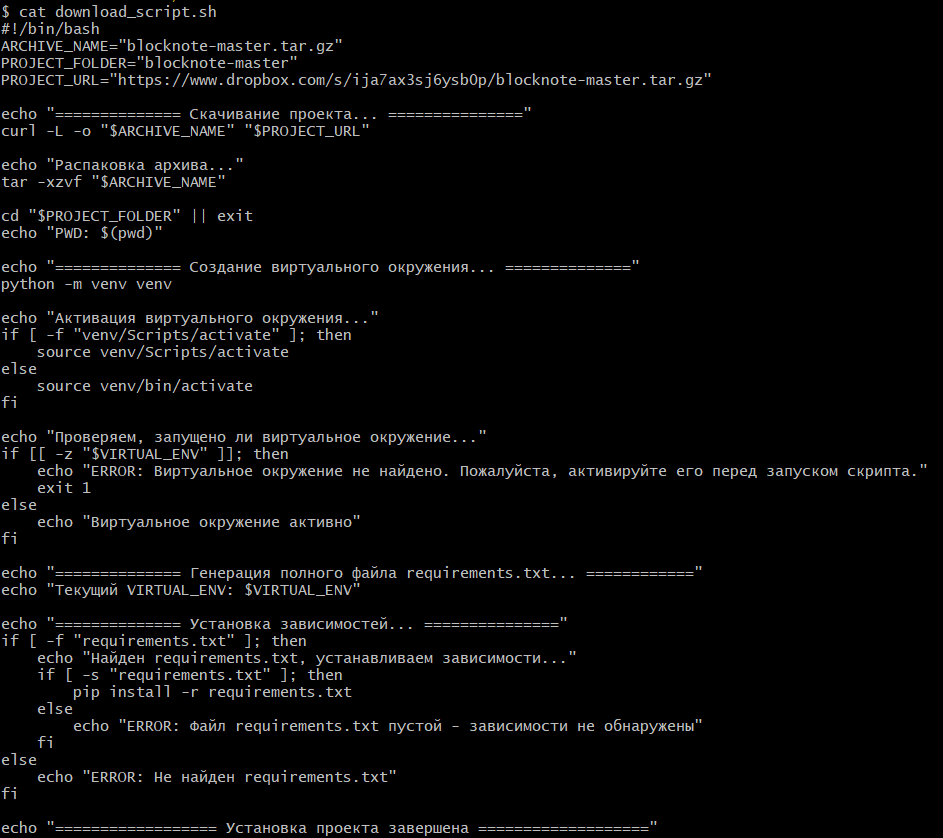


Рисунок 15 – Скачивание проекта и создание виртуального окружения

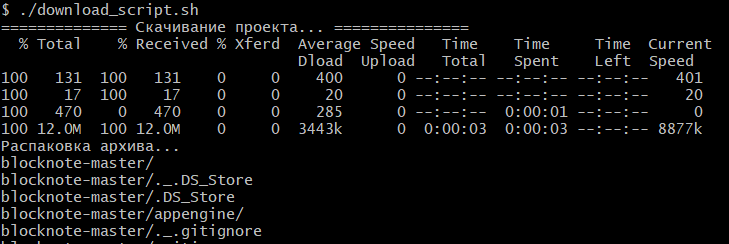


Рисунок 16 – Скачивание проекта и создание виртуального окружения. Часть 1

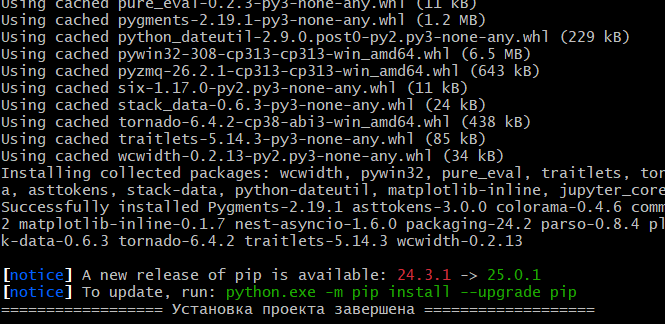


Рисунок 17 - Скачивание проекта и создание виртуального окружения. Часть 2

3.3 Написание скрипта запуска приложения на новой системе

Пришло время запустить проект, при этом воссоздав полученное на прошлом этапе виртуальное окружение со всеми зависимости, после чего запустить код, представленный на рисунке 18.

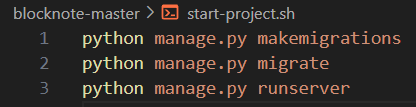


Рисунок 18 – Программная реализация запуска проекта

Проект настроен и готов к запуску (рис. 19-20).

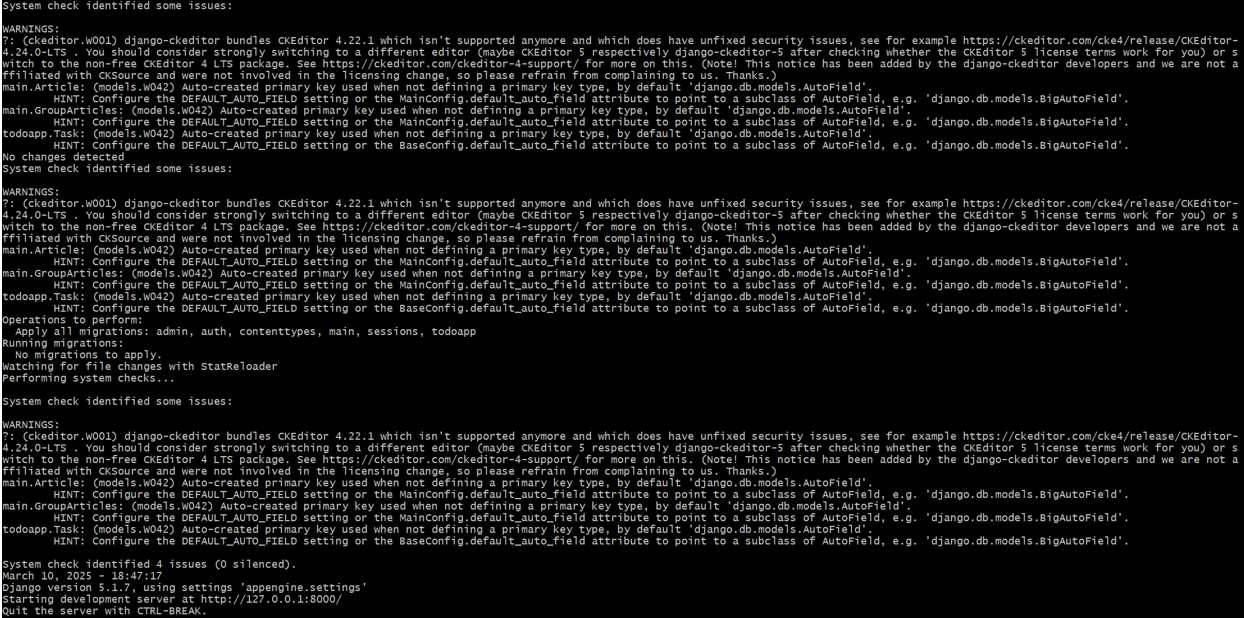


Рисунок 19 – Запуск проекта

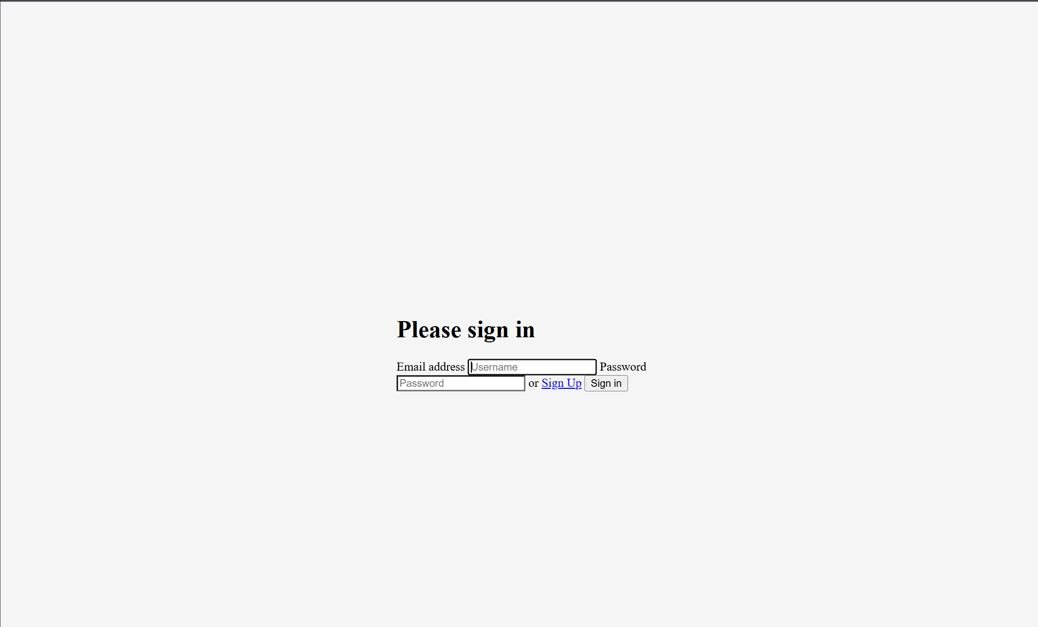


Рисунок 20 – Запущенный проект

1. Выводы
   1. Выводы

В рамках практической работы был детально проанализирован процесс разработки и применения Bash-скриптов в Unix-средах. Основное внимание уделялось фундаментальным аспектам, включая синтаксис шебанг-строки, работу с переменными, механизмы подстановки команд, выполнение арифметических операций, а также применение управляющих конструкций и циклических структур.

Особое значение в работе было уделено вопросам управления зависимостями в программных проектах. Процедура идентификации необходимых компонентов играет критическую роль в обеспечении стабильной работы приложений в различных средах исполнения. Формирование файла requirements.txt, содержащего исчерпывающий перечень требуемых библиотек, существенно упрощает процессы развёртывания и настройки рабочего окружения.

Практическое освоение Bash-скриптинга открывает значительные перспективы для системных администраторов и разработчиков, работающих в Unix-подобных средах. Полученные знания базовых принципов создания скриптов служат отправной точкой для разработки более сложных и функциональных решений.