

Отчёт по лабораторной работе №10

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

Надежда Александровна Рогожина

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	14
	Список литературы	15

Список иллюстраций

4.1	Создание файла	9
4.2	Директория backup	9
4.3	Скрипт №1	9
4.4	Команды	10
4.5	Скрипт №2	10
4.6	Скрипт №2	11
4.7	Скрипт №3	11
4.8	Скрипт №3	12
4.9	Скрипт №4	12
4.10	Скрипт №4	13

Список таблиц

3.1	Описание некоторых арифметических операторов оболочки bash	8
-----	------------------------------------------------------------	---

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 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 оболочкой 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).

Оболочка bash поддерживает встроенные арифметические функции. Команда `let` является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Простейшее выражение — это единичный терм (term), обычно целочисленный.

Целые числа можно записывать как последовательность цифр или в любом базовом формате типа `radix#number`, где radix (основание системы счисления)

— любое число не более 26. Для большинства команд используются следующие основания систем исчисления: 2 (двоичная), 8 (восьмеричная) и 16 (шестнадцатеричная). Простейшими математическими выражениями являются сложение (+), вычитание (-), умножение (*), целочисленное деление (/) и целочисленный остаток от деления (%).

Например, в табл. 3.1 приведено краткое описание некоторых арифметических операторов оболочки `bash`.

Таблица 3.1: Описание некоторых арифметических операторов оболочки `bash`

Оператор	Синтаксис	Результат
!	!exp	Если exp равно 0, то возвращает 1; иначе 0
%	exp1%exp2	Возвращает остаток от деления exp1 на exp2
&	exp1&exp2	Возвращает побитовое AND выражений exp1 и exp2
*	exp1 * exp2	Умножает exp1 на exp2
	exp1 exp2	Побитовое OR выражений exp1 и exp2
~	~exp	Побитовое дополнение до exp
^	exp1 ^ exp2	Исключающее OR выражений exp1 и exp2
>>	exp >> exp2	Сдвигает exp1 вправо на exp2 бит

4 Выполнение лабораторной работы

1. Для начала создадим файл script1 с помощью emacs (рис. 4.1):

```
[narogozhina@narogozhina ~]$ emacs script1
[narogozhina@narogozhina ~]$ ls
script1  work  Документы  Изображения  Общедоступные  Шаблоны
script1~ Видео  Загрузки  Музыка  'Рабочий стол'
```

Рис. 4.1: Создание файла

2. Создадим нужную нам директорию (рис. 4.2):

```
[narogozhina@narogozhina ~]$ mkdir backup
[narogozhina@narogozhina ~]$ ls
backup  script1~  Видео  Загрузки  Музыка  'Рабочий стол'
script1  work  Документы  Изображения  Общедоступные  Шаблоны
```

Рис. 4.2: Директория backup

3. Скрипт, который при запуске будет делать резервную копию самого себя в другую директорию backup в вашем домашнем каталоге (рис. 4.3):

```
tar -cf script1.tar script1
mv script1.tar /home/narogozhina/backup/script1.tar
```

Рис. 4.3: Скрипт №1

4. Чтобы каждый раз не прописывать `"bash script1"` сразу присвоим ему режим выполнения (+x) и также выполним скрипт и проверим (рис. 4.4):

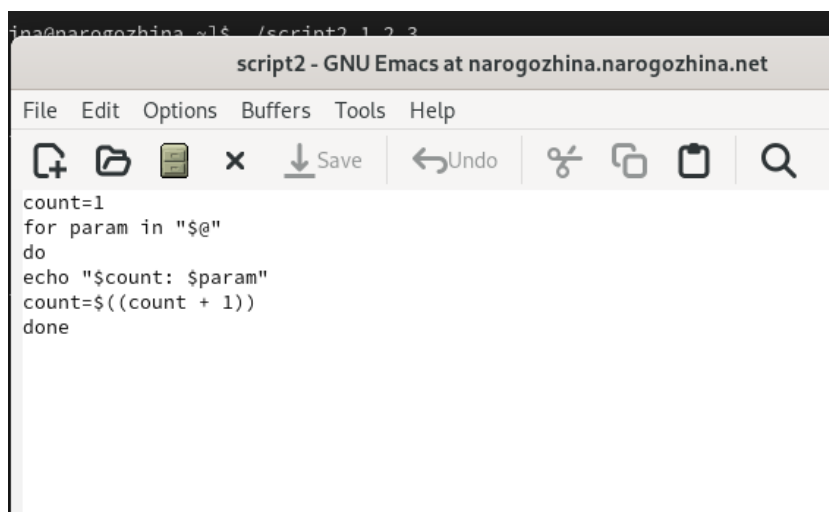
```

[narogozhina@narogozhina ~]$ chmod +x script1
[narogozhina@narogozhina ~]$ ./script1
[narogozhina@narogozhina ~]$ cd backup
[narogozhina@narogozhina backup]$ ls
script1.tar
[narogozhina@narogozhina backup]$ tar -xf script1.tar
[narogozhina@narogozhina backup]$ ls
script1 script1.tar

```

Рис. 4.4: Команды

5. Командный файл, обрабатывающий любое произвольное число аргументов командной строки, в том числе превышающее десять, последовательно распечатывающий значения всех переданных аргументов (рис. 4.5):



```

ina@narogozhina ~$ ./script2 1 2 3
script2 - GNU Emacs at narogozhina.narogozhina.net
File Edit Options Buffers Tools Help
count=1
for param in "$@"
do
echo "$count: $param"
count=$((count + 1))
done

```

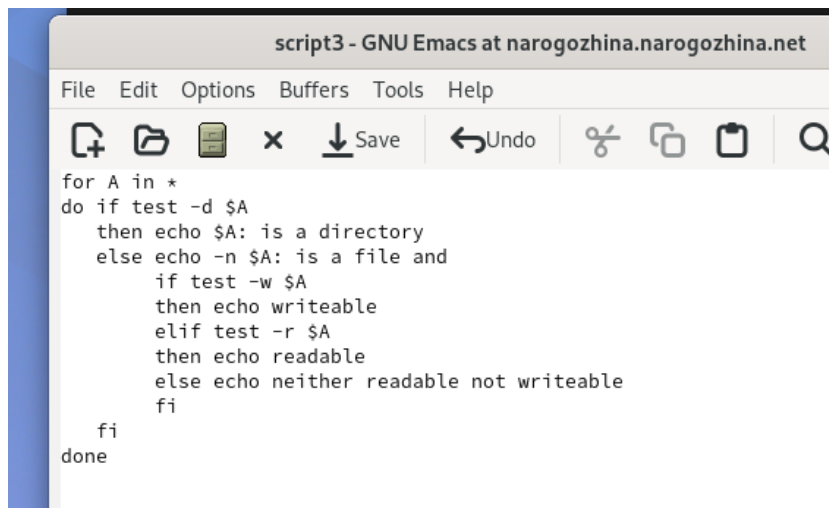
Рис. 4.5: Скрипт №2

Также присвоим режим выполнения и проверим выполнение самого скрипта (рис. 4.6):

```
[narogozhina@narogozhina ~]$ ./script2 1 2 3
1: 1
2: 2
3: 3
[narogozhina@narogozhina ~]$ ./script2 10 20 30 40 50
1: 10
2: 20
3: 30
4: 40
5: 50
[narogozhina@narogozhina ~]$ ./script2 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1: 1
2: 2
3: 3
4: 4
5: 5
6: 6
7: 7
8: 8
9: 9
10: 10
11: 11
12: 12
13: 13
14: 14
15: 15
[narogozhina@narogozhina ~]$
```

Рис. 4.6: Скрипт №2

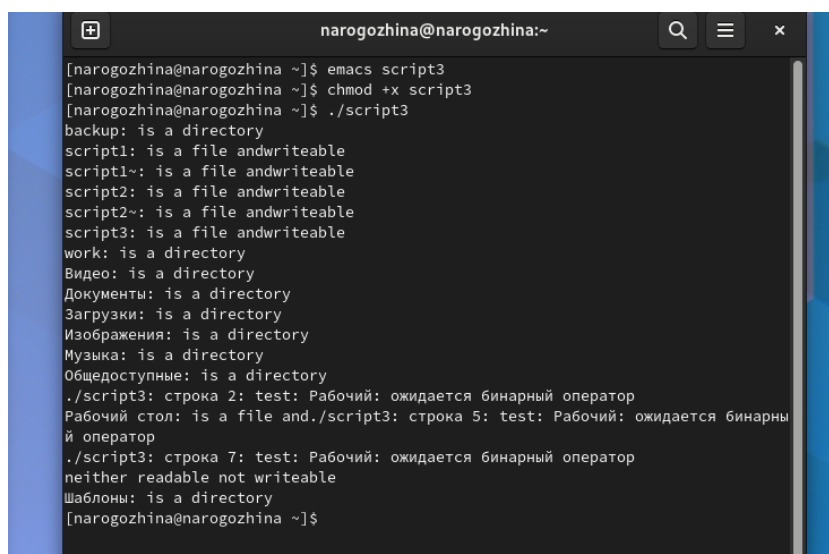
6. Командный файл — аналог команды ls (без использования самой этой команды и команды dir). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога (рис. 4.7):



```
script3 - GNU Emacs at narogozhina.narogozhina.net
File Edit Options Buffers Tools Help
[Icons: New, Open, Save, Close, Save As, Undo, Cut, Copy, Paste, Find]
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
elif test -r $A
then echo readable
else echo neither readable not writeable
fi
fi
done
```

Рис. 4.7: Скрипт №3

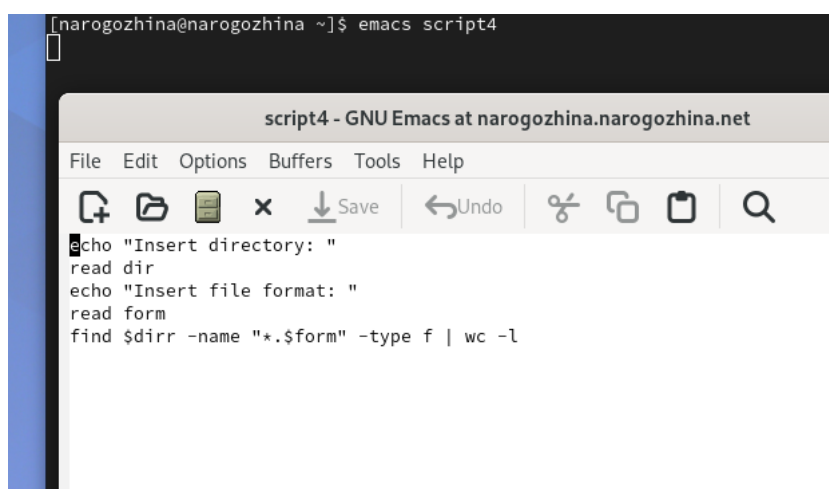
Пример выполнения (рис. 4.8):



```
[narogozhina@narogozhina ~]$ emacs script3
[narogozhina@narogozhina ~]$ chmod +x script3
[narogozhina@narogozhina ~]$ ./script3
backup: is a directory
script1: is a file andwriteable
script1~: is a file andwriteable
script2: is a file andwriteable
script2~: is a file andwriteable
script3: is a file andwriteable
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
./script3: строка 2: test: Рабочий: ожидается бинарный оператор
Рабочий стол: is a file and./script3: строка 5: test: Рабочий: ожидается бинарны
й оператор
./script3: строка 7: test: Рабочий: ожидается бинарный оператор
neither readable not writeable
Шаблоны: is a directory
[narogozhina@narogozhina ~]$
```

Рис. 4.8: Скрипт №3

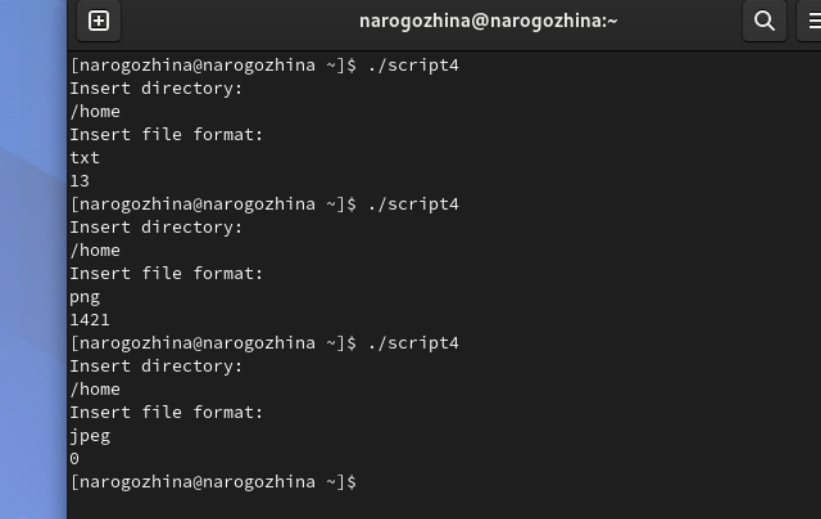
7. Командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки (рис. 4.9):



```
[narogozhina@narogozhina ~]$ emacs script4
script4 - GNU Emacs at narogozhina.narogozhina.net
File Edit Options Buffers Tools Help
[Icons: New, Open, Save, Close, Undo, Cut, Copy, Paste, Find]
echo "Insert directory: "
read dir
echo "Insert file format: "
read form
find $dirr -name ".*$form" -type f | wc -l
```

Рис. 4.9: Скрипт №4

Пример вывода (рис. 4.10):



```
narogozhina@narogozhina:~  
[narogozhina@narogozhina ~]$ ./script4  
Insert directory:  
/home  
Insert file format:  
txt  
13  
[narogozhina@narogozhina ~]$ ./script4  
Insert directory:  
/home  
Insert file format:  
png  
1421  
[narogozhina@narogozhina ~]$ ./script4  
Insert directory:  
/home  
Insert file format:  
jpeg  
0  
[narogozhina@narogozhina ~]$
```

Рис. 4.10: Скрипт №4

5 Выводы

В ходе лабораторной работы мы познакомились с командными файлами Linux, а также научились писать небольшие командные файлы сами.

Список литературы

1. Руководство по выполнению лабораторной работы №10
2. Справочник по архивации в linux