

Лабораторная работа №10

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

Гузева Ирина Николаевна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Вывод	11

Список иллюстраций

3.1	Командный файл №1	7
3.2	Работа программы	8
3.3	Командный файл №2	8
3.4	Командный файл №3	9
3.5	Командный файл №4	10

Список таблиц

1 Цель работы

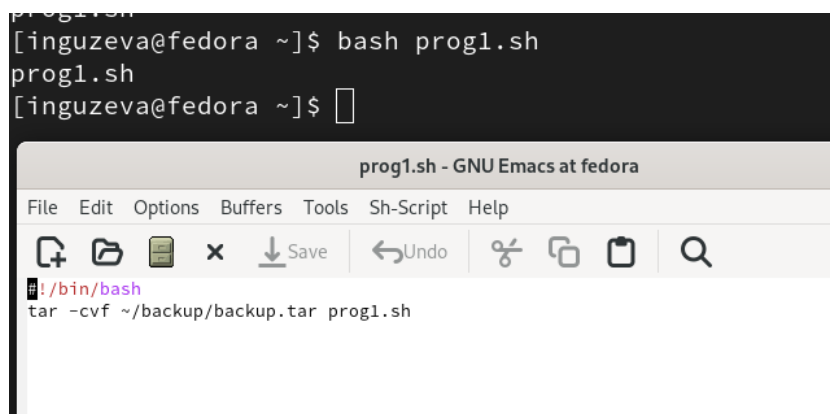
Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

2 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; • C-оболочка (или csh) — надстройка на оболочке Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; • оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; • BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C 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. Рассмотрим основные элементы программирования в оболочке bash. В других оболочках большинство команд будет совпадать с описанными ниже.

3 Выполнение лабораторной работы

- 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 домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar (рис. [3.1])



The image shows two overlapping windows. The top window is a terminal with a dark background. It displays the command `bash prog1.sh` being executed, followed by the output `prog1.sh` and a prompt `[inguzeva@fedora ~]$`. The bottom window is a GNU Emacs editor titled `prog1.sh - GNU Emacs at fedora`. It has a menu bar with `File`, `Edit`, `Options`, `Buffers`, `Tools`, `Sh-Script`, and `Help`. Below the menu is a toolbar with icons for file operations and editing. The main text area shows the command `#!/bin/bash` on the first line and `tar -cvf ~/backup/backup.tar prog1.sh` on the second line.

Рис. 3.1: Командный файл №1

- 2) Можно увидеть, что программа работала правильно (рис. [3.2])

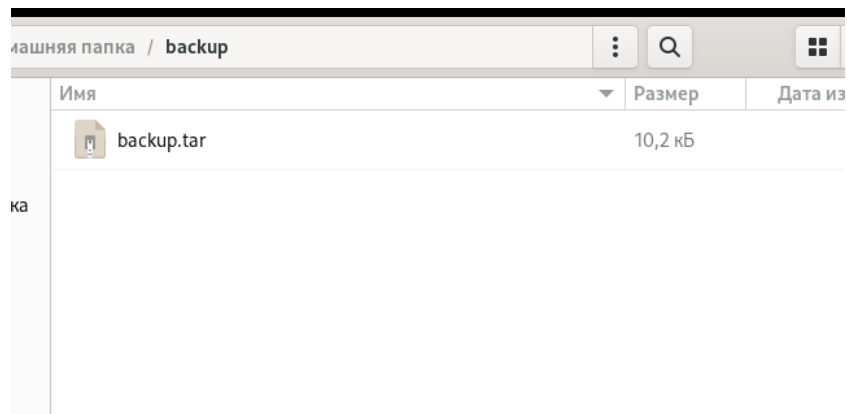


Рис. 3.2: Работа программы

- 3) Написала пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Скрипт может последовательно распечатывает значения всех переданных аргументов. (рис. [3.3])

```
[inguzeva@fedora ~]$ bash prog2.sh
Введите любые числа
12 34 56
[inguzeva@fedora ~]$
```

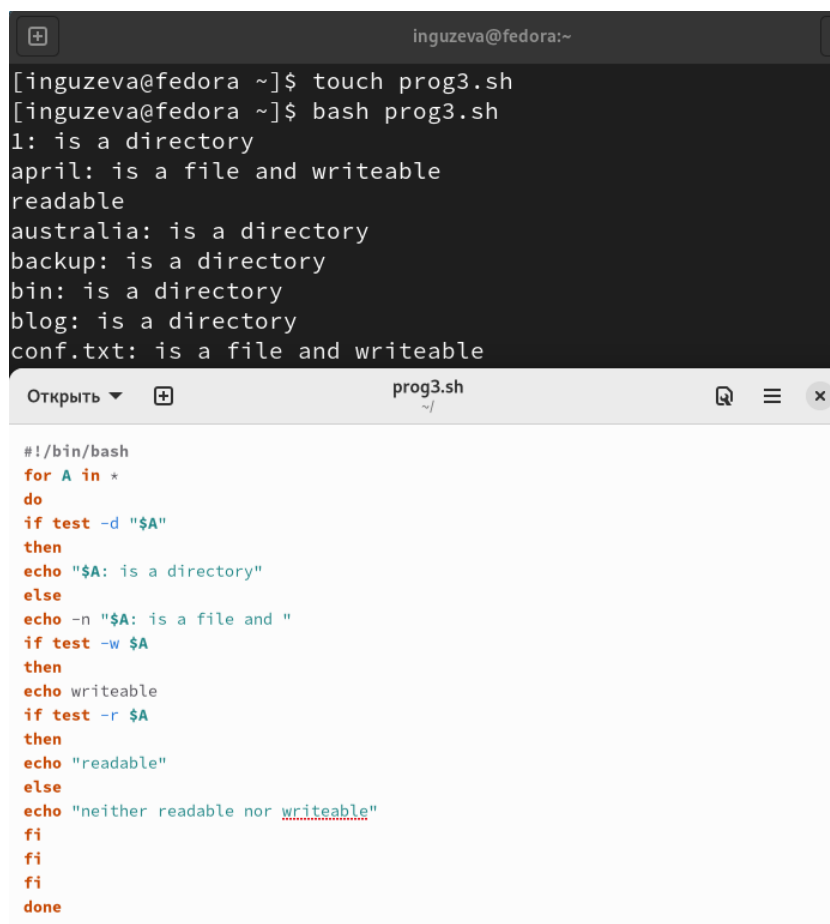
```
prog2.sh - GNU Emacs at fedora
File Edit Options Buffers Tools Sh-Script Help
[Icons: Copy, Paste, Save, Undo, etc.]

#!/bin/bash
echo 'Введите любые числа'
read n
for A in $*
do echo $A
done
```

Рис. 3.3: Командный файл №2

- 4) Написать командный файл — аналог команды `ls` (без использования самой этой команды и команды `dir`). Он выдает информацию о нужном каталоге

и выводит информацию о возможностях доступа к файлам этого каталога.
(рис. [3.4])



The image shows a terminal window and a code editor. The terminal window, titled 'inguzeva@fedora:~', shows the execution of a script 'prog3.sh'. The script iterates over files in the current directory and prints their permissions. The output is as follows:

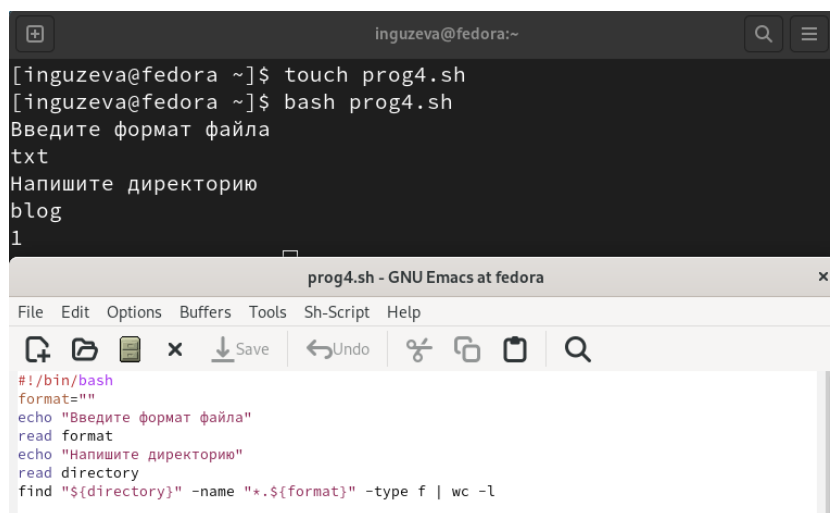
```
[inguzeva@fedora ~]$ touch prog3.sh
[inguzeva@fedora ~]$ bash prog3.sh
1: is a directory
april: is a file and writeable
readable
australia: is a directory
backup: is a directory
bin: is a directory
blog: is a directory
conf.txt: is a file and writeable
```

The code editor, titled 'prog3.sh', shows the script's source code:

```
#!/bin/bash
for A in *
do
if test -d "$A"
then
echo "$A: is a directory"
else
echo -n "$A: is a file and "
if test -w $A
then
echo writeable
if test -r $A
then
echo "readable"
else
echo "neither readable nor writeable"
fi
fi
fi
done
```

Рис. 3.4: Командный файл №3

- 5) Написала 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. (рис. [3.5])



The image shows a terminal window and an Emacs editor window. The terminal window, titled 'inguzeva@fedora:~', shows the following commands and output:

```
[inguzeva@fedora ~]$ touch prog4.sh
[inguzeva@fedora ~]$ bash prog4.sh
Введите формат файла
txt
Напишите директорию
blog
1
```

The Emacs editor window, titled 'prog4.sh - GNU Emacs at fedora', shows the contents of the script 'prog4.sh' with the following code:

```
#!/bin/bash
format=""
echo "Введите формат файла"
read format
echo "Напишите директорию"
read directory
find "${directory}" -name "*.${format}" -type f | wc -l
```

Рис. 3.5: Командный файл №4

4 Вывод

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