

Лаборная работа №10

НММ-6д-02-22

Крухмалев Артём Владиславович

Содержание

1	Цель работы	3
2	Задание	4
3	Выполнение лабораторной работы	5
4	Контрольные вопросы	9
5	Выводы	14

1 Цель работы

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

2 Задание

Написать программы

3 Выполнение лабораторной работы

1. Первая программа-скрипт копирует файл в папку backup

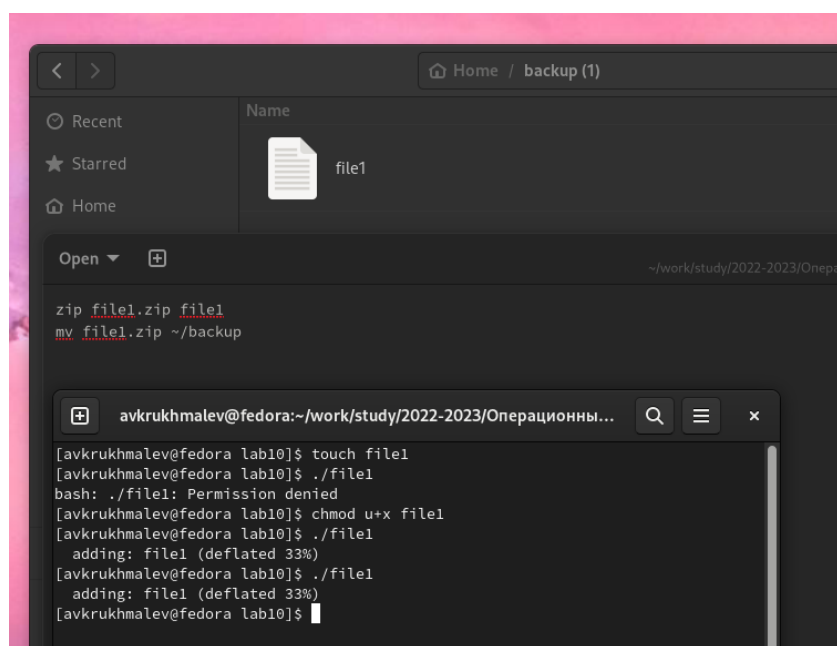



Рис. 3.1: 1

2. Вторая программа

```
[avkrukhmalev@fedora lab10]$ chmod u+x file2
[avkrukhmalev@fedora lab10]$ ./file2 1 2 3 4 5 123
1
2
3
4
5
123
[avkrukhmalev@fedora lab10]$
```

Open ▾ 

```
for A in $*
do echo $A
done
```

Рис. 3.2: 2

3. Третья программа выводит информацию о каталоге

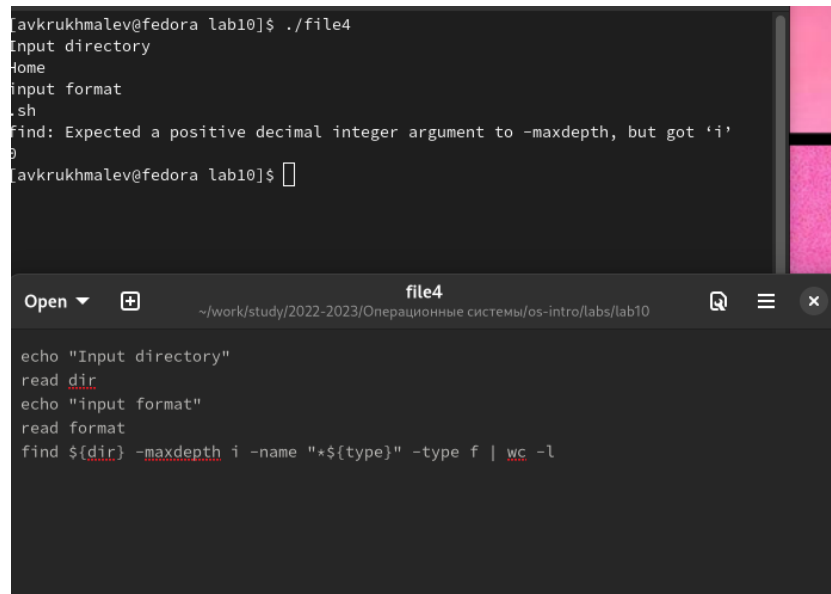
```
123
[avkrukhmalev@fedora lab10]$ touch file3
[avkrukhmalev@fedora lab10]$ chmod u+x file3
[avkrukhmalev@fedora lab10]$ ./file3
file1: is a file andwriteable
file2: is a file andwriteable
file3: is a file andwriteable
presentation: is a directory
report: is a directory
[avkrukhmalev@fedora lab10]$

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. 3.3: 3

4. Четвертая программа ищет файлы с нужным форматом, в нужной директории

```
[avkrukhmalev@fedora lab10]$ ./file4
Input directory
Home
input format
sh
find: Expected a positive decimal integer argument to -maxdepth, but got 'i'
[avkrukhmalev@fedora lab10]$
```



```
Open ▾ + file4
~/work/study/2022-2023/Операционные системы/os-intro/labs/lab10

echo "Input directory"
read dir
echo "input format"
read format
find ${dir} -maxdepth i -name "${type}" -type f | wc -l
```

Рис. 3.4: 4

4 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? 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;

- 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);

5 Выводы

Научился писать небольшие командные файлы