Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 2

По дисциплине «Операционные системы»

На тему «Знакомство с командной оболочкой операционной системы»

Выполнил:

Студент 3 курса 6 группы

Филипюк Илья Андреевич

Преподаватель: асс. Уласевич Н.И.

2025, Минск

# **Введение**

Командная оболочка – это программа, которая предоставляет текстовый интерфейс для взаимодействия пользователя с операционной системой. Она принимает команды, введённые с клавиатуры, выполняет их или запускает программы, и отображает результаты в командной строке (терминале). Оболочки также позволяют создавать скрипты – файлы с последовательностью команд для автоматизации задач, и обладают функциями программирования.

Цель работы: получить практические навыки в работе с командными оболочками такими как cmd, PowerShell, bash.

Задачи исследования:

* рассмотреть основные команды, используемые в cmd;
* рассмотреть основные команды, используемые в bash;
* рассмотреть основные команды, используемые в Powershell для написания собственных скриптов;
* выполнить задания, используя рассмотренные команды;
* узнать, что такое bat-файлы, для чего они нужны и научиться создавать их;
* узнать, что такое sh-файлы, для чего они нужны и научиться создавать их;

Используемые инструменты:

* cmd;
* bash;
* PowerShell;
* Microsoft Hyper-V;

# **1. Изучение основ cmd в Windows**

В данном разделе рассматриваются основные команды cmd, часто используемые на практике, а также выполнение заданий лабораторной работы для закрепления материала.

# **1.1 Предварительная подготовка**

Перед изучением команд в домашнем каталоге пользователя User-ee4aed31 был создан каталог OS2. На рисунке 1.1 показан результат создания данной директории.

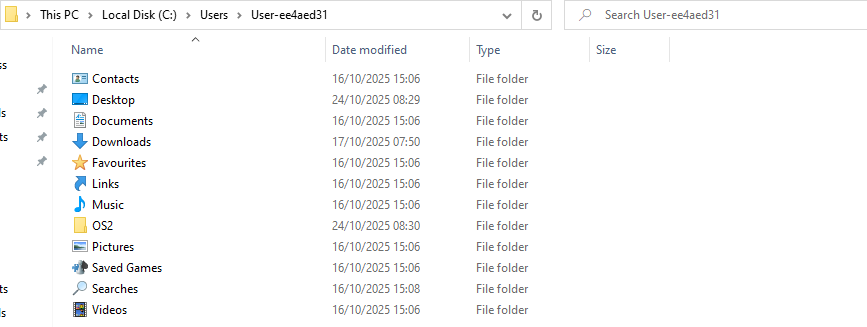


Рисунок 1.1 – Создание директории OS2 в домашнем каталоге пользователя User-ee4aed31

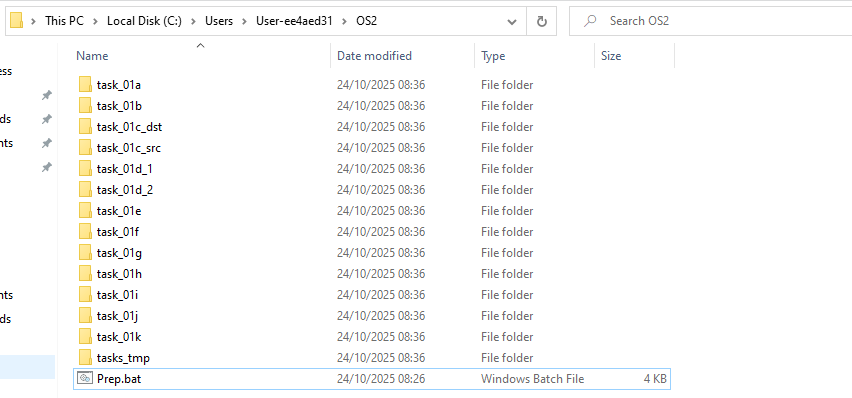
Затем в нем был выполнен файл Prep.bat для формирования структуры папок, необходимой для выполнения заданий. На рисунке 1.2 представлен результат выполнения Prep.bat файла.

Рисунок 1.2 – Результат выполнения файла Prep.bat в директории OS2

Затем был открыт cmd в данном каталоге путём указания слова cmd в адресной строке проводника (рисунок 1.3)

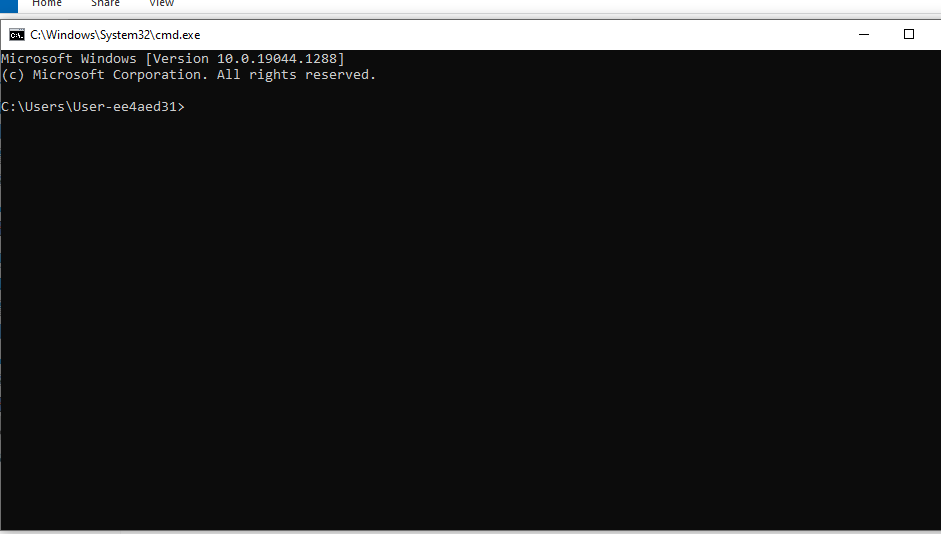


Рисунок 1.3 – Результат открытия cmd в директории OS2

# **1.2 Команда cd**

Команда cd (change directory) предназначена для изменения текущего рабочего каталога. Она поддерживает абсолютные и относительные пути, а также специальные обозначения:

* .. – переход в родительский каталог;
* . – текущий каталог;
* \ – переход к корню диска.

Для работы с путями, содержащими пробелы, рекомендуется заключать их в двойные кавычки. Ключ /d позволяет переключаться между дисками.

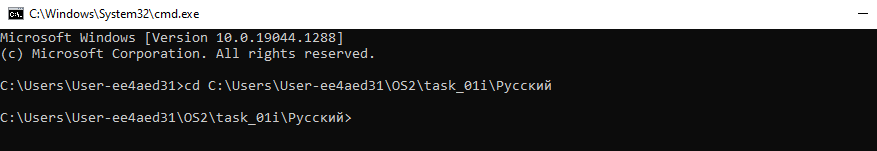
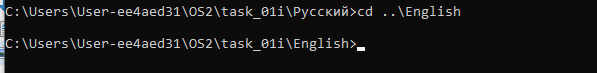


Рисунок 1.4 – Результат выполнения команды cd с указанием абсолютного пути

Рисунок 1.5 – Результат выполнения команды cd с указание относительного пути

# **1.3 Команда copy**

Команда сору выполняет копирование файлов. Ее синтаксис: сору <источник> <назначение>. Поддерживаются шаблоны, например, \*.txt (Рис. 1.6).

Основные ключи:

* /A (копирование текстовых файлов);
* /V (проверка целостности данных);
* /Y и /-Y (автоматический ответ перезаписи или запрос);
* /Z (возобновление копирования после сетевого сбоя).

В случае, если выполнение команды потребует перезаписи существующих файлов в папке назначения и не будет указан ключ /Y, то команда запросит подтверждение перезаписи необходимых файлов (Рис. 1.7).

Для копирования с изменением имени файла следует указать новое имя в параметре назначения (Рис. 1.8).

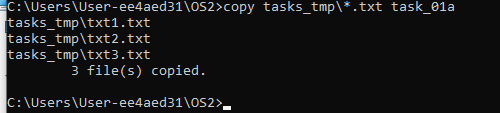


Рисунок 1.6 – Результат выполнения команды copy с шаблоном файла

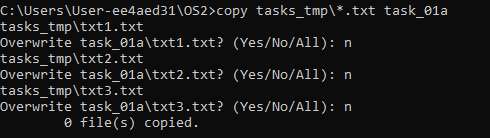
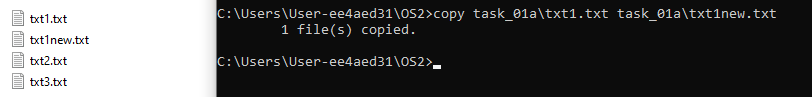
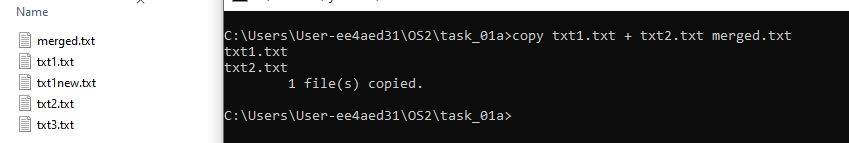
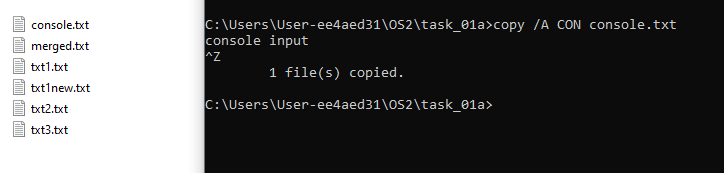


Рисунок 1.7 – Запрос подтверждения на перезапись файлов при повторном выполнении команды copy

Рисунок 1.8 – Результат выполнения команды copy со сменой имени файла

Команда copy также позволяет объединить файлы с помощью символа + (Рис. 1.9) и использовать консоль (CON) как источник или приёмник данных (Рис. 1.10 – Рис. 1.11).

Риснок 1.9 – Результат объединения файлов с помощью команды copy

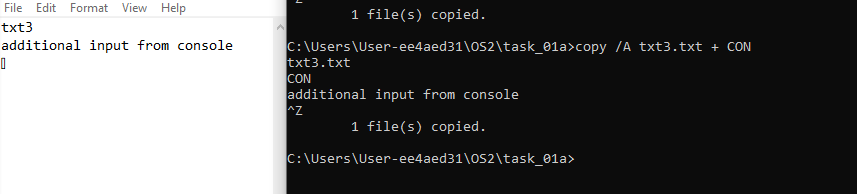
Рисунок 1.10 – Результат выполнения команды copy, где в качестве источника данных выступает консоль

Рисунок 1.11 – Результат выполнения команды copy для записи дополнительного текста, вводимого с консоли, в файл

# **1.3 Команда xcopy**

Команда xcopy предоставляет расширенные возможности копирования. Основные ключи:

* /S (копирование непустых папок и подпапок);
* /E (копирование всех папок, включая пустые);
* /H (включение скрытых и системных файлов);
* /A (копирование файлов с атрибутом «архивный»);
* /L (предварительный просмотр файлов для копирования).

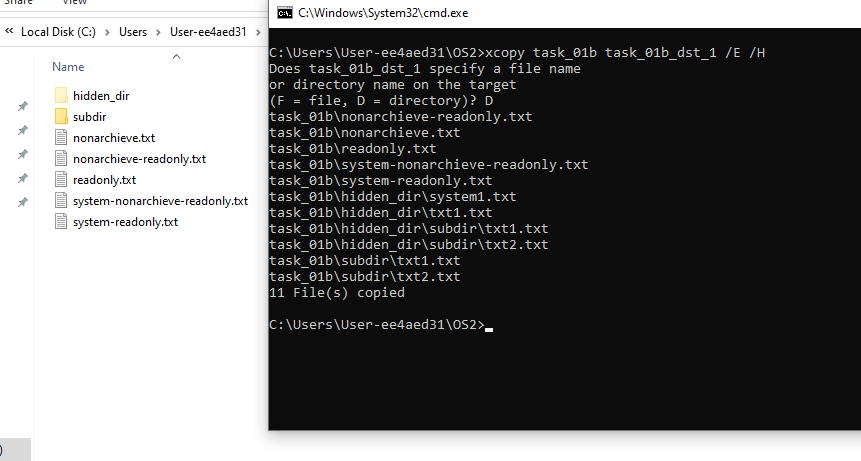
Примеры использования с ключами /H /E и /A /K представлены на Рис. 1.12 и 1.13 соответственно.

Рисунок 1.12 - Результат выполнения команды xcopy с использованием ключей /H и /E

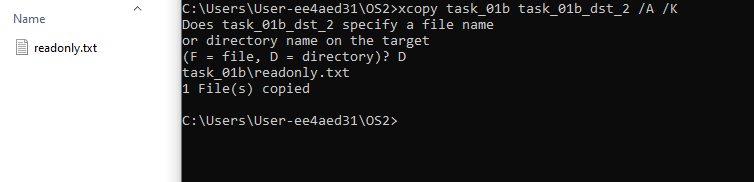


Рисунок 1.13 – Результат выполнения команды xcopy с использованием ключей /A и /K

# **1.4 Команда move**

Команда move предназначена для перемещения или переименования файлов и папок. Поддерживает шаблоны, например, \*.txt (Рис. 1.14). Для очистки каталога можно использовать цикл (Рис. 1.15).

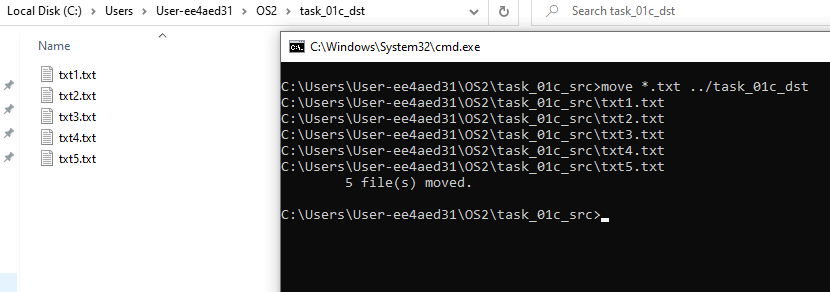
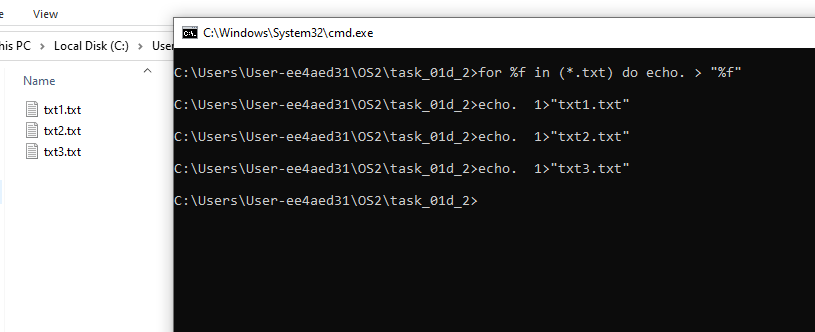
Рисунок 1.14 – Результат выполнения команды move с применением шаблона

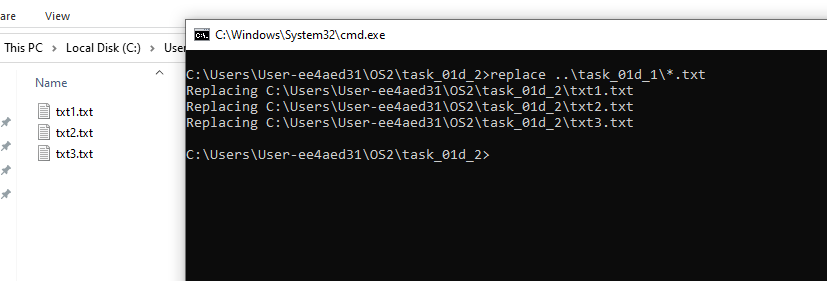
Рисунок 1.15 – Результат выполнения команды очистки файлов

# **1.5 Команда replace**

Команда replace обновляет или добавляет файлы (Рис. 1.16).

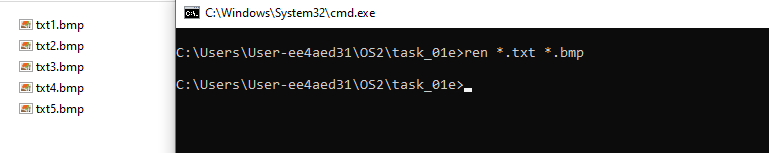
Основные ключи:

* /A (добавление отсутствующих файлов);
* /U (обновление только устаревших файлов);
* /S (рекурсивный поиск и замена).

Рисунок 1.16 – Результат выполнения команды replace

# **1.6 Команда ren**

Команда ren изменяет имена файлов и каталогов. Поддерживает шаблоны, например, смена расширений \*.txt на \*.bmp (Рис. 1.17). Также позволяет одновременно менять имя и расширение (Рис. 1.18).



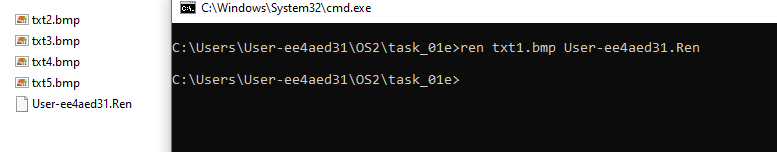
Рисунок 1.17 – Результат выполнения команды ren для смены расширений файлов

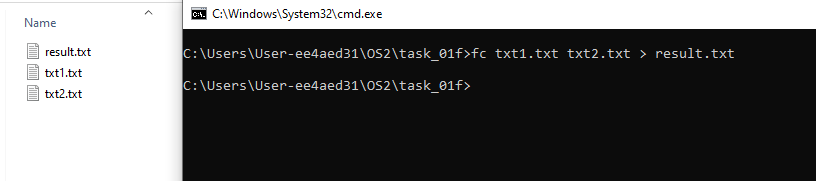
Рисунок 1.18 – Результат выполнения команды ren для смены имени и расширения файла одновременно

# **1.7 Команда fc**

Команда fc сравнивает содержимое файлов (Рис. 1.19).

Основные ключи:

* /A (сокращенный вывод различий);
* /B (двоичное сравнение);
* /L (текстовый режим);
* /N (отображение номеров строк).

Рисунок 1.19 – Результат выполнения команды fc для сравнения двух файлов

# **1.8 Команда del (erase)**

Команда del удаляет файлы (Рис. 1.20 — 1.23).

Основные ключи:

* /P (запрос подтверждения для каждого файла);
* /Q (отключение запросов);
* /S (рекурсивное удаление в подпапках);
* /A (фильтрация по атрибутам).

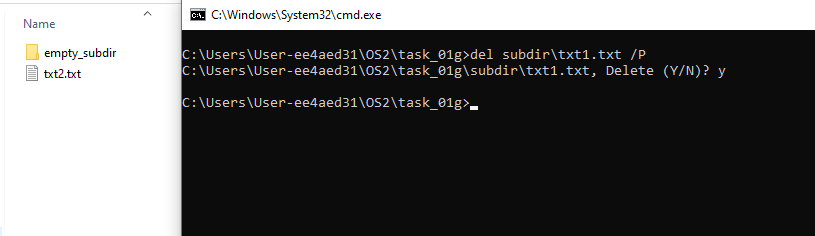


Рисунок 1.20 – Результат выполнения команды del для удаления файла с запросом подтверждения

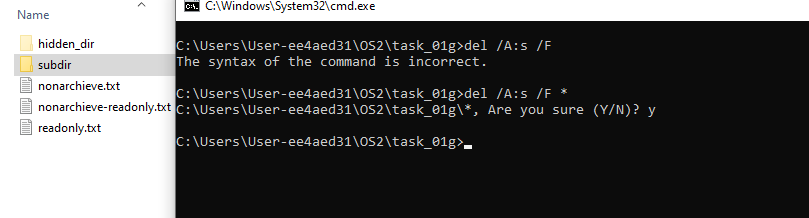
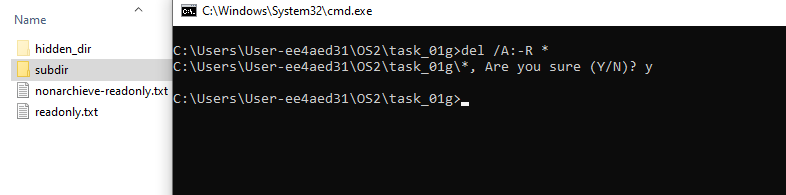


Рисунок 1.21 – Результат выполнения команды del для удаления всех файлов с атрибутом «Системный»

Рисунок 1.22 – Результат выполнения команды del для удаления всех файлов кроме файлов с атрибутом «Только для чтения»

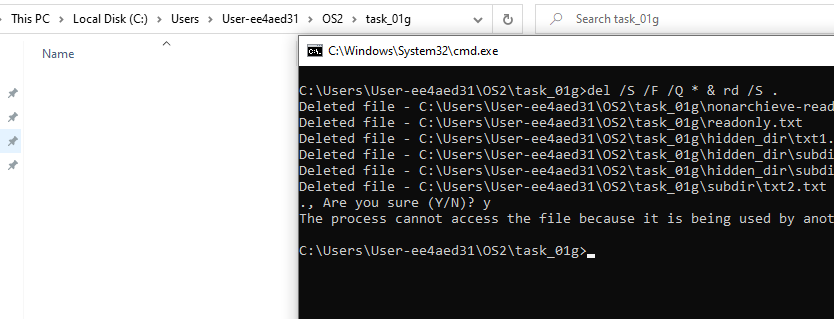


Рисунок 1.23 – Результат выполнения команды del совместно с rd для удаления всех оставшихся файлов и дочерних каталогов

# **1.9 Команда dir**

Команда dir выводит содержимое каталога (Рис. 1.24—1.25).

Основные ключи:

* /A (фильтрация по атрибутам);
* /O (сортировка);
* /S (рекурсивный обход подкаталогов);
* /P (постраничный вывод).

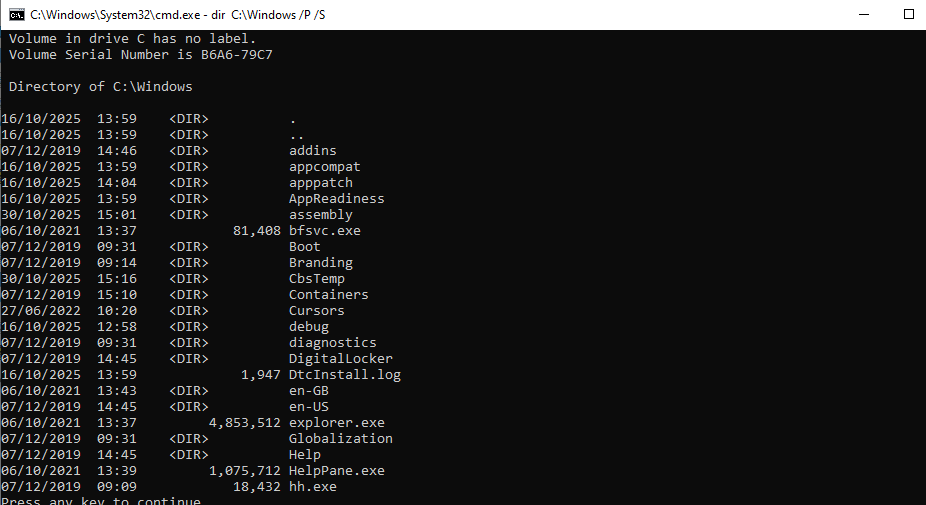
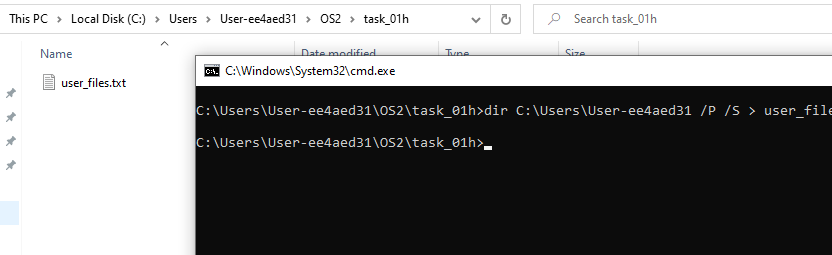
Рисунок 1.24 – Результат выполнения команды dir с ключами /P /S для постраничного вывода содержимого каталога

Рисунок 1.25 – Результат выполнения команды dir с ключами /P /S и с записью в файл

# **1.10 Команда md**

Команда md создает каталоги, включая промежуточные, если они отсутствуют (Рис. 1.26–1.27).

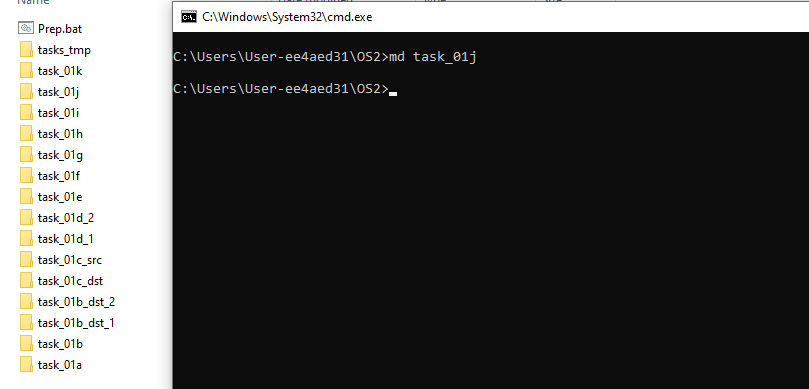
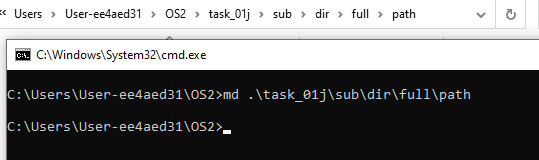
Рисунок 1.26 – Результат выполнения команды md для создания каталога

Рисунок 1.27 – Результат выполнения команды md для создания иерархии папок в предварительно созданном каталоге

# **1.11 Команда rd**

Команда rd удаляет каталоги (Рис. 1.28—1.29).

Основные ключи:

* /S (удаление дерева каталогов);
* /Q (отключение подтверждения);

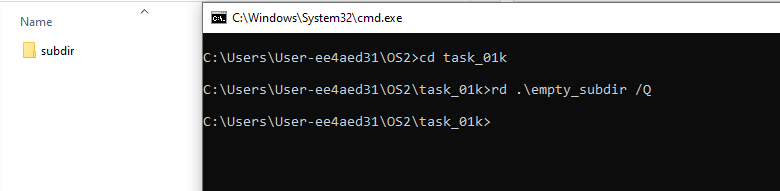
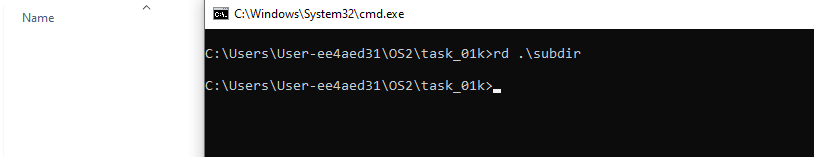


Рисунок 1.28 – Результат выполнения команды rd для удаления пустого каталога

Рисунок 1.29 – Результат выполнения команды rd для удаления непустого каталога

# **1.12 Команда type**

Данная команда предназначена для просмотра содержимого файлов прямо в командной строке. В качестве параметра данная команда принимает название (путь) файла.

# **1.13 Команда systeminfo**

Данная команда предназначена для получения детальной информации о системе, оборудовании и конфигурации Windows. Параметров она не принимает, но у нее есть дополнительные ключи. Вот некоторые мы сейчас и рассмотрим.

Основные ключи:

* /S (удаленный компьютер);
* /FO (формат вывода);
* /U (контекста пользователя);
* /P (пароль пользователя);
* /NH (игнорирование заголовков);

# **1.14 Команда findstr**

Данная команда предназначена для поиска текста в файлах с поддержкой регулярных выражений.

Основные ключи:

* /B (поиск начала по заданному шаблону);
* /I (поиск конца по заданному шаблону);
* /L (использовать шаблон как литерал);
* /I (игнорировать регистр);
* /S (рекурсивный поиск);
* /V (поиск строк, не содержащих шаблон);
* /N (показ номеров строк);
* /M (только полные совпадения);

# **1.15 Операторы**

Для расширения функциональности командной строки используются специальные операторы.

Операторы перенаправления ввода/вывода:

* > – перенаправление вывода с перезаписью файла.
* >> – перенаправление вывода с добавлением в конец файла.

Оператор конвейера (|) передает вывод одной команды на вход другой, позволяя объединять команды в цепочки.

Операторы управления выполнением:

* & – последовательное выполнение команд.
* && – условное выполнение (следующая команда выполняется только при успехе предыдущей).
* || – условное выполнение (следующая команда выполняется только при ошибке предыдущей).
* () – группировка команд.

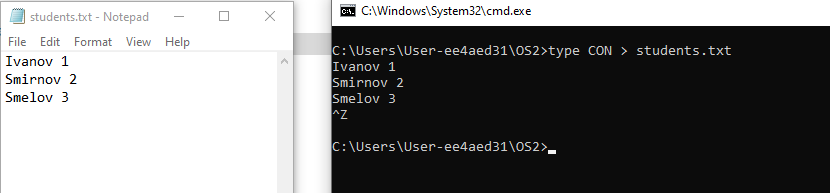


Рисунок 1.30 – Результат выполнения команды type с использованием оператора перенаправления

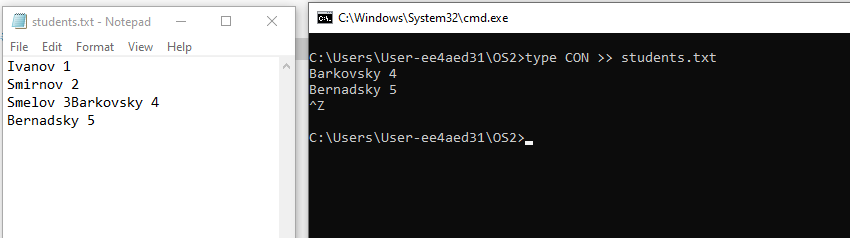
Рисунок 1.31 - Результат выполнения команды findstr с использованием оператора конвейеризации

Рисунок 1.32 – Результат выполнения команды type с использованием оператора перенаправления для записи в конец файла

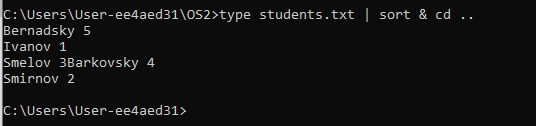
Для вывода в отсортированном порядке нам понадобятся команды type для вывода, sort для сортировки и cd для выхода из папки, а также оператор конвейеризации для того, чтобы вывод type передать команде сортировки, а также оператор соединения команд, чтобы затем еще выполнить команду выхода из папки.

Рисунок 1.33 – Результат выполнения совокупности команд type, sort, cd и операторов | и &

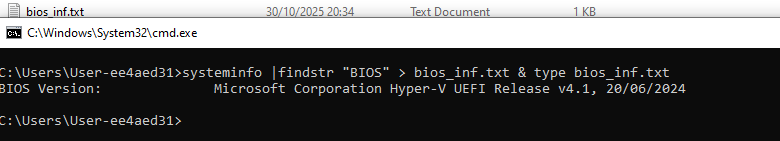
Ну и последним заданием нам требуется получить версию BIOS через файл. Для получения версии BIOS через файл нам понадобится команда systeminfo, чтобы получить всю информацию о системе, команда findstr “BIOS” и команда type для вывода результата на консоль. Используем оператор конвейера, чтобы передать результата вывода информации от systeminfo в findstr, затем оператор перенаправления для записи найденной информации в файл, потом оператор соединения команд, чтобы потом еще вывести эту информацию.

Рисунок 1.34 – Результат выполнения совокупности команд systeminfo, findstr, type и операторов |, & и >

# **1.16 Пакетные файлы (BAT-файлы)**

Пакетный файл (BAT-файл) – это текстовый файл с расширением .bat, содержащий последовательность команд для выполнения в командной строке Windows. Основное назначением является автоматизация рутинных операций.

# **1.17 Системные переменные**

Системные переменные хранят данные о конфигурации системы и пользовательском окружении.

Основные группы переменных:

* системные пути: %SystemRoot%, %ProgramFiles%;
* пользовательские пути: %USERPROFILE%, %APPDATA%, %TEMP%;
* системная информация: %COMPUTERNAME%, %USERNAME%, %OS%.

Для просмотра всех переменных используется команда set. Ключевой является переменная %PATH%, содержащая список каталогов для поиска исполняемых файлов.

# **1.18 Параметры и их расширения**

В BAT-файлах параметры передаются через аргументы командной строки: %0 – имя скрипта, %1-%9 – аргументы, %\* – все аргументы.

Расширения параметров позволяют модифицировать значения преимущественно пути:

* %~f1 (полный путь);
* %~d1 (буква диска);
* %~p1 (путь без имени файла);
* %~n1 (имя файла без расширения);
* %~x1 (расширение файла);
* %~t1 (дата и время изменения);
* %~z1 (размер файла).

# **1.19 Правила именования команд и параметров**

Названия команд чаще всего формируются по следующим принципам:

* аббревиатура: cd (Change Directory), fc (File Compare);
* усечение: del (delete), ren (rename);
* комбинация: xcopy (extended copy), ipconfig (IP configuration);
* прямое указание действия: copy, sort, find.

Параметры команд, как правило, представляют собой одну или несколько букв, указывающих на их функцию: /S (Subdirectories), /Q (Quiet), /A (Attributes).

# **1.20 Написание собственных скриптов**

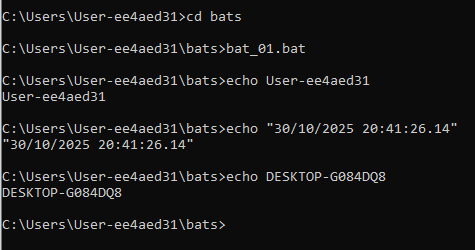
Создаем наш первый скрипт, используя системные переменные для получения данных. В листинге 1.1 представлены команды для вывода требуемой информации. На рисунке 1.35 продемонстрирован результат выполнения.

echo %USERNAME%

echo "%DATE% %TIME%"

echo %COMPUTERNAME%

Листинг 1.1 – Содержимое файла bat\_01.bat

Рисунок 1.35 – Результат выполнения bat\_01.bat

Следующий скрипт выводит метаданные о самом скрипте при помощи расширения параметров. В листинге 1.2 представлены команды, которые как раз и позволяют нам вывести нужную информацию для второго задания. На рисунке 1.36 продемонстрирован результат выполнения.

|  |
| --- |
| echo %~nx0  echo %~t0  echo %~f0 |

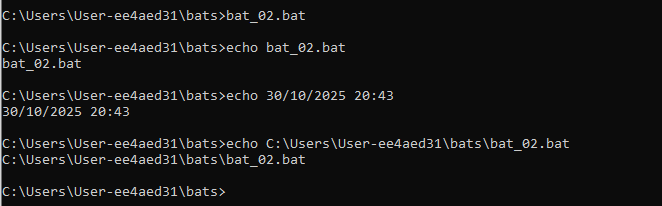
Листинг 1.2 – Содержимое файла bat\_02.bat

Рисунок 1.36 – Результат выполнения файла bat\_02.bat

В третьем задании необходимо работать с параметрами, передаваемые скрипту. В листинге 1.3 представлены команды для вывода наших параметров заданным образом. На рисунке 1.37 продемонстрирован результат выполнения.

|  |
| --- |
| echo %\*  echo %1  echo %2  echo %3  echo %4 |

Листинг 1.3 Содержимое bat-файла bat\_03.bat

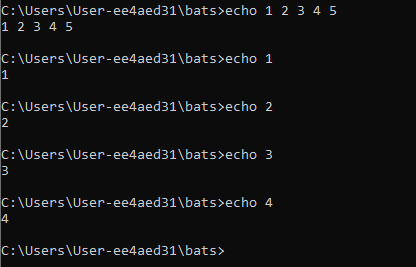


Рисунок 1.37 – Результат выполнения файла bat\_03.bat

В четвертом задании требуется вычислить заданные арифметические выражения. Для этого мы будем использовать команду set с ключом /a, которая позволяет нам задавать значения переменным. В листинге 1.4 представлены команды для выполнения данного задания. На рисунке 1.38 продемонстрирован результат выполнения.

|  |
| --- |
| set /a a=%1  set /a b=%2  set /a c=%3  set /a res1=a-b  echo %a% - %b% = %res1%  set /a res2=a+b  echo %a% + %b% = %res2%  if %b% equ 0 (     echo "Error: attempt to divide by zero!"  ) else (     set /a res3=c/b     echo %c% / %b% = %res3%  )  set /a res4=c\*b     echo %c% \* %b% = %res4%  set /a res5=(a-b)\*(b-a)  echo (%a% - %b%) \* (%b% - %a%) = %res5% |

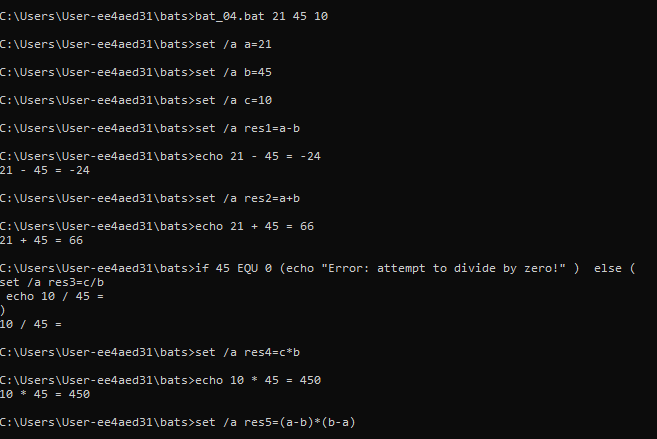
Листинг 1.4 Содержимое bat-файла bat\_04.bat

Рисунок 1.38 – Результат выполнения файла bat\_04.bat

В последнем скрипте требуется написать программу для создания или удаления файла с проверкой корректности аргументов и существования файлов с выводом сообщений при неправильном вводе. В листинге 1.5 представлен полный скрипт для данного задания. На рисунке 1.39 продемонстрирован результат выполнения.

|  |
| --- |
| if "%1" == "" if "%2" == "" (echo "Help: set first param to work mode (create, delete), second to a file name."  goto :eof)  if "%1" == "" (echo "Error: work mode param is missing"  goto :eof)  if not "%1" == "create" if not "%1" == "delete" (echo "Help: wrong work mode input(try create/delete)"  goto :eof)  if "%2" == "" (echo "Error: file name is missing"  goto :eof)  if "%1" == "create" if exist "%2" (echo "Error: file already exists"  goto :eof) else (echo. > "%2"  goto :eof)  if "%1" == "delete" if not exist "%2" (echo "Error: no file was found"  goto :eof) else (del "%2") |

Листинг 1.5 Содержимое bat-файла bat\_05.bat

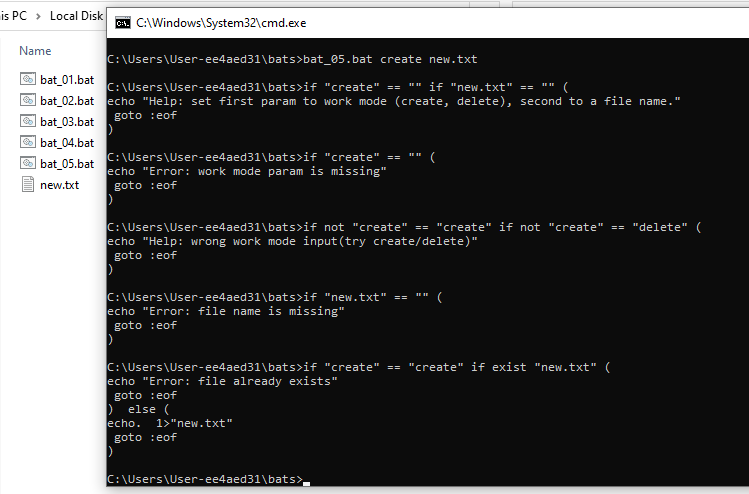


Рисунок 1.39 – Результат выполнения файла bat\_05.bat

# **2. Некоторые основы PowerShell и написание скриптов**

PowerShell – это современная кроссплатформенная оболочка командной строки и язык сценариев, разработанный Microsoft. В отличие от традиционной командной строки (cmd), PowerShell предлагает объектно-ориентированный подход и глубокую интеграцию с платформой .NET.

Ключевые характеристики:

* объектно-ориентированность: Команды возвращают объекты .NET, а не текст, что упрощает обработку и передачу данных;
* кроссплатформенность: работает в операционных системах Windows, Linux и macOS;
* мощный язык сценариев: предоставляет возможности для автоматизации сложных задач;
* интеграция с .NET: позволяет использовать всю мощь библиотек .NET Framework.

# **2.1 Написание собственных скриптов**

В качестве первого задания нам было необходимо написать интерактивный калькулятор для вычисления математических выражений. В приложении А представлен скрипт, написанный для выполнения требуемых задач.

Основные функции:

* Input-Number – обрабатывает вводимые в консоль данные;
* Read-Number – обрабатывает вводимые в консоль числовые данные;
* Read-Int – обрабатывает вводимые в консоль целочисленные данные;
* To-Radians – преобразует введённые данные из градусов в радианы;
* From-Radians – преобразует введённые данные из радиан в градусы;
* Show-Menu – выводит меню доступных операций;
* Do-Trigonometry – обрабатывает введённые тригонометрические функции и проводит вычисления;
* Do-Logarythm – обрабатывает введённые логарифмические функции и проводит вычисления;

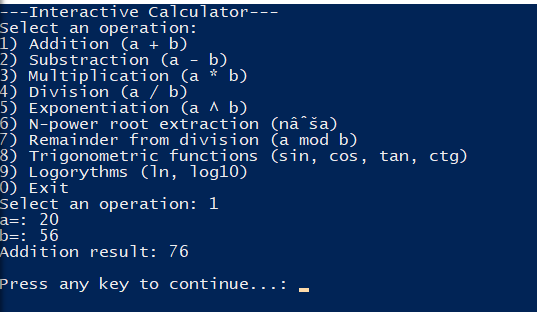
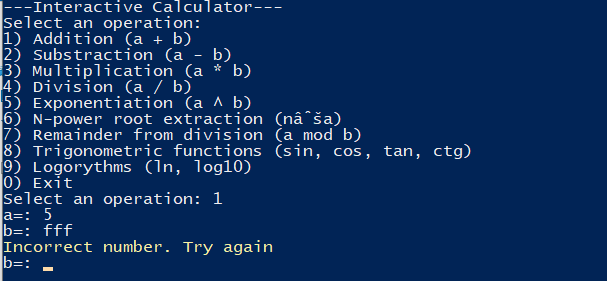
Для обработки ошибок используем структуру try-catch.

Рисунок 2.1 – Результат выполнения скрипта калькулятора

Теперь проверим различные сценарии, которые могут привести к ошибкам.

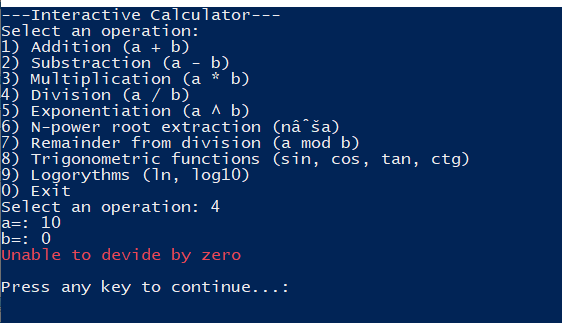
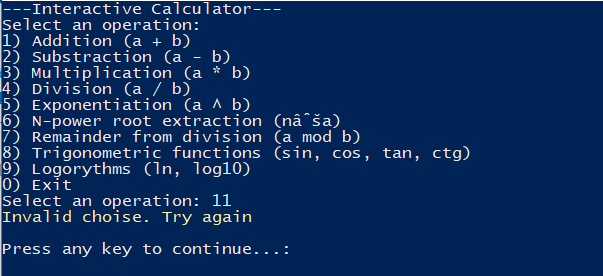
Рисунок 2.2 – Тестирование скрипта с некорректным вводом

Рисунок 2.3 – Тестирование скрипта с делением на ноль

Рисунок 2.4 – Тестирование скрипта с вводом неверного номера операции

Для следующего задания была разработана программа по преобразованию вводимого показателя температуры в иные системы счисления. В приложении Б представлен скрипт, написанный для выполнения требуемых задач.

Основные функции:

* Parse-TempInput– обрабатывает вводимые в консоль данные;
* Convert-Temperature – преобразовывает данные в необходимый вид;

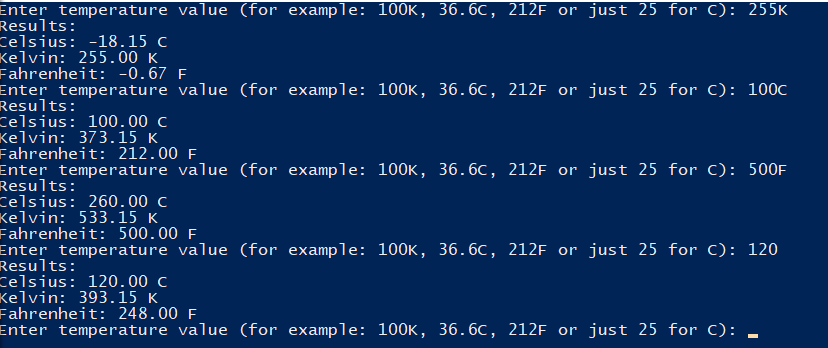
Для обработки ошибок используем структуру try-catch.

Рисунок 2.5 – Результат выполнения скрипта перевода температур

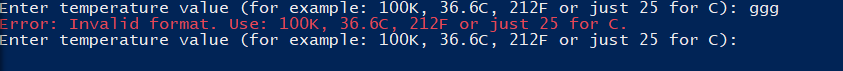


Рисунок 2.6 – Тестирование некорректного ввода

# **3. Изучение основ bash в Linux**

Bash – это командная оболочка для для Unix-подобных операционных систем (таких как Linux, macOS). Она предоставляет интерфейс командной строки (CLI) для выполнения задач, таких как навигация по файлам, управление процессами и настройка системы. Кроме того, Bash поддерживает создание скриптов – файлов с последовательностями команд, которые используются для автоматизации рутинных задач.

# **3.1 Предварительная подготовка**

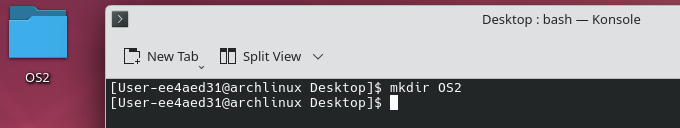
Перед изучением команд необходимо создать каталог OS2 для дальнейшего выполнения лабораторной работы (Рис. 3.1).

Рисунок 3.1 – Создание директории OS2 на Linux

# **3.2 Команда cd**

Команда cd (change directory) в Bash функционально аналогична своей реализации в Windows, но имеет дополнительные возможности:

* cd или cd ~ (переход в домашнюю директорию);
* cd – (возврат в предыдущую рабочую директорию);
* cd ~username (переход в домашний каталог указанного пользователя).

На Рисунке 3.2 демонстрируется использование абсолютных и относительных путей.