

# **Отчёт по лабораторной работе №12**

**Программирование в командном процессоре ОС UNIX. Командные  
файлы**

**Сергей Витальевич Павлюченков**

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>11</b>
<b>5</b>	<b>Контрольные вопросы</b>	<b>12</b>

## Список иллюстраций

3.1	Создание файла . . . . .	7
3.2	Код программы . . . . .	7
3.3	Меняю права доступа к файлу . . . . .	7
3.4	Запуск программы . . . . .	8
3.5	Создание файла . . . . .	8
3.6	Код программы . . . . .	8
3.7	Меняю права доступа к файлу . . . . .	8
3.8	Запуск программы . . . . .	8
3.9	Создание файла . . . . .	9
3.10	Код программы . . . . .	9
3.11	Меняю права доступа к файлу . . . . .	9
3.12	Запуск программы . . . . .	9
3.13	Создание файла . . . . .	10
3.14	Код программы . . . . .	10
3.15	Меняю права доступа к файлу . . . . .	10
3.16	Запуск программы . . . . .	10

## Список таблиц

# 1 Цель работы

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

## 2 Задание

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.

### 3 Выполнение лабораторной работы

Создаю директорию для лабораторной работы и файл для первого задания

```
[svpavliuchnikov@fedora work]$ mkdir os/lab12
[svpavliuchnikov@fedora work]$ cd os/lab12
[svpavliuchnikov@fedora lab12]$ touch backup_script
[svpavliuchnikov@fedora lab12]$
```

Рис. 3.1: Создание файла

Написал 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 архивируется архиватором tar.

```
[svpavliuchnikov@fedora lab12]$ cat backup_script
#!/bin/bash

filename=$(basename "$0")

tar -czvf "$HOME/backup/${filename}.tar.gz" $filename[svpavliuchnikov@fedora lab12]$
```

Рис. 3.2: Код программы

Делаю файл исполняемым

```
backup_script
[svpavliuchnikov@fedora lab12]$ chmod +x backup_script
[svpavliuchnikov@fedora lab12]$
```

Рис. 3.3: Меняю права доступа к файлу

Проверяю правильность выполнения программы

```
[svpavliuchenkov@fedora lab12]$ ./backup_script
backup_script
[svpavliuchenkov@fedora lab12]$ ls ~/backup/
backup_script.tar.gz
```

Рис. 3.4: Запуск программы

Создаю файл для второго задания

```
[svpavliuchenkov@fedora lab12]$ touch printing_script
[svpavliuchenkov@fedora lab12]$
```

Рис. 3.5: Создание файла

Написал командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Скрипт может последовательно распечатывать значения всех переданных аргументов.

```
[svpavliuchenkov@fedora lab12]$ cat printing_script
#!/bin/bash
for val in $@
do
    echo $val
done
[svpavliuchenkov@fedora lab12]$
```

Рис. 3.6: Код программы

Делаю файл исполняемым

```
svpavliuchenkov@fedora lab12]$ chmod +x printing_script
svpavliuchenkov@fedora lab12]$
```

Рис. 3.7: Меняю права доступа к файлу

Проверяю правильность выполнения программы

```
[svpavliuchenkov@fedora lab12]$ ./printing_script
[svpavliuchenkov@fedora lab12]$ ./printing_script 1 2 3 4 5
1
2
3
4
5
```

Рис. 3.8: Запуск программы

Создаю файл для третьего задания



```
[svpavliuchenkov@fedora lab12]$ touch ls_script
[svpavliuchenkov@fedora lab12]$ ls
backup_script  ls_script  printing_script
[svpavliuchenkov@fedora lab12]$
```

Рис. 3.9: Создание файла

Написал командный файл — аналог команды ls (без использования самой этой команды и команды dir).

```
[svpavliuchenkov@fedora lab12]$ cat ls_script
#!/bin/bash
catalog=$1
for file in $catalog/*
do if test -d $file
then echo "$file: is a directory"
elif test -f $file
then echo -n "$file: is a file and "
if test -w $file && test -r $file && test -e $file
then echo readable and writable and executable
elif test -w $file && test -r $file
then echo writable and readable
elif test -r $file && test -e $file
then echo readable and executable
elif test -r $file && test -w $file
then echo readable and writable
elif test -w $file
then echo writable
elif test -r $file
then echo readable
elif test -e $file
then echo executable
else echo neither readable nor writable nor executable
fi
fi
done[svpavliuchenkov@fedora lab12]$
```

Рис. 3.10: Код программы

Делаю файл исполняемым

```
[svpavliuchenkov@fedora lab12]$ chmod +x ls_script
[svpavliuchenkov@fedora lab12]$
```

Рис. 3.11: Меняю права доступа к файлу

Проверяю правильность выполнения программы

```
[svpavliuchenkov@fedora lab12]$ ./ls_script /home/svpavliuchenkov/work
/home/svpavliuchenkov/work/os: is a directory
/home/svpavliuchenkov/work/script: is a file and readable and writable and executable
/home/svpavliuchenkov/work/study: is a directory
[svpavliuchenkov@fedora lab12]$
```

Рис. 3.12: Запуск программы

Создаю файл для четвертого задания

```
[svpavliuchenkov@fedora lab12]$ touch extencion_counter
[svpavliuchenkov@fedora lab12]$
```

Рис. 3.13: Создание файла

Написал 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.

```
[svpavliuchenkov@fedora lab12]$ cat extencion_counter
#!/bin/bash
extention=$1
path=$2
let ans=0
for file in $path/*$extention
do
    let ans=ans+1
done
echo $ans[svpavliuchenkov@fedora lab12]$
```

Рис. 3.14: Код программы

Делаю файл исполняемым

```
[svpavliuchenkov@fedora lab12]$ chmod +x extencion_counter
[svpavliuchenkov@fedora lab12]$
```

Рис. 3.15: Меняю права доступа к файлу

Проверяю правильность выполнения программы

```
[svpavliuchenkov@fedora lab12]$ ./extencion_counter .txt ~
4
[svpavliuchenkov@fedora lab12]$ ls ~
abc  backup  conf.txt  file.txt  monthly  reports  text.txt  Видео  Загрузки  Музыка  'Рабочий стол'
abc1 conf2.txt feathers may      play      ski.places  work      Документы  Изображения  Общедоступные  Шаблоны
[svpavliuchenkov@fedora lab12]$
```

Рис. 3.16: Запуск программы

## 4 Выводы

Я узнал много нового о bash и программирование в командной строке. Научился создавать командные файлы

## 5 Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?

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; – C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; Главное отличие состоит в разном, но похожем синтаксисом.

2. Что такое POSIX?

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ

3. Как определяются переменные и массивы в языке программирования bash? Для создания массива используется команда set с флагом -A. А переменные создаются простым присваиванием - a="g"
4. Каково назначение операторов let и read? let указывает что указанная переменная это число, а read позволяет записывать данные в переменную с прямого ввода (input).

5. Какие арифметические операции можно применять в языке программирования bash?

Доступно сложение, деление, вычитание, умножение, взятие остатка от деления, побитовое сложение и вычитание тоже доступны.

6. Что означает операция (( ))?

Что внутри скобок происходит арифметические вычисления

7. Какие стандартные имена переменных Вам известны?

8. Что такое метасимволы?

Такие символы, как ' < > \* ? | " &, являются метасимволами и имеют для командного процессора специальный смысл

9. Как экранировать метасимволы? Поставить перед метасимволом обратный слеш - \

10. Как создавать и запускать командные файлы?

Для создание командного файла нужно указать внутри файла для какой оболочки написан текст внутри файла, а для запуска файл нужно сделать исполняемым.

11. Как определяются функции в языке программирования bash?

Группу команд можно объединить в функцию. Для этого существует ключевое слово function, после которого следует имя функции и список команд, заключённых в фигурные скобки. Удалить функцию можно с помощью команды unset с флагом -f.

12. Каким образом можно выяснить, является файл каталогом или обычным файлом? test -f/-d название файла/каталога

13. Каково назначение команд `set`, `typeset` и `unset`? Первая команда используется для создания массивов. Команда `typeset` имеет четыре опции для работы с функциями: `-f` — перечисляет определённые на текущий момент функции; `-ft` — при последующем вызове функции инициализирует её трассировку; `-fx` — экспортирует все перечисленные функции в любые дочерние программы оболочек; `-fu` — обозначает указанные функции как автоматически загружаемые. Автоматически загружаемые функции хранятся в командных файлах, а при их вызове оболочка просматривает переменную `FPATH`, отыскивая файл с одноимёнными именами функций, загружает его и вызывает эти функции. Изъять переменную из программы можно с помощью команды `unset`.

14. Как передаются параметры в командные файлы?

Они записываются в память и их можно достать оттуда с помощью `$номер параметра`, или достать все сразу `$*`

15. Назовите специальные переменные языка `bash` и их назначение. `$*` — отображается вся командная строка или параметры оболочки; `$!` — номер процесса, в рамках которого выполняется последняя вызванная на выполнение в командном режиме команда; `$$` — уникальный идентификатор процесса, в рамках которого выполняется командный процессор;