

# **Отчёт по лабораторной работе №12**

**Операционные системы**

Бекауов Артур Тимурович

# 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>12</b>
<b>5</b>	<b>Ответы на онтрольные вопросы</b>	<b>13</b>

## Список иллюстраций

3.1	Программа №1 . . . . .	7
3.2	Программа №2 - создание файла . . . . .	7
3.3	Программа №2 - выполнение . . . . .	8
3.4	Программа №3 - создание файла . . . . .	8
3.5	Программа №3 - выполнение . . . . .	10
3.6	Программа №4 - создание файла . . . . .	10
3.7	Программа №4 - выполнение . . . . .	11

## Список таблиц

# 1 Цель работы

Цель данной лабораторной 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 файл prog1.sh для новой программы меняю права доступа, разрешая его выполнение, таким образом файл становится исполняемым. Открываю файл в редакторе nano и записываю следующий код программы:

```
#!/bin/bash  
tar -cvf ~/backup/prog1.tar prog1.sh
```

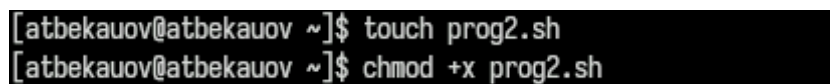
Сохраняю файл и закрываю редактор nano, далее запускаю исполняемый файл с помощью команды bash. Затем проверяю, что файл создал резервную копию самого себя (в виде архива) и поместил её в директорию backup.(рис. 3.1).



```
[atbekauov@atbekauov os-intro]$ cd ~  
[atbekauov@atbekauov ~]$  
[atbekauov@atbekauov ~]$ touch prog1.sh  
[atbekauov@atbekauov ~]$ chmod +x prog1.sh  
bash: chmod: команда не найдена  
[atbekauov@atbekauov ~]$ chmod +x prog1.sh  
[atbekauov@atbekauov ~]$ ls  
australia  feathers  hello.cpp  'lab07-3.sh'  layout.txt  monthly  prog1.sh  text.txt  Документы  Музыка  Шаблоны  
conf.txt  fun      'lab07-1.sh'  'lab07.sh'  LICENSE    my_os    reports   work      Загрузки  Общедоступные  
Downloads git-extended 'lab07-2.sh'  lab07.sh    may      play     ski.places  Видео     Изображения  'Рабочий стол'  
[atbekauov@atbekauov ~]$ mkdir backup  
[atbekauov@atbekauov ~]$ nano prog1.sh  
[atbekauov@atbekauov ~]$ bash prog1.sh  
prog1.sh  
[atbekauov@atbekauov ~]$ ls backup/  
prog1.tar  
[atbekauov@atbekauov ~]$
```

Рис. 3.1: Программа №1

Создаю файл prog2.sh, меняю права доступа, разрешая его выполнение. Открываю файл в nano (рис. 3.2).



```
[atbekauov@atbekauov ~]$ touch prog2.sh  
[atbekauov@atbekauov ~]$ chmod +x prog2.sh
```

Рис. 3.2: Программа №2 - создание файла

Затем ввожу в файл текст программы:

```
#!/bin/bash
for A in $*
do echo $A
done
```

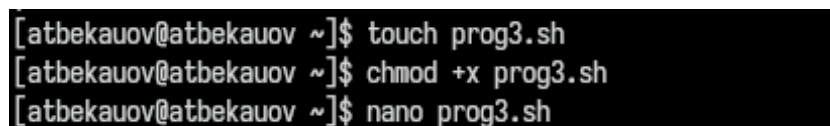
Сохраняю файл, выхожу из nano и запускаю файл через bash, введя в качестве аргумента разные символы. Как видим символы были продублированы скриптом. (рис. 3.3).



```
[atbekauov@atbekauov ~]$ bash prog2.sh 1 2 3 4 5 6 7 8 9 q w e r t y
1
2
3
4
5
6
7
8
9
q
w
e
r
t
y
[atbekauov@atbekauov ~]$
```

Рис. 3.3: Программа №2 - выполнение

Создаю файл prog3.sh, меняю права доступа, разрешая его выполнение. Открываю файл в nano (рис. 3.4).



```
[atbekauov@atbekauov ~]$ touch prog3.sh
[atbekauov@atbekauov ~]$ chmod +x prog3.sh
[atbekauov@atbekauov ~]$ nano prog3.sh
```

Рис. 3.4: Программа №3 - создание файла

Затем ввожу в файл текст программы:

```
#!/bin/bash
echo "Type in dir : "
read directory
echo " "
cd "${directory}"
```



```

for A in *
do
    if test -d "$A"
    then
        echo -n "$A is directory"
    else
        echo -n "$A is file "
        if test -r "$A"
        then
            echo -n "and readable "
        fi
        if test -w "$A"
        then
            echo -n " and writable "
        fi
        if test -e "$A"
        then
            echo -n "and executable "
        fi
    fi
done
echo
done

```

Сохраняю файл, выхожу из nano и запускаю файл через bash, скрипт просит ввести директорию - ввожу имя директории work. Скрипт выводит информацию о всех объектах этой директории. (рис. 3.5).

```
[atbekauov@atbekauov ~]$ bash prog3.sh
work
file1.txt is file and readable and writable and executable
file.txt is file and readable and writable and executable
os is directory
study is directory
[atbekauov@atbekauov ~]$ |
```

Рис. 3.5: Программа №3 - выполнение

Создаю файл prog4.sh, меняю права доступа, разрешая его выполнение. Открываю файл в nano (рис. 3.6).

```
[atbekauov@atbekauov ~]$ touch prog4.sh
[atbekauov@atbekauov ~]$ chmod +x prog4.sh
[atbekauov@atbekauov ~]$ nano prog4.sh
[atbekauov@atbekauov ~]$ |
```

Рис. 3.6: Программа №4 - создание файла

Затем ввожу в файл текст программы:

```
#!/bin/bash
format=""
directory=""
echo "Type in form:"
read format
echo "Type in dir"
read directory
find "${directory}" -name "*.${format}" -type f |wc -l
```

Сохраняю файл, выхожу из nano и запускаю файл через bash, скрипт просит ввести формат - ввожу txt, скрип просит ввести директорию - ввожу имя директории work. Кол-во txt файлов в директории work - 2.

```
[atbekauov@atbekauov ~]$ bash prog4.sh
Type in form:
txt
Type in dir
work
2
[atbekauov@atbekauov ~]$
```

Рис. 3.7: Программа №4 - выполнение

## 4 Выводы

В ходе данной лабораторной работы я изучил основы программирования в оболочке ОС UNIX/Linux. Научился писать небольшие командные файлы.

## 5 Ответы на онтрольные вопросы

1. Кратко охарактеризуйте редактор emacs.

Emacs — один из наиболее мощных и широко распространённых редакторов, используемых в мире UNIX. Написан на языке высокого уровня Lisp.

2. Какие особенности данного редактора могут сделать его сложным для освоения новичком?

Большое разнообразие сложных комбинаций клавиш, которые необходимы для редактирования файла и в принципе для работа с Emacs.

3. Своими словами опишите, что такое буфер и окно в терминологии emacs'а.

Буфер - это объект в виде текста. Окно - это прямоугольная область, в которой отображен буфер.

4. Можно ли открыть больше 10 буферов в одном окне?

Да, можно.

5. Какие буферы создаются по умолчанию при запуске emacs?

Emacs использует буферы с именами, начинающимися с пробела, для внутренних целей. Отчасти он обращается с буферами с такими именами особым образом — например, по умолчанию в них не записывается информация для отмены изменений.

6. Какие клавиши вы нажмёте, чтобы ввести следующую комбинацию C-с | и C-с C-|?

Ctrl + с, а потом | и Ctrl + с Ctrl + |

7. Как поделить текущее окно на две части?

С помощью команды Ctrl + x 3 (по вертикали) и Ctrl + x 2 (по горизонтали).

8. В каком файле хранятся настройки редактора emacs?

Настройки emacs хранятся в файле . emacs, который хранится в домашней директории пользователя. Кроме этого файла есть ещё папка . emacs.

9. Какую функцию выполняет клавиша и можно ли её переназначить?

Выполняет функцию стереть, думаю можно переназначить.

10. Какой редактор вам показался удобнее в работе vi или emacs? Поясните почему.

Для меня удобнее был редактор Emacs, так как у него есть командная оболочка. А vi открывается в терминале, и выглядит своеобразно.