

Федеральное государственное автономное  
образовательное учреждение высшего образования  
Российский Университет Дружбы Народов им. Патриса Лумумбы  
Математический университет имени Никольского  
Факультет Физико-математических и Естественных наук  
Кафедра Прикладной математики и информатики

Отчет по лабораторной работе № 9  
“ Программирование в командном процессоре ОС UNIX. Командные файлы”

Выполнил:  
Студент группы НПМбв-02–20  
Сарновский Даниил

Москва  
2024 год

**Цель работы:** изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

**Задачи:**

1. 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.
2. 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.
3. 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.
4. 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.

**Теоретическое введение**

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
- С-оболочка (или csh) — надстройка на оболочке Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна.

## Выполнение лабораторной работы:

Для начала напишем скрипт, который при запуске будет делать резервную копию самого себя в другую директорию backup в домашнем каталоге. Файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar.

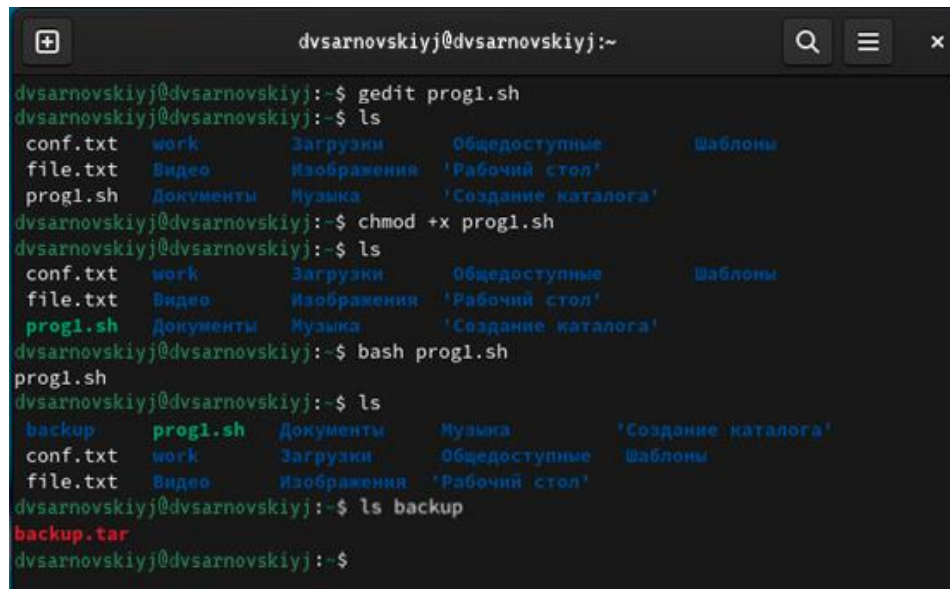
Для этого введем gedit prog1.sh, чтобы вызвать gedit редактор и создать файл для скрипта. После ввожу скрипт необходимый заданию.



```
1 #!/bin/bash
2
3 mkdir -p ~/backup
4 tar -cvf ~/backup/backup.tar prog1.sh
```

Рис. 1. Архивная копия

Сделаем файл исполняемым с помощью chmod +x prog1.sh и проверяем, сработал ли скрипт.



```
dvsarnovskiyj@dvsarnovskiyj:~$ gedit prog1.sh
dvsarnovskiyj@dvsarnovskiyj:~$ ls
conf.txt  work  Загрузки  Общедоступные  Шаблоны
file.txt  Видео  Изображения  'Рабочий стол'
prog1.sh  Документы  Музыка  'Создание каталога'
dvsarnovskiyj@dvsarnovskiyj:~$ chmod +x prog1.sh
dvsarnovskiyj@dvsarnovskiyj:~$ ls
conf.txt  work  Загрузки  Общедоступные  Шаблоны
file.txt  Видео  Изображения  'Рабочий стол'
prog1.sh  Документы  Музыка  'Создание каталога'
dvsarnovskiyj@dvsarnovskiyj:~$ bash prog1.sh
prog1.sh
dvsarnovskiyj@dvsarnovskiyj:~$ ls
backup  prog1.sh  Документы  Музыка  'Создание каталога'
conf.txt  work  Загрузки  Общедоступные  Шаблоны
file.txt  Видео  Изображения  'Рабочий стол'
dvsarnovskiyj@dvsarnovskiyj:~$ ls backup
backup.tar
dvsarnovskiyj@dvsarnovskiyj:~$
```

Рис. 2. Результат работы скрипта №1

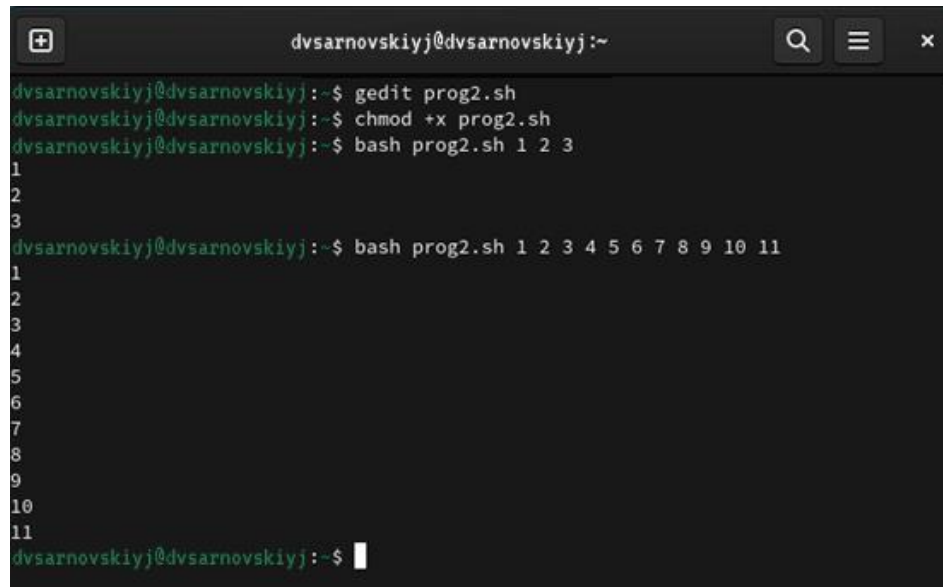
Напишем пример командного файла, обрабатывающего любое произвольное число аргументов командной строки:



```
1 #!/bin/bash
2
3 for arg in $*
4 do echo $arg
5 done
```

Рис. 3. Вывод аргументов командной строки

Делаем файл исполняемым и выводим результат. В данном случае будет выводиться последовательность аргументов командной строки.



```
dvsarnovskiyj@dvsarnovskiyj:~$ gedit prog2.sh
dvsarnovskiyj@dvsarnovskiyj:~$ chmod +x prog2.sh
dvsarnovskiyj@dvsarnovskiyj:~$ bash prog2.sh 1 2 3
1
2
3
dvsarnovskiyj@dvsarnovskiyj:~$ bash prog2.sh 1 2 3 4 5 6 7 8 9 10 11
1
2
3
4
5
6
7
8
9
10
11
dvsarnovskiyj@dvsarnovskiyj:~$
```

Рис. 4. Результат работы скрипта №2

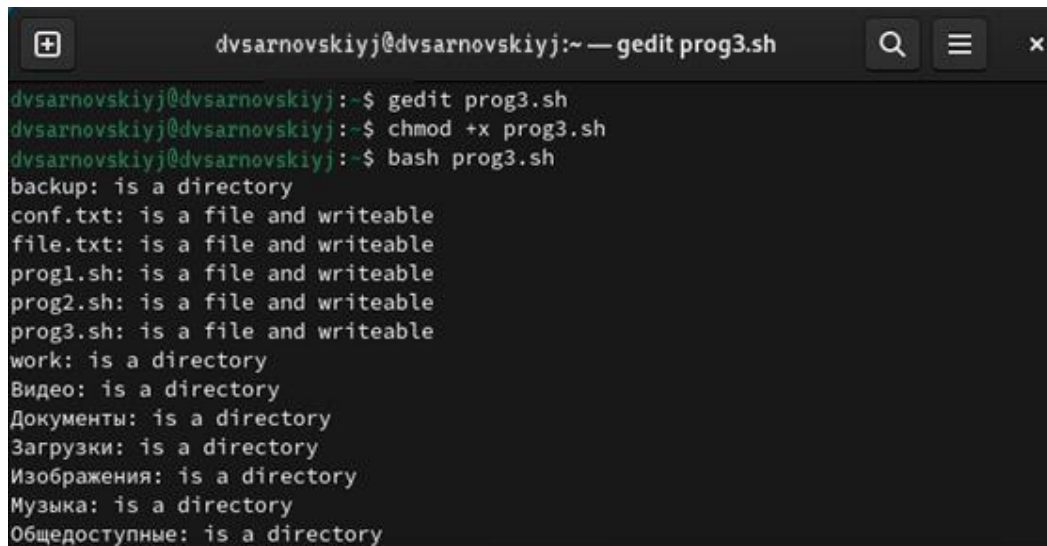
Напишем командный файл — аналог команды ls (без использования команды и команды dir), который выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.



```
1 #!/bin/bash
2
3 for A in *
4 do if test -d $A
5 then echo "$A: is a directory"
6 else echo -n "$A: is a file and "
7 if test -w $A
8 then echo writeable
9 elif test -r $A
10 then echo readable
11 else echo neither readable or writeable
12 fi
13 fi
14 done
15
```

Рис. 5. Аналог команды ls

Делаем файл исполняемым и выводим результат.



```
dvsarnovskiyj@dvsarnovskiyj:~ — gedit prog3.sh
dvsarnovskiyj@dvsarnovskiyj:~$ gedit prog3.sh
dvsarnovskiyj@dvsarnovskiyj:~$ chmod +x prog3.sh
dvsarnovskiyj@dvsarnovskiyj:~$ bash prog3.sh
backup: is a directory
conf.txt: is a file and writeable
file.txt: is a file and writeable
prog1.sh: is a file and writeable
prog2.sh: is a file and writeable
prog3.sh: is a file and writeable
work: is a directory
Видео: is a directory
Документы: is a directory
Загрузки: is a directory
Изображения: is a directory
Музыка: is a directory
Общедоступные: is a directory
```

Рис. 6. Результат работы скрипта №3

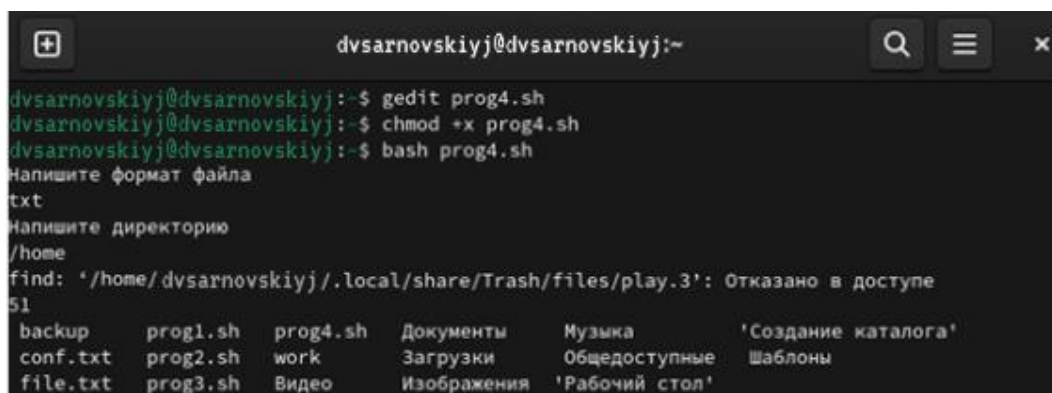
4. Напишем командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т. д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки.



```
*prog4.sh
1 #!/bin/bash
2
3 format=""
4 directory=""
5
6 echo "Напишите формат файла"
7 read format
8 echo "Напишите директорию"
9 read directory
10 find "${directory}" -name "*.${format}" -type f | wc -l
11 ls
```

Рис. 7. Подсчет количества файлов нужного формата

Делаем файл исполняемым и выводим результат.



```
dvsarnovskiyj@dvsarnovskiyj:~  
dvsarnovskiyj@dvsarnovskiyj:~$ gedit prog4.sh  
dvsarnovskiyj@dvsarnovskiyj:~$ chmod +x prog4.sh  
dvsarnovskiyj@dvsarnovskiyj:~$ bash prog4.sh  
Напишите формат файла  
txt  
Напишите директорию  
/home  
find: '/home/dvsarnovskiyj/.local/share/Trash/files/play.3': Отказано в доступе  
$1  
backup      prog1.sh    prog4.sh    Документы    Музыка      'Создание каталога'  
conf.txt    prog2.sh    work        Загрузки     Общедоступные  Шаблоны  
file.txt    prog3.sh    Видео       Изображения  'Рабочий стол'
```

Рис. 8. Результат работы скрипта №4

### Контрольные вопросы

1. Командная оболочка – это интерфейс между пользователем и операционной системой, который позволяет пользователю взаимодействовать с операционной системой путем ввода текстовых команд. Примеры командных оболочек включают Bash (Bourne Again Shell), Zsh (Z Shell), Fish (Friendly Interactive Shell) и другие. Они отличаются по своим возможностям, синтаксису, встроенным функциям и поддерживаемым расширениям.
2. POSIX (Portable Operating System Interface) – это семейство стандартов, разработанных для обеспечения совместимости между различными операционными системами Unix. Он определяет общие интерфейсы для программирования на языке C, командной строки и управления файлами.
3. В языке программирования bash переменные определяются путем присваивания значений их именам. Например:
  - Переменные: `variable_name=value`
  - Массивы: `array_name[index]=value`
4. Оператор `let` используется для выполнения арифметических выражений в bash. Оператор `read` используется для считывания значений из стандартного ввода и присваивания их переменным.
5. В языке программирования bash можно применять стандартные арифметические операции, такие как сложение, вычитание, умножение и деление.
6. Операция `(( ))` в bash используется для выполнения арифметических вычислений.
7. Некоторые стандартные имена переменных в bash:
  - HOME: домашний каталог текущего пользователя.
  - PWD: текущий рабочий каталог.
  - PATH: список каталогов, в которых операционная система ищет исполняемые файлы.

- USER: имя текущего пользователя.
- 8. Метасимволы – это символы, которые имеют специальное значение в контексте командной строки или шаблонов файлов. Некоторые примеры метасимволов включают \*, ?, [ ], { }, |, ; и &.
- 9. Для экранирования метасимволов в bash используется обратная косая черта \. Например, чтобы использовать символ \* как обычный символ, его можно экранировать так: \\*.
- 10. Для создания и запуска командных файлов в bash можно использовать текстовый редактор для создания файла с расширением .sh, затем присвоить ему права на выполнение с помощью команды `chmod +x filename.sh`, и, наконец, запустить файл с помощью команды `./filename.sh`.
- 11. Функции в языке программирования bash определяются с использованием ключевого слова `function` или просто с именем функции, после чего идет блок кода. Например:

```
function my_function {  
    # Код функции  
}
```

- 12. Для определения, является ли файл каталогом или обычным файлом, можно использовать команду `test`. Например:
  - Проверка на каталог: `test -d filename`
  - Проверка на обычный файл: `test -f filename`
- 13. Команды `set`, `typeset` и `unset` используются для работы с переменными в bash:
  - `set`: устанавливает значения и флаги для параметров командной строки.
  - `typeset`: используется для объявления переменных с определенными свойствами, такими как `readonly` или `integer`.
  - `unset`: удаляет значения переменных.
- 14. Параметры передаются в командные файлы в виде аргументов командной строки. Они доступны внутри скрипта через специальные переменные `$1`, `$2`, `$3` и так далее, где `$1` содержит первый аргумент, `$2` – второй и т.д.
- 15. Некоторые специальные переменные языка bash и их назначение:
  - `$0`: имя текущей выполняемой программы.
  - `$#`: количество аргументов, переданных скрипту.
  - `$?`: код возврата последней выполненной команды.
  - `$$`: PID (идентификатор процесса) текущего скрипта.
  - `$_`: PID последнего запущенного фонового процесса.

## Вывод

В данной лабораторной работе я изучила основы программирования в оболочке ОС UNIX/Linux и научилась писать небольшие командные файлы.

#### Список литературы

1. Руководство к лабораторной работе №9.