

Лабораторная работа No 10

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

Дмитрий Владимирович Орлюк

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	12
	Список литературы	13

Список иллюстраций

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 указанной директории. Путь к директории также передаётся в виде аргумента командной строки

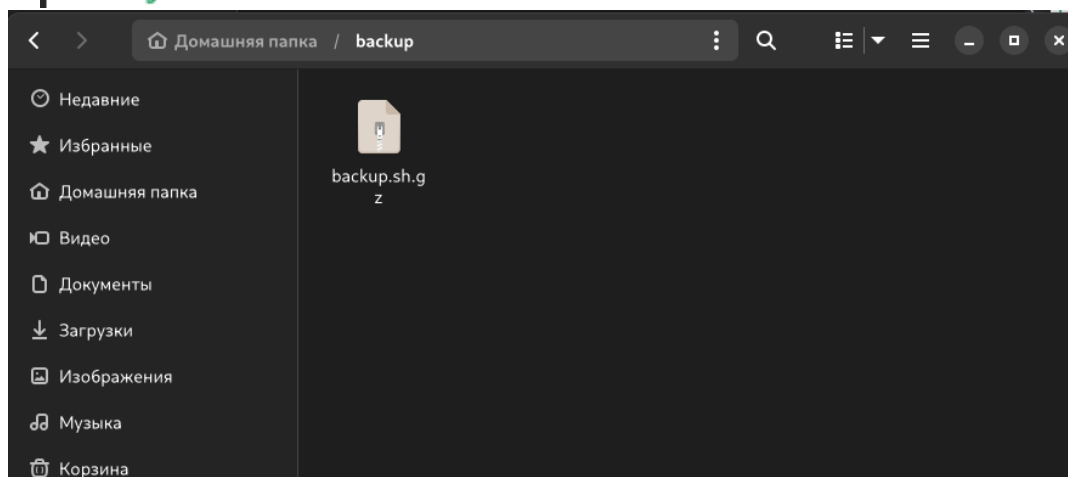
3 Теоретическое введение

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.

4 Выполнение лабораторной работы

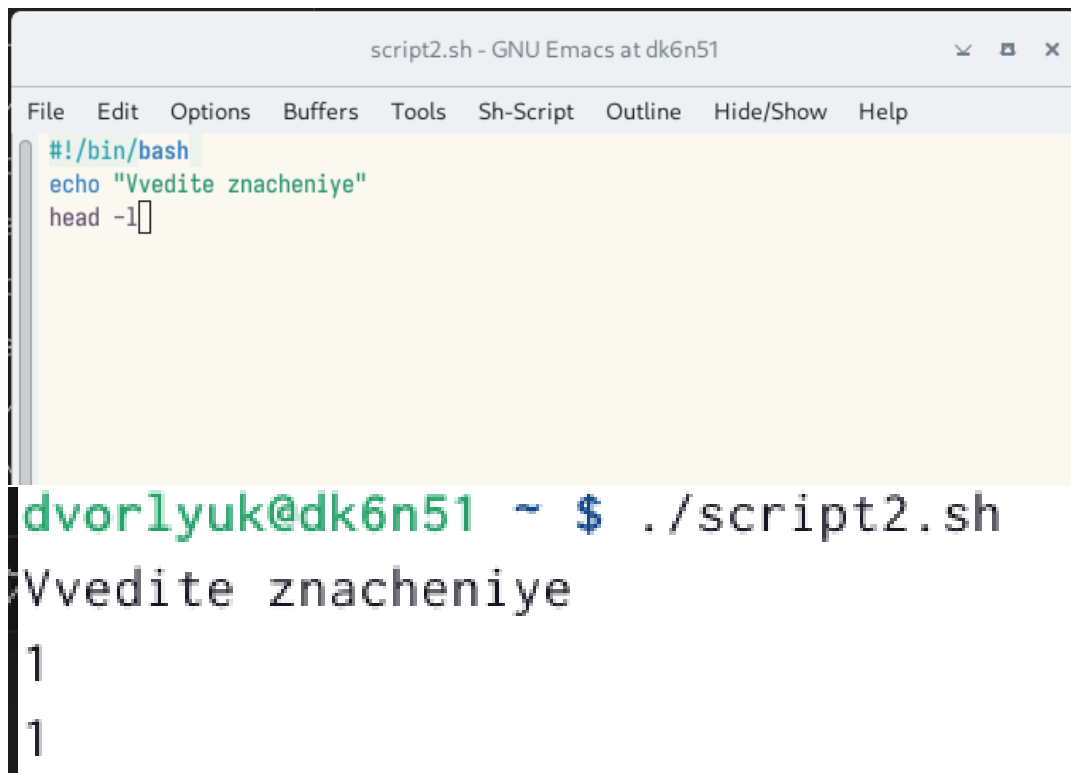
1. Создал файл и через emacs написал в нем нужный скрипт

```
dvorlyuk@dk6n51 ~ $ man tar
dvorlyuk@dk6n51 ~ $ touch script.sh
dvorlyuk@dk6n51 ~ $ chmod +x script.sh
dvorlyuk@dk6n51 ~ $ emacs
```



2. Создаем файл 2 вводим нужный скрипт и проверяем

```
dvorlyuk@dk6n51 ~ $ touch script2.sh
dvorlyuk@dk6n51 ~ $ chmod +x script2.sh
dvorlyuk@dk6n51 ~ $ emacs
```

The screenshot shows a terminal window titled "script2.sh - GNU Emacs at dk6n51". The window contains a menu bar with "File", "Edit", "Options", "Buffers", "Tools", "Sh-Script", "Outline", "Hide/Show", and "Help". The script content is as follows:

```
#!/bin/bash
echo "Vvedite znachenije"
head -1
```

Below the window, the terminal shows the command `./script2.sh` being executed. The output is:

```
Vvedite znachenije
1
1
```

3. Создаем файл 3 вводим нужный скрипт и проверяем

```
dvorlyuk@dk6n51 ~ $ touch script3.sh
dvorlyuk@dk6n51 ~ $ chmod +x script3.sh
dvorlyuk@dk6n51 ~ $ emacs
```

```
script3.sh - GNU Emacs at dk6n51
File Edit Options Buffers Tools Sh-Script Outline Hide/Show
#!/bin/bash
for A in *
do if test -d $A
  then echo $A: is a directory
  else ech -n $A: is a file and
    if test -w $A
    then ech writeable
    elif teat -r $A
    then echo readable
    else echo neither readable nor writeable
    fi
  fi
done

dvorlyuk@dk6n51 ~ $ touch script4.sh
dvorlyuk@dk6n51 ~ $ chmod +x script4.sh
dvorlyuk@dk6n51 ~ $ emacs
```

4. Создаем файл 3 вводим нужный скрипт и проверяем

```
#!/bin/bash
direct=''
form=''
echo 'write format'
read form
echo 'write directory'
read direct
find "$direct" -name ".*$form" -type f | wc -l
ls
```

```
dvorlyuk@dk6n51 ~ $ ./script4.sh
write format
34
write directory
21
find: '21': Нет такого файла или каталога
0
```

5 Выводы

Изучил основы программирования в оболочке ОС UNIX/Linux, научился писать небольшие командные файлы

Список литературы