Отчет по лабораторной работе №12

Программирование в командном процессоре ОС UNIX. Командные файлы

Чигладзе Майя Владиславовна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы

2 Порядок выполнения лабораторной работы

2.1 Задание 1.

Задача: Написать скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации необходимо узнать, изучив справку.

1. Код/файл (рис. 1).

```
/home/mvchigladze/Рабочий стол/study_2022-2023_os-intro/Labs/lab11/backup/first.sh
#!/bin/bash

##################### переменные ##############################

### Установите правильное местоположение источника и места назначения
SOURCE_DIR="/home/mvchigladze/Рабочий стол/study_2022-2023_os-intro/Labs/lab11"
DEST_DIR="/home/mvchigladze/Рабочий стол/study_2022-2023_os-intro/Labs/lab11/backup"
TMP_FILE=/tmp/copyfileslist.txt

### Установите имя пользователя и имя группы, чтобы установить права на скопированные файлы

### Установите для CHANGE_OWNERSHIP значение 1, чтобы изменить владельца, или 0, чтобы не изменять его.
CHANGE_OWNERSHIP=1
USER='root'
GROUP='root'


########## Не редактируйте, пока не потребуется #################

### Проверка, существует ли исходный каталог

### Скрипт остановится, если источника не существует

if [ -d "${SOURCE_DIR}" ]; then
echo "Source directory found"
else
echo "Source directory not found. Please check above variables are set correctly"
echo "script exited"
exit 1
fi

### Проверка, существует ли каталог назначения
### Скрипт создаст каталог назначения, если он не существует.
### Если не удалось создать каталог, скрипт будет прерван

if [ -d "${DEST_DIR}" ]; then
echo "Destination directory found, all ok"
else
echo "Destination directory not found, creating now"
mkdir -p "${DEST_DIR}"
if [ $? -eq 0 ]; then
echo "Successfully created destination directory."
else
echo "Failed to create destination directory. Script exited"
exit 1
fi
```

Рис. 2.1: Код

2. Результат (рис. 2).

```
root@mvchigladze:/home/mvchigladze/Рабочий стол/study_2022-2023_os-intro/Labs/lab11# chmod +x first.sh
root@mvchigladze:/home/mvchigladze/Рабочий стол/study_2022-2023_os-intro/Labs/lab11# ./first.sh
Source directory found
Destination directory found, all ok
File ./first.sh successfully copied.
```

Рис. 2.2: Результат 1

3. Результат (рис. 3).

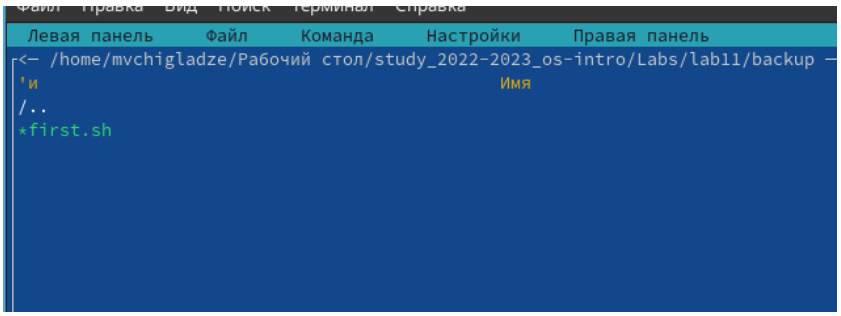

Рис. 2.3: Результат 2

2.2 Задание 2.

Задача: Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов

1. Код/файл (рис. 4).

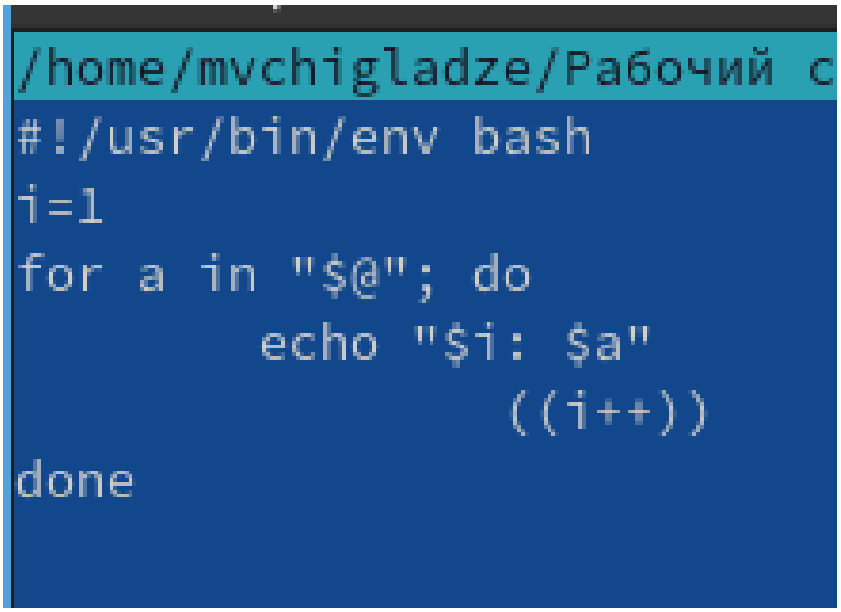

Рис. 2.4: Код

2. Результат (рис. 5).

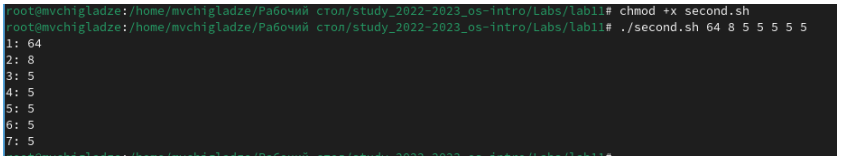

Рис. 2.5: Результат

2.3 Задание 3.

Задача: Написать командный файл — аналог команды ls (без использования самой этой команды и команды dir). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.

1. Код/файл (рис. 6).

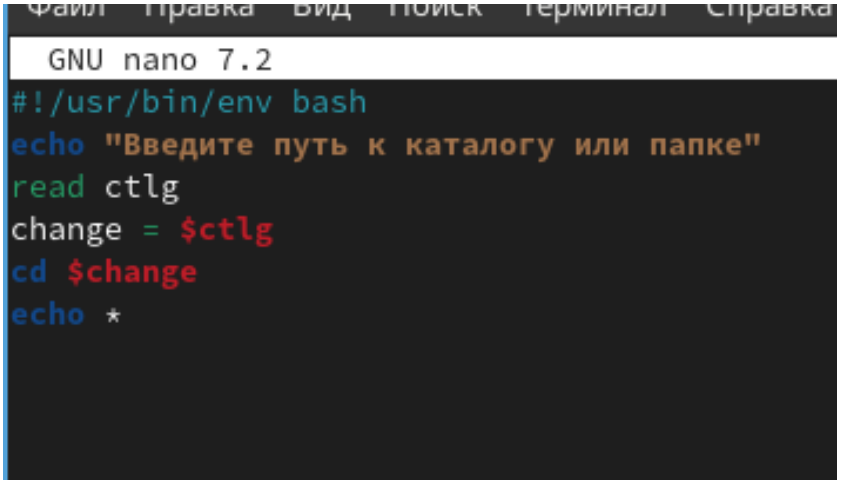

Рис. 2.6: Код

2. Результат (рис. 7).

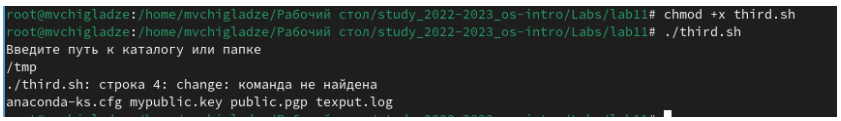

Рис. 2.7: Результат

2.4 Задание 4.

Задача: Написать командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки

1. Код/файл (рис. 8).

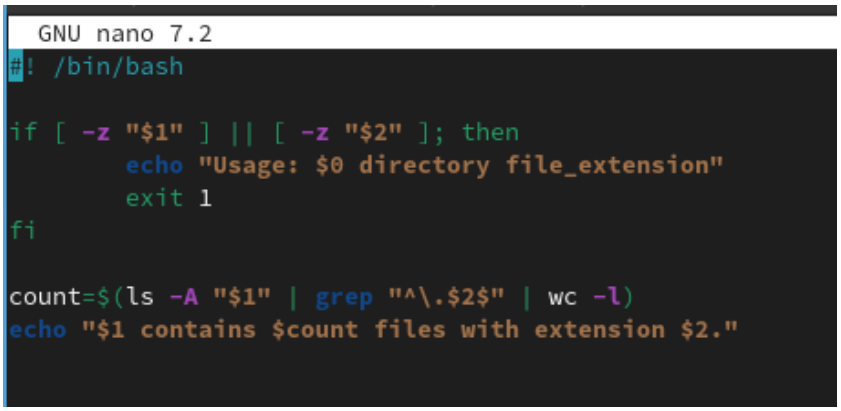

Рис. 2.8: Код

2. Результат (рис. 9).

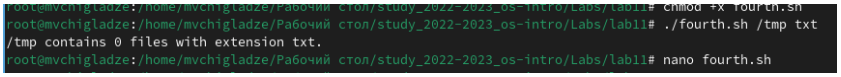

Рис. 2.9: Результат

3 Ответы на контрольные вопросы

1. Командная оболочка - это программа, которая обеспечивает интерфейс между пользователем и операционной системой. Она принимает команды от пользователя и передает их на выполнение операционной системе. Примеры командных оболочек включают в себя Bash, Zsh, Fish и PowerShell. Они отличаются синтаксисом команд, возможностями настройки и поддержкой дополнительных функций.
2. POSIX (Portable Operating System Interface) - это стандарт, определяющий интерфейс между операционной системой и приложениями. Он описывает, как приложения должны взаимодействовать с операционной системой для обеспечения переносимости кода между различными платформами.
3. Переменные и массивы определяются с помощью символов доллара ($), знака равенства (=) и имени переменной. Массивы создаются с использованием квадратных скобок ([]).
4. Оператор let используется для выполнения арифметических операций и присвоения значений переменным. Оператор read считывает ввод пользователя.
5. Арифметические операции включают сложение (+), вычитание (-), умножение (*) и деление (/).
6. Операция (( )) используется для вычисления арифметических выражений.
7. Стандартные имена переменных включают $PWD (текущий рабочий каталог), $RANDOM (случайное число), $? (код возврата последней команды) и $PATH (путь поиска для исполняемых файлов).

8. Метасимволы - это символы, имеющие специальное значение в контексте регулярных выражений или командных строк.
9. Для экранирования метасимволов необходимо использовать обратный слэш ().
10. Командные файлы создаются с помощью текстового редактора и сохраняются с расширением .sh. Для запуска командного файла нужно выполнить его в командной строке.
11. Функции определяются с использованием ключевого слова function, за которым следует имя функции и список параметров.
12. Команда “ls -la” может использоваться для определения типа файла (обычный файл или каталог).
13. Команды set, typeset и unset используются для определения, изменения и удаления переменных соответственно.
14. Параметры передаются в командной строке через разделитель (обычно пробел).
15. Специальные переменные включают $*, $?, $!, $@ и $#.

4 Выводы

В ходе лабораторной работы, я изучила основы программирования в оболочке ОС UNIX/Linux. Научилась писать небольшие командные файлы

Список литературы