

Отчет по лабораторной работе №10

Дисциплина: операционные системы

Лобанова Полина Иннокентьевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Контрольные вопросы	12
4	Вывод	17

Список иллюстраций

2.1	Создание файлов.	6
2.2	Текст командного файла.	7
2.3	Результат.	7
2.4	Текст командного файла.	8
2.5	Результат.	8
2.6	Текст командного файла.	9
2.7	Результат.	10
2.8	Текст командного файла.	11
2.9	Результат.	11

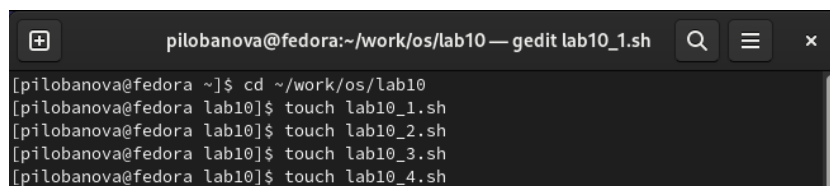
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1. Создадим необходимые файлы, чтобы в дальнейшем их заполнить.



```
pilobanova@fedora:~/work/os/lab10 — gedit lab10_1.sh
[pilobanova@fedora ~]$ cd ~/work/os/lab10
[pilobanova@fedora lab10]$ touch lab10_1.sh
[pilobanova@fedora lab10]$ touch lab10_2.sh
[pilobanova@fedora lab10]$ touch lab10_3.sh
[pilobanova@fedora lab10]$ touch lab10_4.sh
```

Рис. 2.1: Создание файлов.

2. Напишем скрипт, который при запуске будет делать резервную копию самого себя в другую директорию backup в нашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar.

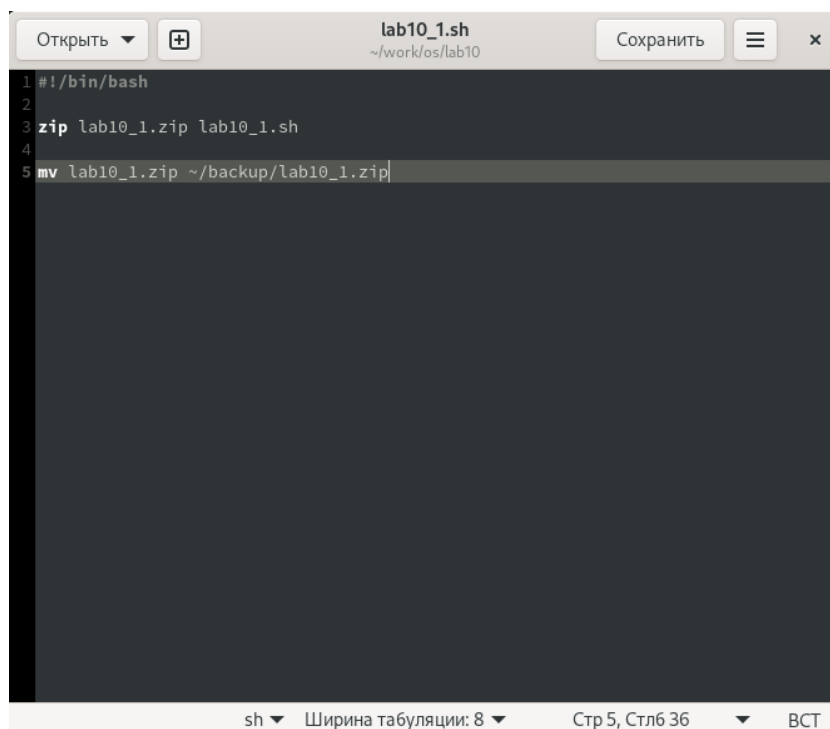


Рис. 2.2: Текст командного файла.

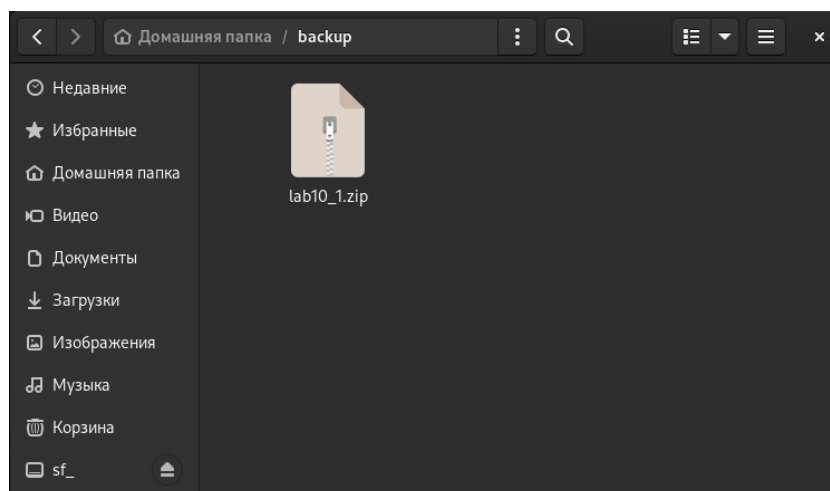
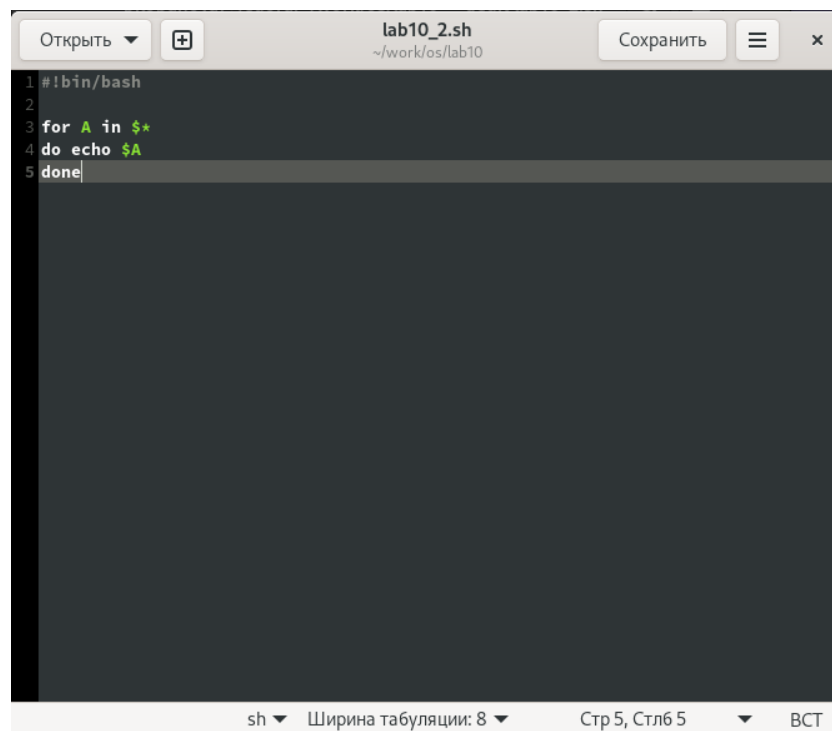


Рис. 2.3: Результат.

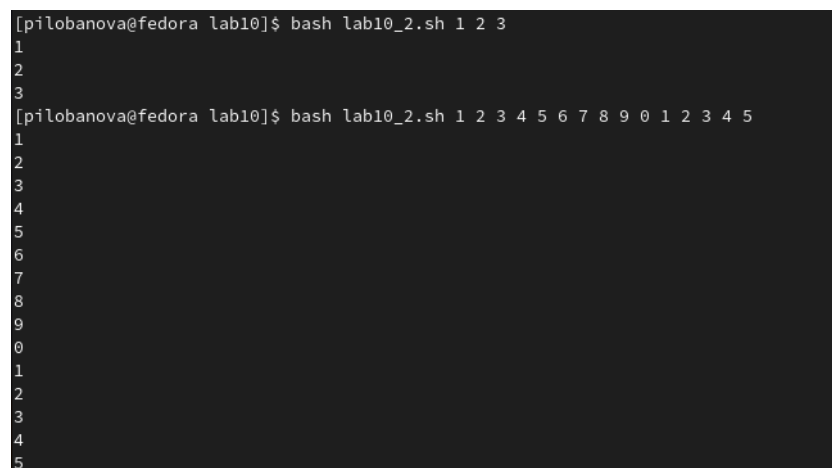
3. Напишем пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт будет последовательно распечатывать значения всех

переданных аргументов.



```
1 #!bin/bash
2
3 for A in $*
4 do echo $A
5 done
```

Рис. 2.4: Текст командного файла.

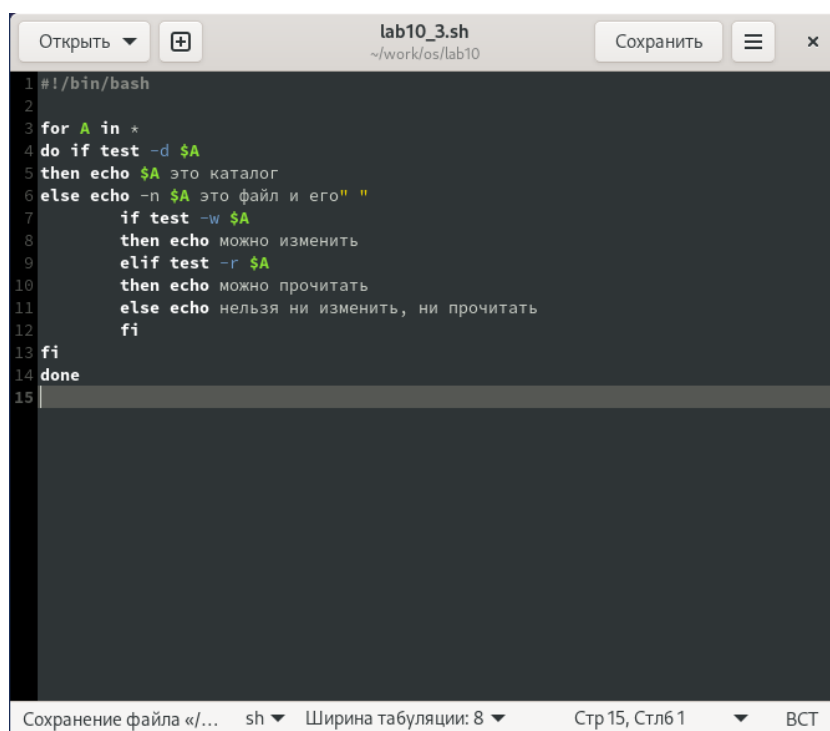


```
[pilobanova@fedora lab10]$ bash lab10_2.sh 1 2 3
1
2
3
[pilobanova@fedora lab10]$ bash lab10_2.sh 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
```

Рис. 2.5: Результат.

4. Напишем командный файл — аналог команды `ls` (без использования самой этой команды и команды `dir`) так, чтобы он выдавал информацию о нужном

каталоге и выводил информацию о возможностях доступа к файлам этого каталога.



```
1#!/bin/bash
2
3for A in *
4do if test -d $A
5then echo $A это каталог
6else echo -n $A это файл и его" "
7     if test -w $A
8     then echo можно изменить
9     elif test -r $A
10    then echo можно прочитать
11    else echo нельзя ни изменить, ни прочитать
12    fi
13fi
14done
15
```

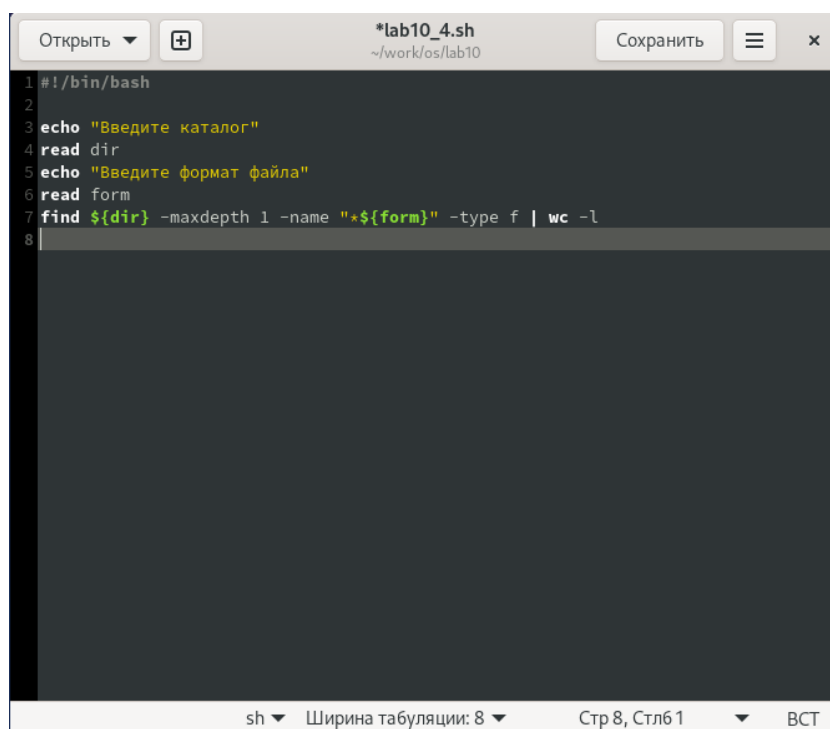
Сохранение файла «/... sh ▾ Ширина табуляции: 8 ▾ Стр 15, Стлб 1 ▾ ВСТ

Рис. 2.6: Текст командного файла.

```
[pilobanova@fedora ~]$ bash ~/work/os/lab10/lab10_3.sh
5 это каталог
abc1 это файл и его можно изменить
april это файл и его можно изменить
backup это каталог
bin это каталог
conf.txt это файл и его можно изменить
#demo# это файл и его можно изменить
feathers это файл и его можно изменить
file.txt это файл и его можно изменить
in_out.asm это файл и его можно изменить
l4 это каталог
l5 это каталог
lab07 это каталог
lab07.zip это файл и его можно изменить
#lab09.sh# это файл и его можно изменить
lab09.sh это файл и его можно изменить
lab09.sh~ это файл и его можно изменить
lab4 это каталог
lab4.tar это файл и его можно изменить
lab5 это каталог
lab7 это каталог
lab7~4 это файл и его можно изменить
lab7~4.asm это файл и его можно изменить
lab7~4.o это файл и его можно изменить
may это файл и его можно изменить
monthly это каталог
monthly.00 это каталог
play это каталог
ski.places это каталог
text.txt это файл и его можно изменить
work это каталог
Видео это каталог
Документы это каталог
Записки это каталог
```

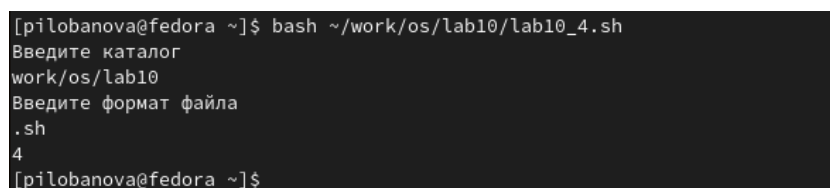
Рис. 2.7: *Результат.*

5. Напишем 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.



```
1 #!/bin/bash
2
3 echo "Введите каталог"
4 read dir
5 echo "Введите формат файла"
6 read form
7 find ${dir} -maxdepth 1 -name "${form}" -type f | wc -l
8
```

Рис. 2.8: Текст командного файла.



```
[pilobanova@fedora ~]$ bash ~/.work/os/lab10/lab10_4.sh
Введите каталог
work/os/lab10
Введите формат файла
.sh
4
[pilobanova@fedora ~]$
```

Рис. 2.9: Результат.

3 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

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

`mark=/usr/andy/bin` Данная команда присваивает значение строки символов `/usr/andy/bin` переменной `mark` типа строка символов. Для создания массива используется команда `set` с флагом `-A`. За флагом следует имя переменной, а затем список значений, разделённых пробелами. Например,

```
set -A states Delaware Michigan "New Jersey"
```

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

Команда `let` является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Команда `read` позволяет читать значения переменных со стандартного ввода

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?

Переменные `PS1` и `PS2` предназначены для отображения промптера командного процессора. `PS1` — это промптер командного процессора, по умолчанию его значение равно символу `$` или `#`. Если какая-то интерактивная программа, запущенная командным процессором, требует ввода, то используется промптер `PS2`. Он по умолчанию имеет значение символа `>`. Другие стандартные переменные:

- `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

9. Как экранировать метасимволы?

Снятие специального смысла с метасимвола называется экранированием метасимвола. Экранирование может быть осуществлено с помощью предшествующего метасимволу символа, который, в свою очередь, является метасимволом.

10. Как создавать и запускать командные файлы?

Командный файл можно создать с помощью какого-либо редактора, затем сделать его исполняемым и запустить его из терминала, введя “./название файла”.

11. Как определяются функции в языке программирования bash?

С помощью ключевого слова function.

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

Вводим команду `ls -lrt` и если первым в правах доступа стоит `d` то это каталог. Иначе это файл

13. Каково назначение команд set, typeset и unset?

Для создания массива используется команда set с флагом -A. Если использовать typeset -i для объявления и присвоения переменной, то при последующем её применении она станет целой. Изъять переменную из программы можно с помощью команды unset.

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. • \$* — 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`.

4 Вывод

Я изучила основы программирования в оболочке ОС UNIX/Linux и научилась писать небольшие командные файлы.