**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

A logo of a triangle

Description automatically generated**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

|  |  |
| --- | --- |
| Институт | компьютерных наук |
| Кафедра | автоматизированных систем управления |

ЛАБОРАТОРНАЯ РАБОТА №4

По дисциплине "Операционные системы Linux"

На тему "Создание и использование сценариев (скриптов) в Linux"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ПИ-22-1 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Первеева Е. Ю. |
| Руководитель | |  |  |  |
| канд.техн.наук, доцент  ученая степень, ученое звание | |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Кургасов В.В. |

Липецк, 2024 г.

Оглавление

[**Цель работы** 3](#_Toc182519863)

[**Ход работы** 4](#_Toc182519864)

[**1. Часть I** 4](#_Toc182519865)

[**2. Часть Ⅱ** 7](#_Toc182519866)

[**Вывод** 17](#_Toc182519867)

[**Контрольные вопросы** 17](#_Toc182519868)

**Цель работы**

Изучить основные возможности языка программирования высокого уровня Shell, получить навыки написания и использования скриптов.

**Ход работы**

**1. Часть I**

1. Используя команды ECHO, PRINTF, вывести информационные сообщения на экран.

ECHO используется для прямого вывода текста, если использовать команду без флага -e, то такие управляющие символы как перенос строки (\n) будут распознаны как обычный текст и выведены на экран.

PRINTF более гибкая, чем echo, так как поддерживает форматирование текста. Для переноса на новую строку требует явного указания новой строки (\n), поддерживает форматные строки и плейсхолдеры (например, %s, %d).

Файл с кодом скрипта представлен на рисунке 1.

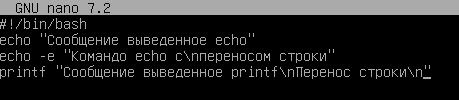


Рисунок 1 - Код скрипта

Результат работы исполнения файла представлен на рисунке 2.

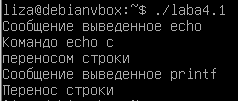


Рисунок 2 - Результат работы скрипта

2. Присвоить переменной Aцелочисленное значение. Посмотреть значение переменной A.

Код скрипта представлен на рисунке 3.

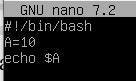


Рисунок 3 - Код скрипта

Результат работы скрипта представлен на рисунке 4.



Рисунок 4 - Результат работы скрипта

3. Присвоить переменной Bзначение переменной A. Посмотреть значение переменной B.

Код скрипта представлен на рисунке 5.



Рисунок 5 - Код скрипта

Результат работы скрипта представлен на рисунке 6.



Рисунок 6- Результат работы скрипта

4. Присвоить переменной Cзначение "путь до своего каталога". Перейти в этот каталог с использованием переменной.

Код скрипта представлен на рисунке 7. Он выводит сначала путь к текущему каталогу, потом значение переменной C, переходим в каталог, указанный в переменной C(cd) и снова выводим путь к текущему каталогу, чтобы убедиться что он изменился.

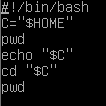


Рисунок 7- Код скрипта

Результат работы скрипта представлен на рисунке 8.



Рисунок 8- Результат работы скрипта

5. Присвоить переменной Dзначение "имя команды". Выполнить эту команду, используя значение переменной.

Код скрипта представлен на рисунке 9. Присвоим переменной значение имя команды, например ls и выведем его на экран.



Рисунок 9- Код скрипта

Результат работы скрипта представлен на рисунке 10.



Рисунок 10- Результат работы скрипта

6. Присвоить переменной Eзначение "имя команды", а именно, команды просмотра содержимого файла, посмотреть содержимое переменной. Выполнить эту команду, используя значение переменной.

Код скрипта представлен на рисунке 11.

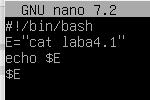


Рисунок 11- Код скрипта

Результат работы скрипта представлен на рисунке 12.

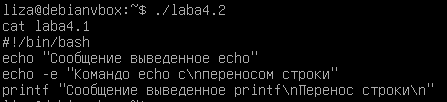


Рисунок 12- Результат работы скрипта

7. Присвоить переменной Fзначение "имя команды", а именно, сортировки содержимого файла. Выполнить эту команду, используя значение переменной.

Код скрипта представлен на рисунке 13.



Рисунок 13- Код скрипта

Результат работы скрипта представлен на рисунке 14.

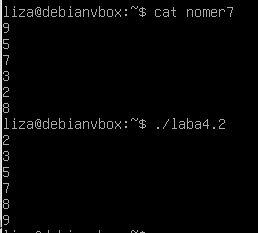


Рисунок 14- Результат работы скрипта

## **2. Часть Ⅱ**

1. Программа запрашивает значение переменной, а затем выводит значение этой переменной.

Код программы представлен на рисунке 15.

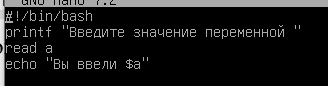


Рисунок 15- Код скрипта

Результат работы программы представлен на рисунке 16.



Рисунок 16- Результат работы скрипта

2. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введённой переменной.

Код программы представлен на рисунке 17.

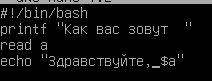


Рисунок 17- Код скрипта

Результат работы программы представлен на рисунке 18.



Рисунок 18- Результат работы скрипта

3. Программа запрашивает значение двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран.

Код программы представлен на рисунке 19.

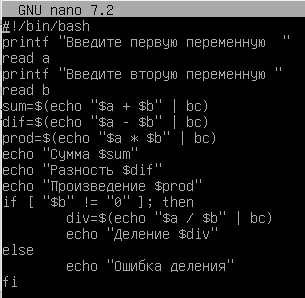


Рисунок 19- Код скрипта

Результат работы программы представлен на рисунке 20.

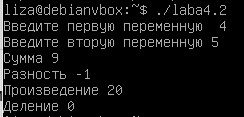


Рисунок 20- Результат работы скрипта

4. Вычислить объём цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран.

Код программы представлен на рисунке 21.

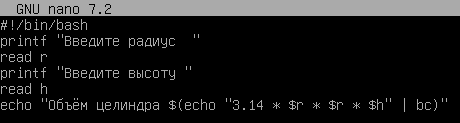


Рисунок 21- Код скрипта

Результат работы программы представлен на рисунке 22.

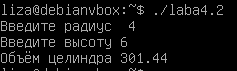


Рисунок 22- Результат работы скрипта

2.5. Используя позиционные параметры, отобразить имя программы, количество аргументов командной строки, значение каждого аргумента командной строки.

Код программы представлен на рисунке 23.

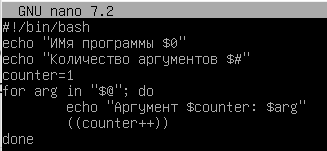


Рисунок 23- Код скрипта

Результат работы программы представлен на рисунке 24.

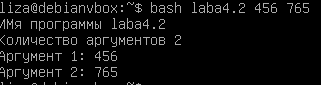


Рисунок 24- Результат работы скрипта

2.6. Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки.

Код программы представлен на рисунке 25.

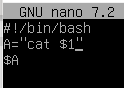


Рисунок 25- Код скрипта

Результат работы программы представлен на рисунке 26.

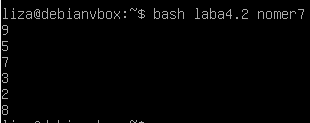


Рисунок 26- Результат работы скрипта

7. Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога поэкранно.

Код программы представлен на рисунке 27.

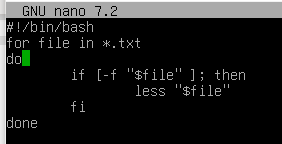


Рисунок 27- Код скрипта

8. Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения.

Код программы представлен на рисунке 28.

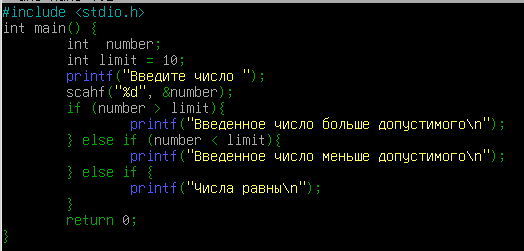


Рисунок 28- Код скрипта

Результат работы программы представлен на рисунке 29.

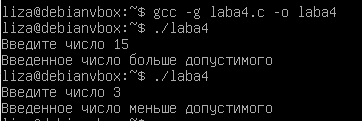


Рисунок 29- Результат работы скрипта

9. Программой запрашивается год, определяется, високосный ли он. Результат выводится на экран.

Код программы представлен на рисунке 30.

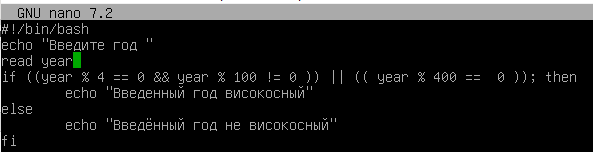


Рисунок 30- Код скрипта

Результат работы программы 9 представлен на рисунке 31.

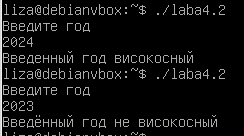


Рисунок 31- Результат работы скрипта

10. Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются.

Код программы представлен на рисунке 32.

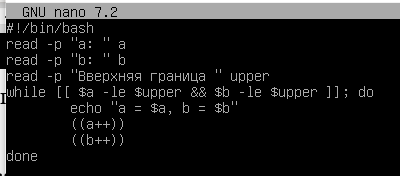


Рисунок 32- Код скрипта

Результат работы программы представлен на рисунке 33.

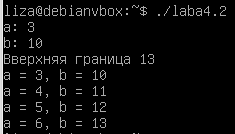


Рисунок 33- Результат работы скрипта

11. В качестве аргумента командной строки указывается пароль. Если пароль введён верно, постранично отображается содержимое каталога /etc. Код программы представлен на рисунке 34.

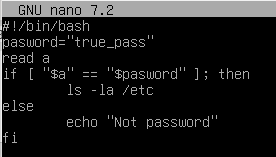


Рисунок 34- Код скрипта

Результат работы программы представлен на рисунке 35.



Рисунок 35- Результат работы скрипта

12. Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдаётся соответствующее сообщение.

Код программы представлен на рисунке 36.

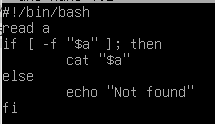


Рисунок 36- Код скрипта

Результат работы программы представлен на рисунке 37.

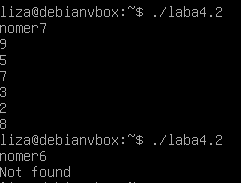


Рисунок 37- Результат работы скрипта

13. Если файл есть каталог и этот каталог можно читать, просматривается содержимое этого каталога. Если каталог отсутствует, он создаётся. Если файл не есть каталог, просматривается содержимое файла.

Код программы представлен на рисунке 38.

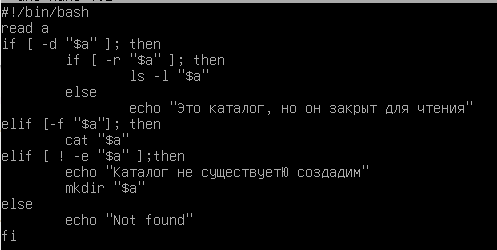


Рисунок 38- Код скрипта

Результат работы программы представлен на рисунке 39.

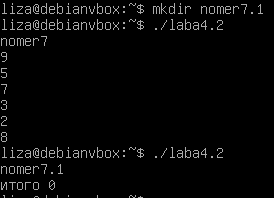


Рисунок 39- Результат работы скрипта

14. Анализируются атрибуты файла. Если первый файл существует и используется для чтения, а второй файл существует и используется для записи, то содержимое первого файла перенаправляется во второй файл.

Код программы представлен на рисунке 40.

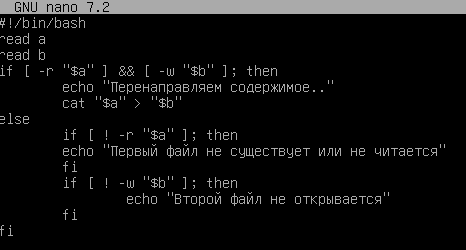


Рисунок 40- Код скрипта

Результат работы программы представлен на рисунке 41.

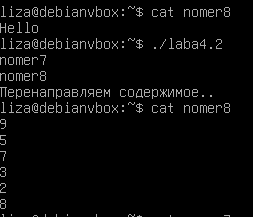


Рисунок 41- Результат работы скрипта

15. Если файл запуска программы найден, программа запускается. Код программы представлен на рисунке 42.

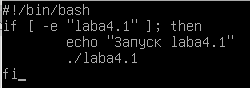


Рисунок 42- Код скрипта

Результат работы программы представлен на рисунке 43.

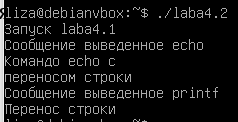


Рисунок 43- Результат работы скрипта

16. В качестве позиционного параметра задаётся файл, проанализируйте его размер. Если размер файла больше нуля, содержимое файла сортируется по первому столбцу по возрастанию, отсортированная информация помещается в другой файл.

Код программы представлен на рисунке 44.

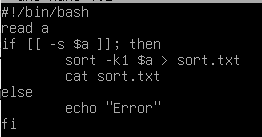


Рисунок 44- Код скрипта

Результат работы программы представлен на рисунке 45.

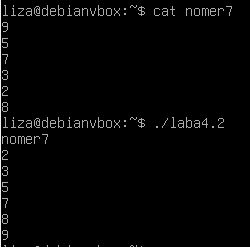


Рисунок 45- Результат работы скрипта

# **Вывод**

Были изучены основные возможности языка программирования высокого уровня Shell, получены навыки написания и использования скриптов.

# **Контрольные вопросы**

**1. В чём отличие пользовательских переменных от переменных среды?**

Пользовательские переменные и переменные среды отличаются по своему назначению и области видимости:

Пользовательские переменные — это переменные, которые создаются и используются только в текущем скрипте или сеансе. Они не наследуются дочерними процессами.

Переменные среды — это системные переменные, доступные для всех процессов, запускаемых в текущем сеансе, включая дочерние. Они обычно устанавливаются операционной системой или настраиваются пользователем и передаются всем запущенным программам и скриптам.

**2. Математические операции в SHELL.**

В Shell доступны основные математические операции, такие как сложение, вычитание, умножение и деление. Вот основные способы выполнения операций:

1. Арифметическое расширение $(( ... )):

2. Команда expr (устарела, но иногда используется):

3. Команда bc — для операций с числами с плавающей точкой:

result=$(echo "5.5 + 3.2" | bc)

**3. Условные операторы в SHELL.**

Основные условные конструкции:

1. Оператор if:

if [ условие ]; then

fi

2. Оператор case — используется для сравнения переменной с множеством значений.

3. Тернарный оператор (выполняется внутри скобок $(...)):

echo $((условие ? значение\_если\_истинно : значение\_если\_ложно))

4. Логические операторы для комбинирования условий:

&& (И) — выполняет следующую команду, если предыдущая завершилась успешно.

|| (ИЛИ) — выполняет следующую команду, если предыдущая завершилась неудачно.

**4. Принципы построения простых и составных условий.**

Простое условие состоит из одного логического выражения. Оно может включать: сравнение чисел, строк, проверка файлов.

Составные условия объединяют несколько простых условий с помощью логических операторов && (И) и || (ИЛИ):

1. Оператор И (&&): Все условия должны быть истинными.

2. Оператор ИЛИ (||): Достаточно одного истинного условия.

3. Комбинирование условий: Можно использовать несколько уровней условий с () для группировки

5. **Циклы в SHELL.**

1. Цикл for - Используется для перебора списка значений или файлов.

2. Цикл while - Выполняет команды, пока условие истинно.

3. Цикл until - Цикл until аналогичен while, но выполняется, пока условие ложно.

4. Цикл select - Часто используется для создания меню. select ждет, пока пользователь выберет один из вариантов.

Эти циклы можно использовать с операторами break (для выхода) и continue (для перехода к следующей итерации) в зависимости от условий.

6. **Массивы и модули в SHELL.**

Массивы: В bash массивы индексируются с нуля и поддерживают как одно-, так и многомерные структуры. Пример:

array=(1 2 3 4)

echo ${array[0]} # выводит 1

Модули: В bash нет прямой концепции модулей, но можно использовать внешние скрипты или функции для организации кода.

**7. Чтение параметров командной строки.**

Параметры передаются в скрипт через $1, $2, ..., $N для позиционных аргументов

**8. Как различать ключи и параметры?**

Ключи: Обычно начинаются с дефиса (например, -f, --file).

Параметры: Значения, передаваемые после ключей или позиции (например, имя файла или аргумент).

**9. Чтение данных из файлов.**

Для чтения данных из файлов используется команда cat, while read, или перенаправление:

while read line; do

echo $line

done < input.txt

**10. Стандартные дескрипторы файлов.**

Стандартные дескрипторы:

0 — стандартный ввод (stdin)

1 — стандартный вывод (stdout)

2 — стандартная ошибка (stderr)

Пример перенаправления: command > output.txt 2>&1 # перенаправление stdout и stderr в файл

**11. Перенаправление вывода.**

Перенаправление вывода в Shell позволяет перенаправить стандартный вывод (stdout), стандартную ошибку (stderr) или оба потока в файл, другой процесс или устройство, такое как /dev/null.

command > file.txt # перенаправление stdout

command 2> error.txt # перенаправление stderr

**12. Подавление вывода.**

Подавление вывода можно сделать с помощью перенаправления в /dev/null

**13. Отправка сигналов скриптам.**

Для отправки сигналов используются команды kill или trap:

kill — отправляет сигнал процессу: kill -SIGTERM <PID> # отправка сигнала TERM

trap — ловит сигнал в скрипте.

**14. Использование функций.**

Функции в Shell позволяют структурировать код, улучшая его читаемость и повторное использование:

function greet() {

echo "Hello, $1"

}

greet "Alice" # вызов функции с параметром

**15. Обработка текстов (чтение, выбор, вставка, замена данных).**

Чтение: Использование команд, таких как cat, while read, для чтения данных из файлов.

Выбор: Использование команд вроде grep, awk, sed для фильтрации данных.

Вставка и замена данных: Использование sed для замены текста

**16. Отправка сообщений в терминал пользователя.**

Используется команда echo или printf для вывода сообщений в терминал.

**17. BASH и SHELL – синонимы?**

Bash — это один из видов оболочки (shell), наиболее распространённый в Unix-подобных системах. Он предоставляет больше функциональности по сравнению с другими оболочками (например, sh).

Shell — это общее название для командных оболочек, которые взаимодействуют с операционной системой через командную строку. Bash является одним из наиболее популярных shell-ов, но есть и другие (например, zsh, fish, ksh).

**18. PowerShell в операционных системах семейства Windows: назначение и особенности.**

PowerShell — это командная оболочка, ориентированная на администрирование Windows. Она предоставляет более мощные возможности по сравнению с традиционными командными оболочками Windows (cmd), включая:

1. Объектно-ориентированное управление через .NET-объекты.

2. Возможности для автоматизации и управления через скрипты.

3. Командлеты, которые позволяют выполнять различные задачи (например, работа с реестром, сетевыми настройками, файлами).

4. Поддержка работы с различными типами данных и форматов вывода, таких как JSON, XML, и CSV.