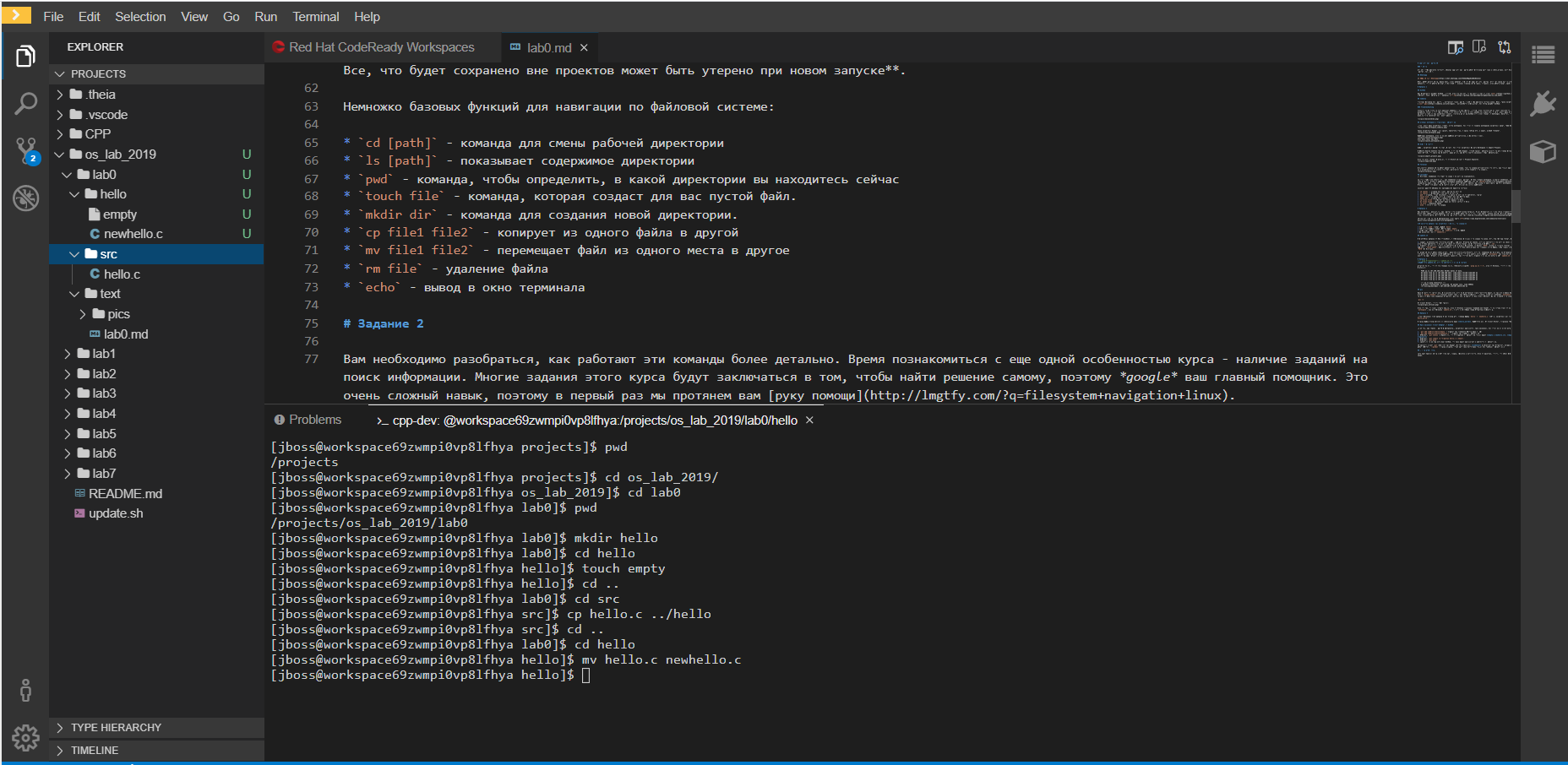
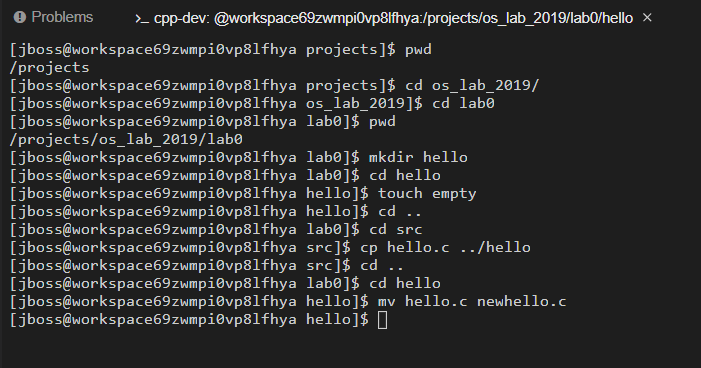
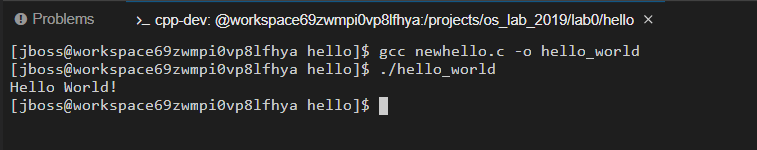
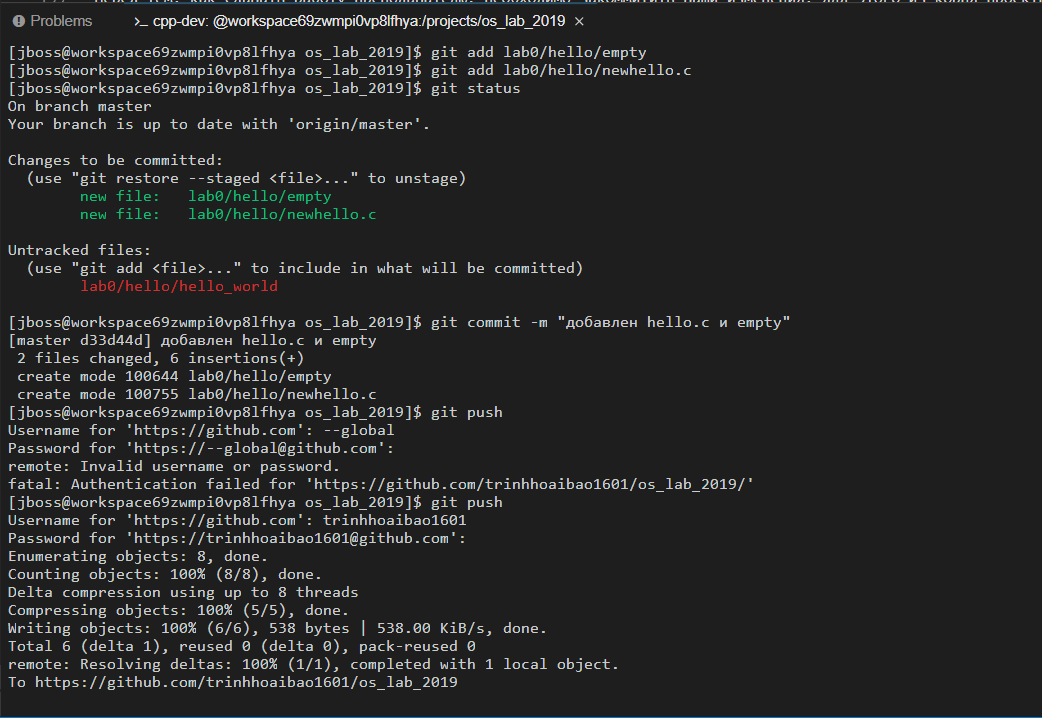
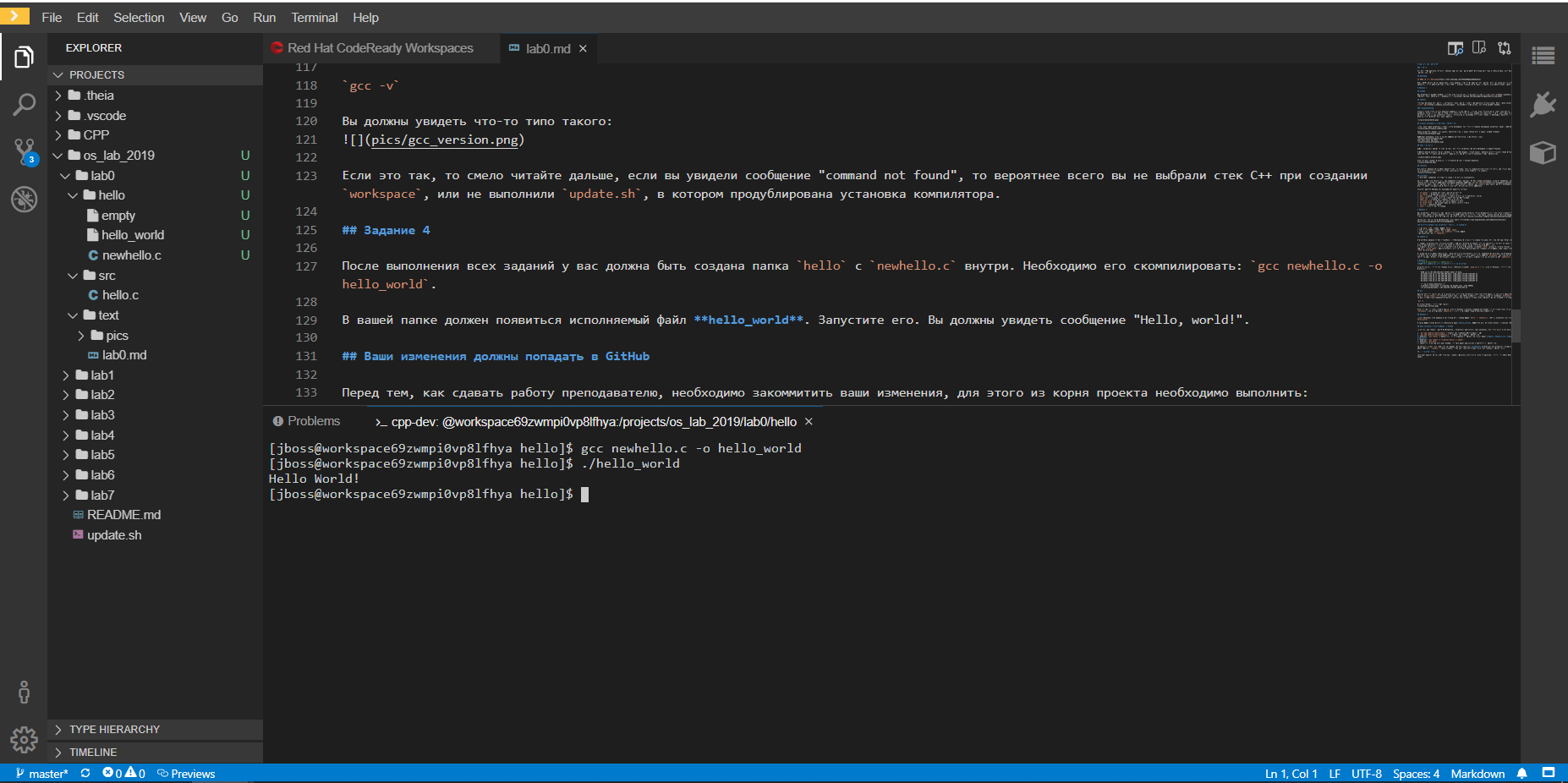
**Лабораторная работа №0**

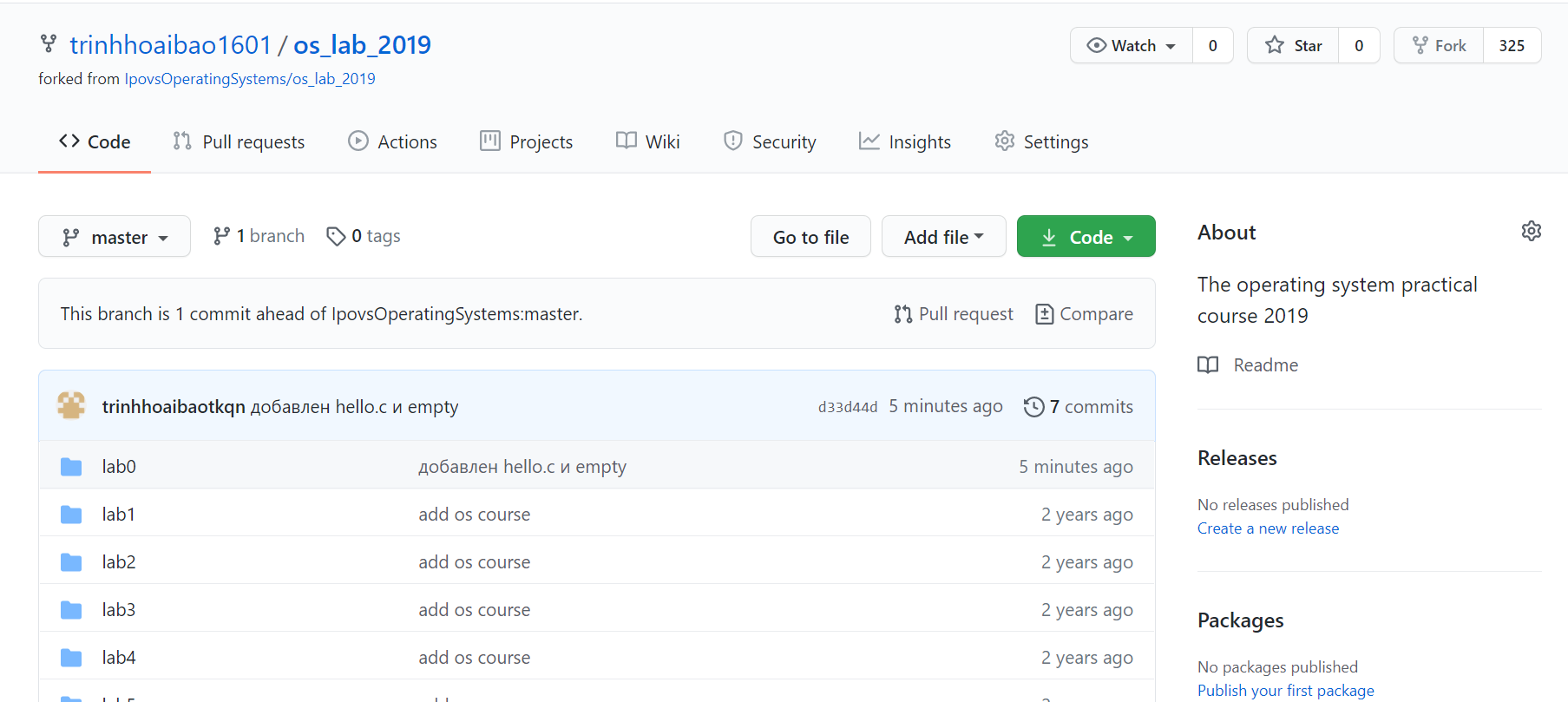
# *Задание 2:*

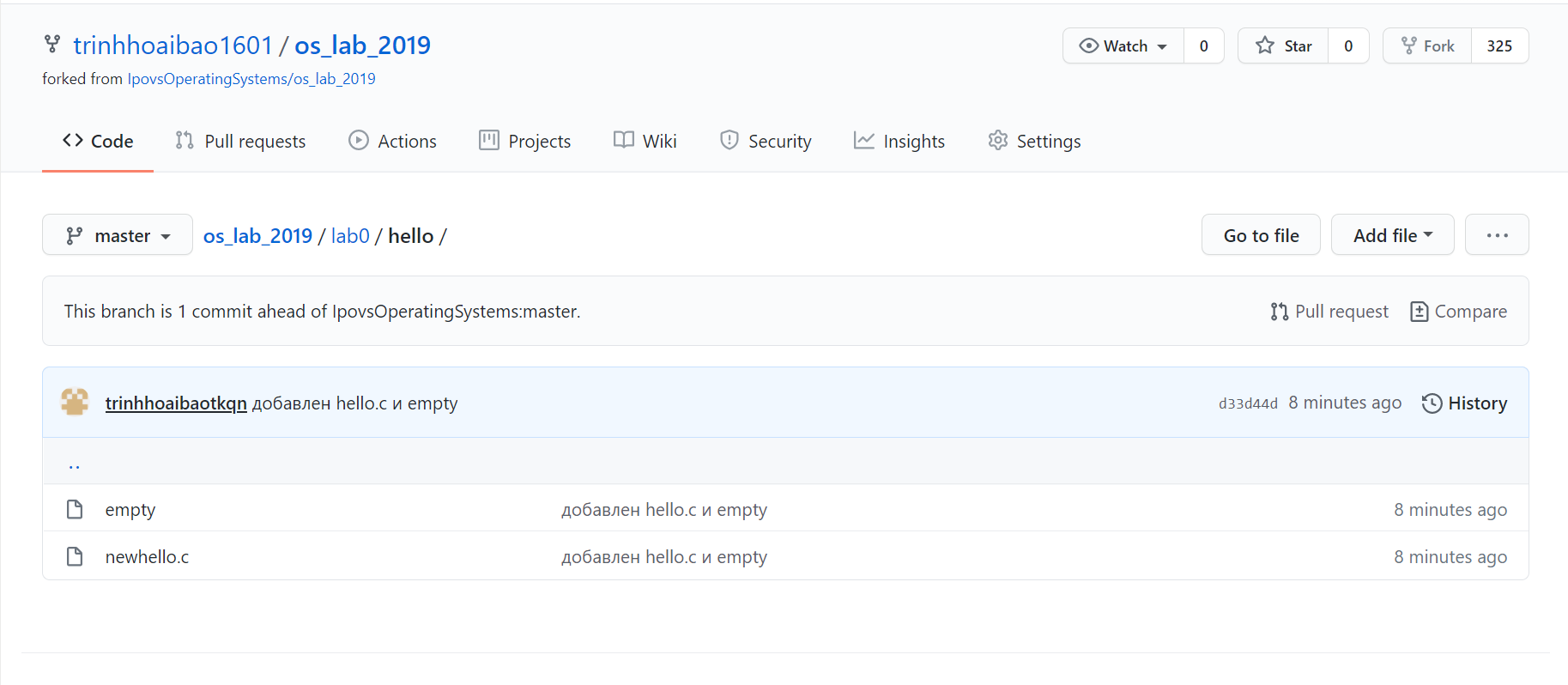


## ***Задание 4:***









# Лабораторная работа №1

## **Задание 1:**

### Необходимые знания

& в shell

Знак «&» заставляет команду выполняться в фоновом режиме. Если команда завершается оператором управления &, оболочка выполняет команду в фоновом режиме в подоболочке. Оболочка не ждет завершения команды, и статус возврата равен 0.

Как работает команда cat

## Команда Cat: Синтаксис

Команда принимает имя файла в качестве аргумента вместе с опциями, уточняющими отдельные операции.

cat [ОПЦИЯ] [ФАЙЛ]

Файл создан, и вы можете начать заполнять его текстом. Если вы хотите добавить несколько строк, просто нажимайте **Enter** в конце каждой строки. Когда вы закончите, нажмите **CTRL+D**, чтобы выйти из файла.

### Создание Файла с Помощью Команды Cat

cat > imiafaila.txt

Чтобы убедиться, что файл действительно создан, просто введите в терминале следующую команду ls:

ls -l

### Просмотр Содержимого Файла с Помощью Команды Cat

cat imiafaila.txt

Для удобного просмотра длинных файлов, используйте параметр **| more,**который позволит скролить выходные данные по одному скрину за раз:

cat filename.txt | more

Также вы можете отобразить содержимое более одного файла. Например, чтобы вывести содержимое всех текстовых файлов.

cat \*.txt

### Редирект Контента с Помощью Команды Cat

Вы можете не только отобразить содержимое файла в консоли, но и перенаправить вывод в другой файл, используя параметр **>**. Вот пример команды:

cat ishodnyifail.txt > failnaznachenija.txt

Если файл назначения не существует, команда создаст его или перезапишет существующий с таким же именем.

Чтобы добавить содержимое в файл назначения, используйте опцию **>>**:

cat ishodnyifail.txt >> failnaznachenija.txt

### Конкатенация («Склеивание») Файлов с Помощью Команды Cat

cat ishodnyifail1.txt ishodnyifail2.txt > failnaznachenija.txt

### Обозначение Конца Строки Командой Cat

cat -E imiafaila.txt

### Нумерация Строк с Помощью Команды Cat в Linux

Эта команда также позволяет отобразить содержимое файла с пронумерованными строками. Реализует эту функцию опция  **-n**:

cat -n imiafaila.txt

### Отображение Непечатаемых Символов

Чтобы показать непечатаемые символы, добавьте к команде cat опцию **-v**, например:

cat -v imiafaila.txt

Чтобы показать только символы табуляции, используйте параметр **-T**:

cat -T imiafaila.txt

Табуляции будут отображаться как **^I**.

### Удаление Пустых Строк

Чтобы удалить пустые повторяющиеся строки и сэкономить место на дисплее, вы можете применить опцию**-s**. Но имейте в виду, что этот параметр удалит только повторяющиеся пустые строки, оставив одну пустую строку.

cat -s imiafaila.txt

### Нумерация Непустых Строк с Помощью Команды Cat в Linux

Параметр **-b** отменяет опцию **-n**:

cat -b imiafaila.txt

### Отображение Файла в Обратном Порядке

tac imiafaila.txt

Как работает команда clear

Она полностью очищает вывод и к предыдущему выводу вы уже вернуться не сможете:

clear

Для того чтобы утилита clear не очищала полностью историю вывода, доступную при прокрутке, а работала аналогично сочетанию клавиш Ctrl+L используйте опцию -x:

clear –x

Как работает команда wc

Анализ файлов - неотъемлемая часть работы с ними. Иногда возникает необходимость подсчитать количество строк или слов в тексте. С этой задачей эффективно справляется команда **wc**

**Синтаксис :**

wc [ОПЦИИ] [ФАЙЛ]

**Опции :**

l lines- Распечатать количество строк.

-w words- Распечатать количество слов.

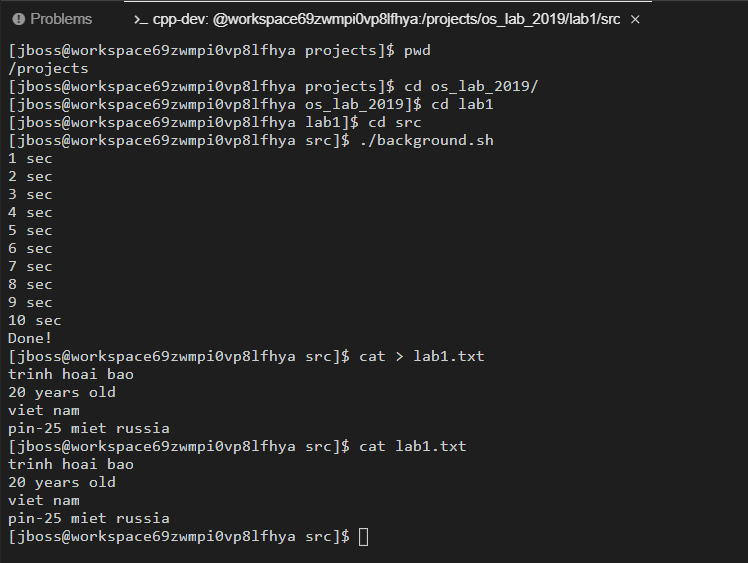
-m chars- Распечатать количество символов.

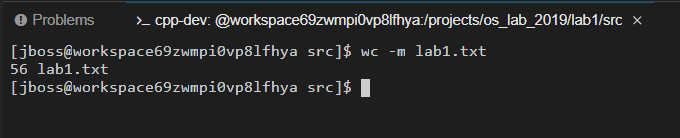
-c bytes- Вывести количество байтов.

-L max-line-length- Вывести длину самой длинной строки

### Задание

* Запустите скрипт background.sh в фоновом режиме.
* Создайте текстовый файл и выведите его содержимое на экран терминала.
* Очистите окно терминала.
* Посчитайте количество символов в файле.





## **Задание 2:**

### Необходимые знания

Как работает команда grep

Синтаксис команды grep при поиске в одном файле выглядит следующим образом:

grep [опции] значение [ФАЙЛ]

* **grep** — команда
* **[опции]** — модификаторы команды
* **значение** — поисковый запрос
* **[ФАЙЛ]** — файл, в котором вы выполняете поиск

Как видите, команда предлагает нам множество опций. Однако наиболее важными и часто используемыми являются параметры:

* **-i** — поиск не будет чувствителен к регистру. То есть, если вы хотите найти слово «автомобиль», написанные как «АВТОМОБИЛЬ» слова тоже будут найдены.
* **-c** — покажет только количество строк, содержащих поисковый запрос
* **-r** — включает рекурсивный поиск в текущем каталоге
* **-n** — выведет номера строк, содержащих поисковый запрос
* **-v** — обратный поиск, выводит только строки, в которых нет указанного поискового запроса

### Поиск слова в текстовом файле

grep запрос файл

* **запрос** — слово, которое вы ищете
* **файл** — файл, в котором вы ищете слово

### Поиск слово, игнорируя регистр

grep -i запрос файл

### Количество найденных слов по целевому запросу

grep -c запрос файл

### Поиск по нескольким ключевым словам

grep запрос1 файл | grep запрос2 файл

Команда работает очень просто. Сначала мы ищем **запрос1**, а затем с помощью вертикальной черты и grep продолжаем поиск  второго слова — **запрос2**.

### Поиск слова в нескольких файлах

Также вы можете искать слово в нескольких файлах одновременно с помощью одной команды:

grep -l слово\_которое\_ищете ./\*

Файлы, содержащие слово, которое вы искали, будут выведены в терминал.

Как работает редирект в bash

Программы Linux автоматически подключаются к 3 потокам данных при их выполнении:

stdin (стандартный ввод): это поток, который помещает данные в программу для обработки (0).

stdout (стандартный вывод): этот поток используется для вывода данных на дисплей после завершения выполнения без ошибок (1).

stderr (стандартная ошибка): этот поток имеет те же функции, что и stdout, однако он используется только для печати сообщений об ошибках, и в то же время сигнал об ошибке также отправляется в операционную систему (2).

**Типы редиректов**

**Перенаправляет в файл**

Один из самых простых способов навигации: выходные данные будут сохраняться в файл, а не выводиться на дисплей.

Чтобы перенаправить команду в файл, Linux предоставляет пользователю два синтаксиса: <(запись содержимого в файл с начальной точки, если файл уже имеет содержимое, затем перезаписать его) и << (аналогично <, но вместо перезаписывая старый контент, он будет писать с конца старого контента)

**Перенаправить из файла**

Оставшееся простое перенаправление вместе с Redirect to file похоже на чтение данных из файла и использование этих данных в качестве входных данных для программы.

Единственный символ этого подхода - <

**Перенаправляет на stderr**

Обычно, когда команда терпит неудачу, информация об ошибке всегда отображается на экране вместе с выходными данными.

Linux предоставляет символ 2> для вывода сообщений об ошибках в файл вместо отображения на экране.

**Перенаправить на другую команду**

Особое перенаправление, которое дает вывод одного оператора другому в качестве ввода.

Использование | перенаправить

Что такое pipe в bash

Он соединяет стандартный вывод команды слева со стандартным вводом команды справа. То есть он создает специальный файл, канал, который открывается как предназначение записи для левой команды и как источник чтения для правой команды.

Специальные девайсы в \*nix системах (/dev/null, /dev/full и т.д.)

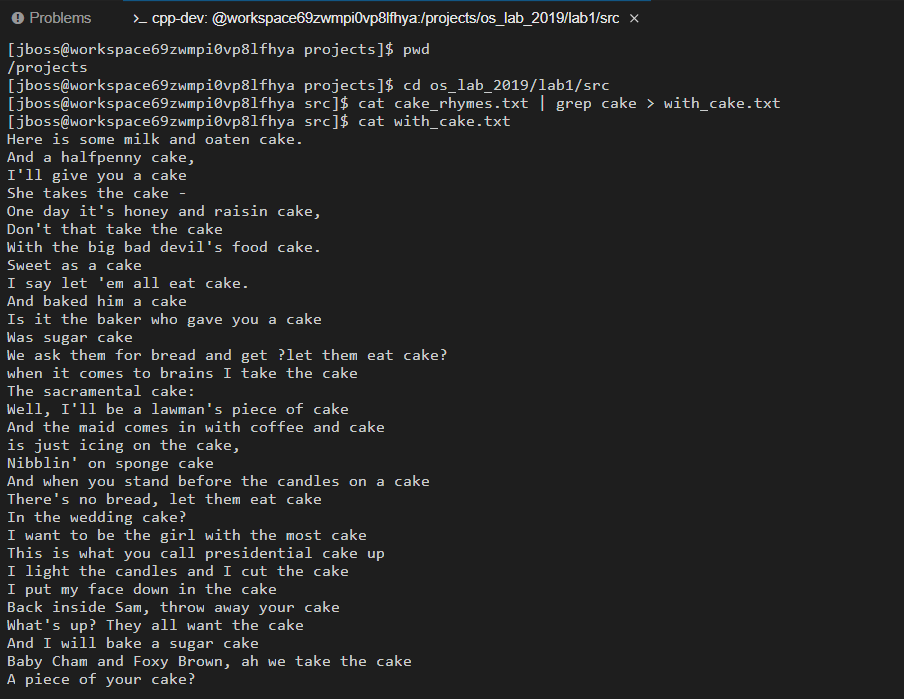
Узлы устройств в Unix-подобных системах не обязательно должны соответствовать физическим устройствам. Узлы, не имеющие такого соответствия, образуют группу псевдоустройств. Они предоставляют различные функции, выполняемые операционной системой. Некоторые из наиболее часто используемых (символьных) псевдоустройств включают:

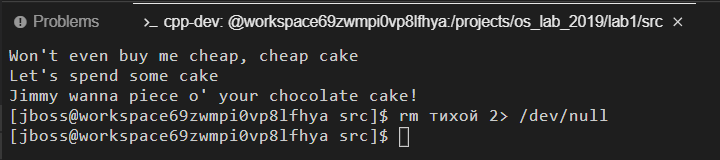
/ dev / null - принимает и отбрасывает весь записанный в него ввод; обеспечивает индикацию конца файла при чтении из.

/ dev / zero - принимает и отбрасывает весь записанный в него ввод; производит непрерывный поток нулевых символов (байтов с нулевым значением) в качестве вывода при чтении из.

/ dev / full - создает непрерывный поток нулевых символов (байтов с нулевым значением) в качестве вывода при чтении из и генерирует ошибку ENOSPC («диск заполнен») при попытке записи в него.

/ dev / random - генерирует байты, генерируемые криптографически безопасным генератором псевдослучайных чисел ядра. Его точное поведение зависит от реализации, и иногда также предоставляются такие варианты, как / dev / urandom или / dev / arandom.





## **Задание 3:**

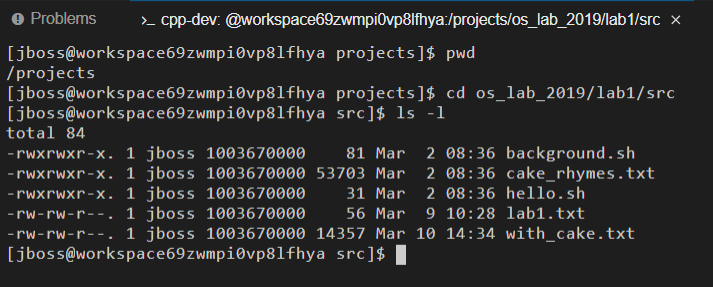
### Необходимые знания

Права в linux

## Просмотр прав доступа на файлы и каталоги в Linux.

Для того, **чтобы просмотреть права доступа на файлы и каталоги в нужной директории**, переходим в неё и выполняем команду:

ls –l



Для наших файлов и каталогов **вывелась подробная информация**

Слева отображены **права доступа на файл и директорию** вида:

**rwxrwxr-x**

**Чтобы просмотреть права доступа на определенный файл, на определенный каталог**

ls -l имя\_нужного\_файла

ls -l имя\_нужной\_директории

**Чтобы просмотреть права доступа на директории и файлы рекурсивно внутри каталога**

ls -Rl имя\_нужной\_директории

## Расшифровка "символьной формы" прав доступа на каталоги и файлы в Linux

Данное сообщение состоит из 10 символов

## Первый символ обозначает тип данных

В большинстве случаев это будет:

|  |  |
| --- | --- |
| **-** | обычный файл; |
| **d** | директория/каталог/папка (directory); |
| **l** | символическая ссылка (link). |

Но может быть следующим:

|  |  |
| --- | --- |
| **b** | файл блочного устройства (block); |
| **c** | файл символьного устройства; |
| **s** | доменное гнездо (socket); |
| **p** | именованный канал (pipe). |

## Следующие 9 символов обозначают права доступа

* **Первая группа из трех символов обозначает** права доступа владельца файла или директории (u - user).
* **Вторая группа из трех символов обозначает** права доступа на файл или директорию для системной группы (g - group).
* **Третья группа из трех символов обозначает** права доступа на файл или директорию для всех остальных (o - other).

У этих трёх групп **одинаковая комбинация символов**, то есть:

rwx

|  |  |
| --- | --- |
| **r** | read, то есть, право доступа на чтение файла или директории. |
| **w** | write, то есть, право на изменение и удаление файла или директории. |
| **x** | eXecute, то есть, право на запуск файла как программы или вход в директорию. |

Всегда располагаются в таком порядке:

rwx

Если вместо какого-то символа идёт тире (минус), к примеру:

r-x

то это значит, что отсутствуют права на изменение и удаление файла или директории.

Или так:

r--

# Определение владельца и группы файла или директории.

ls –l

* **в третей колонке** идет имя владельца файла;
* **в четвертой колонке** идет название группы.

# Команда chmod

# Изменение прав доступа на файлы и каталоги в Linux в символьном режиме

chmod personsOperatorRights имя\_файла\_или\_имя\_директории

Вместо **persons** нужно использовать совокупность символов или один из:

|  |  |
| --- | --- |
| **u** | user, пользователь, владелец файла или директории. |
| **g** | group, группа файла или директории. |
| **o** | other, все другие. |
| **a** | all, то есть, все вместе взятые - владелец, группа и все остальные. |

Этот символ обозначает **субъект**, которому будут назначены, удалены или изменены права.

Вместо **Operator**, может быть один из следующих символов:

|  |  |
| --- | --- |
| **+** | "плюс", добавляем нужные права. |
| **-** | "минус", удаляем нужные права. |
| **=** | "равно", устанавливаем нужные права. |

Этот символ обозначает **оператор**, от которого зависит, будут ли добавлены, удалены или установлены нужные права, которые последуют за **оператором**.

Вместо Rights перечисляем символы прав доступа:

|  |  |
| --- | --- |
| **r** | read - чтение. |
| **w** | write - запись. |
| **x** | eXecute - выполнение. |

**Для рекурсивного изменения прав в каталоге используется параметр -R.**

То есть, если нужно изменить права всех файлов и папок внутри указанной нами директории, то после **chmod** просто указываем параметр **-R**. Выглядит это следующим образом:

chmod -R personsOperatorRights имя\_директории

**Важно.**

Если Вы не являетесь владельцем файла или директории или у вас нет прав на изменение файла, то нужно будет использовать права суперпользователя:

sudo chmod personsOperatorRights имя\_файла\_или\_имя\_директории

# Переменные окружения

Переменные окружения являются величинами с определенными значениями. Некоторые переменные среды предоставляются с системными предустановками, а другие устанавливаются пользователем непосредственно в командной строке или при запуске других сценариев.

Переменная окружения представляет собой строку символов, содержащую информацию, используемую одним или несколькими приложениями.

**Просмотр доступных переменных окружения**

set

Или

env

Или

export

**Как установить переменную среды в Linux**

**Переменные окружения, созданные в скрипте доступны только. Дочерние процессы не будут иметь доступа к заданным или измененным значениям. Позволяет дочерним процессам просматривать запрошенные значения с помощью команды вывода.**

|  |  |
| --- | --- |
| **Отображает значение определенной переменной** | echo $SHELL |
| **Экспорт нового значения переменной** | export VARIABLE=value VARIABLE=value export VARIABLE |
| **Добавить постоянную переменную** | Отредактируйте файл $ HOME / .bashrc и добавьте его в строку  export VARIABLE=value  Потом source ~/.bashrc или ~/.bashrc или просто введите команду bash. |

Переменная окружения **"$ HOME"**

Переменная окружения **"$ PATH"**

Переменная окружения **"$ SHELL"**

Переменная окружения **"$ PS1"**

# Команда date в bash

Распечатать текущую дату

date

Получите значение времени Unix в любой момент времени

date +%s

Установить дату, время

date –s “Форматированная строка даты”

date –s “07 August 2014 22:22:22”

Параметры формата даты

% a => Вывести краткое название дня недели

% A => Распечатать полное название дня недели

% b => Вывести краткое название месяца

% B => Распечатать полное название месяца

% c => Распечатать дату и время

% d => Распечатать дату месяца

# Что такое shebang

В системах Unix (Linux, macOS) shebang - это первая строка файла сценария, которая начинается с символа #!

После #! - это имя программы-интерпретатора, которая будет загружать и запускать код в скрипте, со следующим синтаксисом

#!interpreter [optional-arg]

* interpreter это путь к интерпретатору
* [optional-arg] настройки, если необходимо, зависит от interpreter

Задание

