

Отчет по лабораторной работе №10

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

Максим Сергеевич Белов

Содержание

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

Список иллюстраций

4.1	Скрипт 1	8
4.2	Проверка скрипта 1	8
4.3	Скрипт 2	9
4.4	Проверка скрипта 2	9
4.5	Скрипт 3	9
4.6	Проверка скрипта 3	10
4.7	Скрипт 4	10
4.8	Проверка скрипта 4	11

List of Tables

3.1	Описание команд для работы с командными файлами	7
-----	-----------------------------------------------------------	---

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

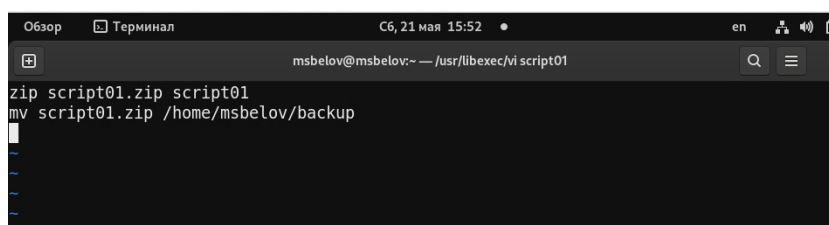
В табл. 3.1 приведено краткое описание команд для работы с командными файлами.

Таблица 3.1: Описание команд для работы с командными файлами

Команда	Описание
<code>getopts</code>	Осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных.
<code>test</code>	Единственная функция этой команды заключается в выработке кода завершения

4 Выполнение лабораторной работы

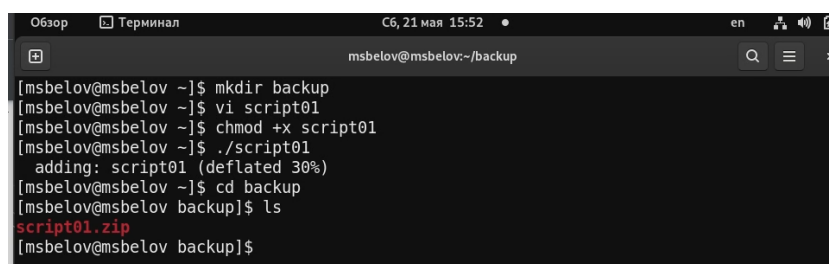
1. Напишем скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. (Рис. 4.1)



```
msbelov@msbelov:~ — /usr/libexec/vi script01
zip script01.zip script01
mv script01.zip /home/msbelov/backup
```

Рис. 4.1: Скрипт 1

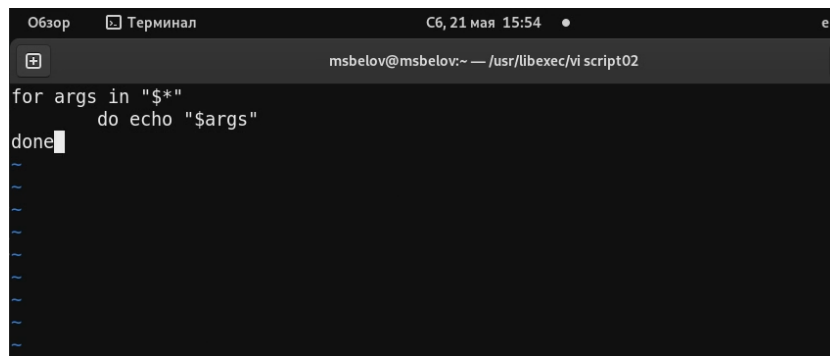
Сделаем файл script01 исполняемым и проверим его работу (Рис. 4.2)



```
[msbelov@msbelov ~]$ mkdir backup
[msbelov@msbelov ~]$ vi script01
[msbelov@msbelov ~]$ chmod +x script01
[msbelov@msbelov ~]$ ./script01
  adding: script01 (deflated 30%)
[msbelov@msbelov ~]$ cd backup
[msbelov@msbelov backup]$ ls
script01.zip
[msbelov@msbelov backup]$
```

Рис. 4.2: Проверка скрипта 1

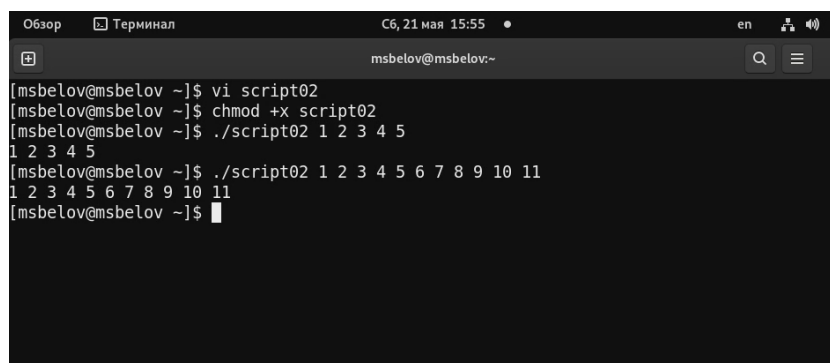
2. Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов. (Рис. 4.3)



```
Обзор Терминал C6, 21 мая 15:54 en
msbelov@msbelov:~ — /usr/libexec/vi script02
for args in "$*"
do echo "$args"
done
~
~
~
~
~
~
~
```

Рис. 4.3: Скрипт 2

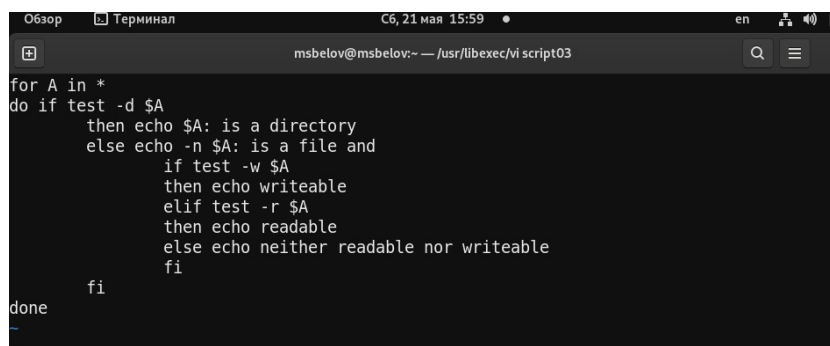
Сделаем файл исполняемым и проверим его работу (Рис. 4.4)



```
Обзор Терминал C6, 21 мая 15:55 en
msbelov@msbelov:~
[msbelov@msbelov ~]$ vi script02
[msbelov@msbelov ~]$ chmod +x script02
[msbelov@msbelov ~]$ ./script02 1 2 3 4 5
1 2 3 4 5
[msbelov@msbelov ~]$ ./script02 1 2 3 4 5 6 7 8 9 10 11
1 2 3 4 5 6 7 8 9 10 11
[msbelov@msbelov ~]$
```

Рис. 4.4: Проверка скрипта 2

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. (Рис. 4.5)

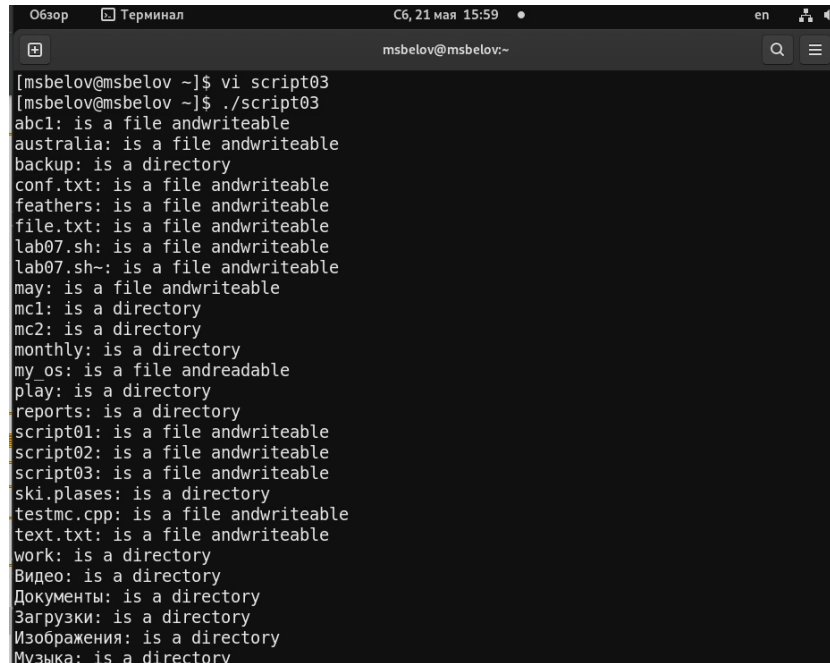


```
Обзор Терминал C6, 21 мая 15:59 en
msbelov@msbelov:~ — /usr/libexec/vi script03
for A in *
do if test -d $A
then echo $A: is a directory
else echo -n $A: is a file and
if test -w $A
then echo writeable
elif test -r $A
then echo readable
else echo neither readable nor writeable
fi
fi
done
~
~
```

Рис. 4.5: Скрипт 3

Сделаем script03 исполняемым и проверим его работу

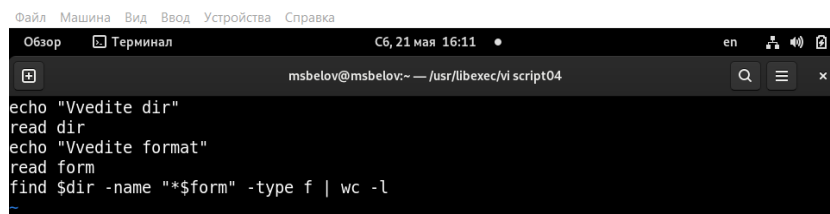
(Рис. 4.6)



```
[msbelov@msbelov ~]$ vi script03
[msbelov@msbelov ~]$ ./script03
abc1: is a file andwriteable
australia: is a file andwriteable
backup: is a directory
conf.txt: is a file andwriteable
feathers: is a file andwriteable
file.txt: is a file andwriteable
lab07.sh: is a file andwriteable
lab07.sh~: is a file andwriteable
may: is a file andwriteable
mc1: is a directory
mc2: is a directory
monthly: is a directory
my_os: is a file andreadable
play: is a directory
reports: is a directory
script01: is a file andwriteable
script02: is a file andwriteable
script03: is a file andwriteable
ski.places: is a directory
testmc.cpp: is a file andwriteable
text.txt: 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
```

Рис. 4.6: Проверка скрипта 3

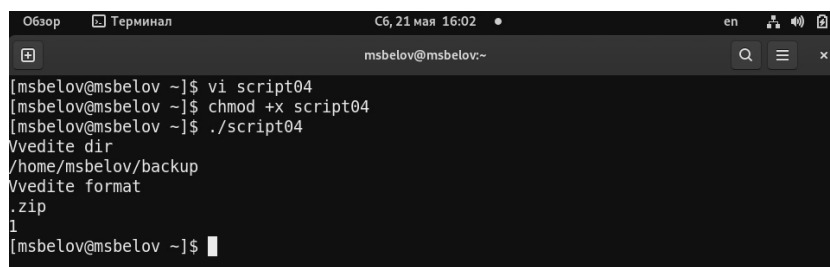
4. Написать командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки (Рис. 4.7)



```
Файл Машина Вид Ввод Устройства Справка
Обзор Терминал Сб, 21 мая 16:11
msbelov@msbelov:~ — /usr/libexec/vi script04
echo "Vvedite dir"
read dir
echo "Vvedite format"
read form
find $dir -name "$form" -type f | wc -l
```

Рис. 4.7: Скрипт 4

Сделаем script04 исполняемым и проверим его работу (Рис. 4.8)



```
Обзор Терминал C6, 21 мая 16:02 en msbelov@msbelov:~
[msbelov@msbelov ~]$ vi script04
[msbelov@msbelov ~]$ chmod +x script04
[msbelov@msbelov ~]$ ./script04
Vvedite dir
/home/msbelov/backup
Vvedite format
.zip
1
[msbelov@msbelov ~]$
```

Рис. 4.8: Проверка скрипта 4

5 Выводы

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

6 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; - 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).

2. Что такое POSIX?

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 Керна.

3. Как определяются переменные и массивы в языке программирования bash?

Командный процессор bash обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем. Пользователь имеет возможность присвоить переменной значение некоторой строки символов. Значение, присвоенное некоторой переменной, может быть впоследствии использовано. Для этого в соответствующем месте командной строки должно быть употреблено имя этой переменной, которому предшествует метасимвол `$`.

4. Каково назначение операторов `let` и `read`?

Команда `let` является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению.

Команда `read` позволяет считывать значения, введенные с клавиатуры.

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

Простейшими математическими выражениями являются сложение (+), вычитание (-), умножение (*), целочисленное деление (/) и целочисленный остаток от деления (%).

6. Что означает операция (())?

Для облегчения программирования можно записывать условия оболочки bash в двойные скобки — `(())`.

7. Какие стандартные имена переменных Вам известны?

- HOME — имя домашнего каталога пользователя. Если команда `cd` вводится без аргументов, то происходит переход в каталог, указанный в этой переменной.
- IFS — последовательность символов, являющихся разделителями в командной строке, например, пробел, табуляция и перевод строки (new line).
- MAIL — командный процессор каждый раз перед выводом на экран промптера проверяет содержимое файла, имя которого указано в этой переменной, и если содержимое этого файла изменилось с момента последнего ввода из него, то перед тем как вывести на терминал промптер, командный процессор выводит на терминал сообщение `You have mail` (у Вас есть почта).
- TERM — тип используемого терминала.
- LOGNAME — содержит регистрационное имя пользователя, которое устанавливается автоматически при входе в систему.

8. Что такое метасимволы?

Такие символы, как `' < > * ? | " &`, являются метасимволами и имеют для командного процессора специальный смысл. Снятие специального смысла с метасимвола называется экранированием метасимвола.

9. Как экранировать метасимволы?

Экранирование может быть осуществлено с помощью предшествующего метасимволу символа `\`, который, в свою очередь, является метасимволом. Для экранирования группы метасимволов нужно заключить её в одинарные кавычки. Строка, заключённая в двойные кавычки, экранирует все метасимволы, кроме `$, ', , "`. Например, `- echo *` выведет на экран символ `*`, `- echo ab'|'cd` выведет на экран строку `ab|*cd`.

10. Как создавать и запускать командные файлы?

Сначала нужно воспользоваться любым редактором и записать в файл какой-либо код, затем сделать этот файл исполняемым командой `chmod +x`. После чего запустить его командой `./`.

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.

12. Каким образом можно выяснить, является файл каталогом или обычным файлом?

Задать условие:

```
if <имя файла> -d
```

13. Каково назначение команд `set`, `typeset` и `unset`?

Значение всех переменных можно просмотреть с помощью команды `set`.

Удалить функцию можно с помощью команды `unset` с флагом `-f`.

Команда `typeset` имеет четыре опции для работы с функциями: `-f` — перечисляет определённые на текущий момент функции; `-ft` — при последующем вызове функции иницирует 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.

14. Как передаются параметры в командные файлы?

При вызове командного файла на выполнение параметры ему могут быть переданы точно таким же образом, как и выполняемой программе. С точки зрения командного файла эти параметры являются позиционными. Символ `$` является метасимволом командного процессора. Он используется, в частности, для ссылки на параметры, точнее, для получения их значений в командном файле. В командный файл можно передать до девяти параметров. При использовании

где-либо в командном файле комбинации символов $\$i$, где $0 < i < 10$, вместо неё будет осуществлена подстановка значения параметра с порядковым номером i , т.е. аргумента командного файла с порядковым номером i . Использование комбинации символов $\$0$ приводит к подстановке вместо неё имени данного командного файла.

15. Назовите специальные переменные языка `bash` и их назначение.

- $\$*$ — отображается вся командная строка или параметры оболочки;
- $\$?$ — код завершения последней выполненной команды;
- $\$\$$ — уникальный идентификатор процесса, в рамках которого выполняется командный процессор;
- $\$!$ — номер процесса, в рамках которого выполняется последняя вызванная на выполнение в командном режиме команда;
- $\$-$ — значение флагов командного процессора;
- $\${\#*}$ — возвращает целое число — количество слов, которые были результатом $\$*$;
- $\${\#name}$ — возвращает целое значение длины строки в переменной `name`;
- $\${name[n]}$ — обращение к n -му элементу массива;
- $\${name[*]}$ — перечисляет все элементы массива, разделённые пробелом;
- $\${name[@]}$ — то же самое, но позволяет учитывать символы пробелы в самих переменных;
- $\${name:-value}$ — если значение переменной `name` не определено, то оно будет заменено на указанное `value`;
- $\${name:value}$ — проверяется факт существования переменной;
- $\${name=value}$ — если `name` не определено, то ему присваивается значение `value`;
- $\${name?value}$ — останавливает выполнение, если имя переменной не определено, и выводит `value` как сообщение об ошибке;
- $\${name+value}$ — это выражение работает противоположно $\${name-value}$. Если переменная определена, то подставляется `value`;

- `${name#pattern}` — представляет значение переменной `name` с удалённым самым коротким левым образцом (`pattern`);
- `${#name[*]}` и `${#name[@]}` — эти выражения возвращают количество элементов в массиве `name`.