

# **Лабораторная работа №10**

**Дисциплина: Операционные системы**

Гибшер Кирилл Владимирович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
3.1	Переменные в языке программирования bash . . . . .	7
3.2	Использование арифметических вычислений. Операторы let и read.	8
3.3	Метасимволы и их экранирование . . . . .	10
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>12</b>
<b>5</b>	<b>Вывод</b>	<b>16</b>

## Список иллюстраций

4.1	Создание каталога и файла . . . . .	12
4.2	Текст первого скрипта . . . . .	12
4.3	Запуск и проверка . . . . .	13
4.4	Текст второго скрипта . . . . .	13
4.5	Запуск и проверка работоспособности скрипта . . . . .	13
4.6	Текст третьего скрипта . . . . .	14
4.7	Запуск и проверка работоспособности скрипта . . . . .	14
4.8	Текст четвертого скрипта . . . . .	15
4.9	Запуск и проверка работоспособности скрипта . . . . .	15

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

## 3 Теоретическое введение

### 3.1 Переменные 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. Например, команда

- `mark=/usr/andy/bin` присваивает значение строки символов `/usr/andy/bin` переменной `mark` типа строка 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 `$`. Например, команда

- `mv afile ${mark}` переместит файл `afile` из текущего каталога в каталог с абсолютным полным именем `/usr/andy/bin`.

Использование значения, присвоенного некоторой переменной, называется подстановкой. Для того чтобы имя переменной не сливалось с символами, которые могут следовать за ним в командной строке, при подстановке в общем случае используется следующая форма записи: `${имя переменной}`

## 3.2 Использование арифметических вычислений.

### Операторы `let` и `read`.

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 (%). Команда `let` берет два операнда и присваивает их переменной. Положительным моментом команды `let` можно считать то, что для идентификации переменной ей не нужен знак доллара; вы можете писать команды типа `let sum=x+7`, и `let` будет искать переменную `x` и добавлять к ней 7. Команда `let` также расширяет другие выражения `let`, если они заключены в двойные круглые скобки. Таким способом вы можете создавать довольно сложные выражения. Команда `let` не ограничена простыми арифметическими выражениями. Табл. 10.1 показывает полный набор `let`-операций. Подобно `C` оболочка `bash` может присваивать переменной любое значение, а произвольное выражение само имеет значение, которое может использоваться. При этом «ноль» воспринимается как «ложь», а любое другое значение выражения — как «истина». Для облегчения программирования можно записывать условия оболочки `bash` в двойные скобки — `(( ))`. Можно присваивать результаты условных выражений переменным, также как и использовать результаты арифметических вычислений в качестве условий.

Наиболее распространённым является сокращение, избавляющееся от слова `let` в программах оболочек. Если объявить переменные целыми значениями, то



любое присвоение автоматически будет трактоваться как арифметическое действие. Если использовать `typeset -i` для объявления и присвоения переменной, то при последующем её применении она станет целой. Также можно использовать ключевое слово `integer` (псевдоним для `typeset -i`) и объявлять таким образом переменные целыми. Выражения типа `x=y+z` будет восприниматься в это случае как арифметические. Команда `read` позволяет читать значения переменных со стандартного ввода

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. В командном процессоре

Существует ещё несколько стандартных переменных. Значение всех переменных можно посмотреть с помощью команды `set`.

### 3.3 Метасимволы и их экранирование

При перечислении имён файлов текущего каталога можно использовать следующие символы: `*` — соответствует произвольной, в том числе и пустой строке;

- `?` — соответствует любому одинарному символу;

`[c1-c2]` — соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. Например,

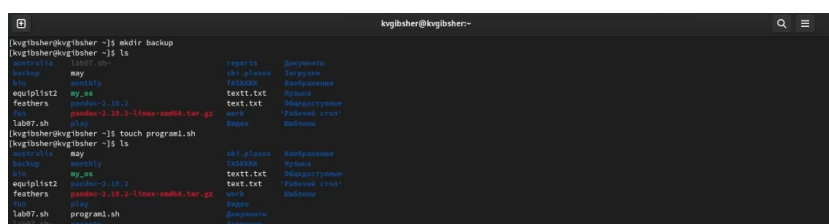
- `echo *` — выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`;
- `ls *.c` — выведет все файлы с последними двумя символами, совпадающими с `.c`.
- `echo prog.?` — выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.`
- `[a-z]*` — соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

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

## 4 Выполнение лабораторной работы

1. Создадим каталог backup и sh файл program1 (рис. [4.1])



```
kvgibsher@kvgibsher:~$ mkdir backup
kvgibsher@kvgibsher:~$ ls
Australia  lab07.sh  reports  documents
backup    my       sh.plans  backup
etc       weekly   tasks    backup
equiplist2 my_sh    text.txt  backup
feathers  pandoc-2.19.3-linux-amd64.tar.gz  text.txt  backup
lab07.sh  pandoc-2.19.3-linux-amd64.tar.gz  work      backup
kvgibsher@kvgibsher:~$ touch program1.sh
kvgibsher@kvgibsher:~$ ls
Australia  lab07.sh  reports  documents
backup    my       sh.plans  backup
etc       weekly   tasks    backup
equiplist2 my_sh    text.txt  backup
feathers  pandoc-2.19.3-linux-amd64.tar.gz  text.txt  backup
lab07.sh  pandoc-2.19.3-linux-amd64.tar.gz  work      backup
program1.sh
```

Рис. 4.1: Создание каталога и файла

2. Напишем скрипт к 1 заданию. Благодаря данной записи наш файл при запуске файла архивирует свою собственную резервную копию в созданный нами каталог ~/backup (рис. [4.2])



```
#!/bin/bash
tar -czf ~/backup/backup.tar program1.sh
```

Рис. 4.2: Текст первого скрипта

3. Запустим готовый файл и проверим получилось ли выполнить задание перейдя в каталог ~/backup (рис. [4.3])

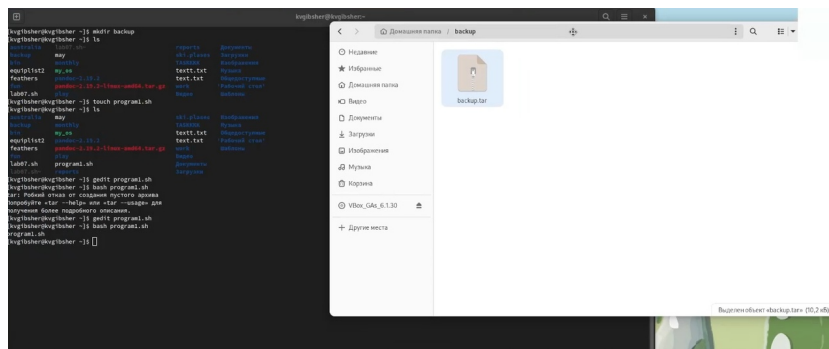


Рис. 4.3: Запуск и проверка

4. Напишем скрипт ко 2 заданию. Благодаря данной записи при запуске файла мы сможем ввести некоторые значения и командная строка по завершению обработки последовательно выведет нам те же самые значения, при чем даже больше 10 (рис. [4.4])



Рис. 4.4: Текст второго скрипта

5. Запустим готовый файл и проверим работоспособность скрипта.(рис. [4.5])

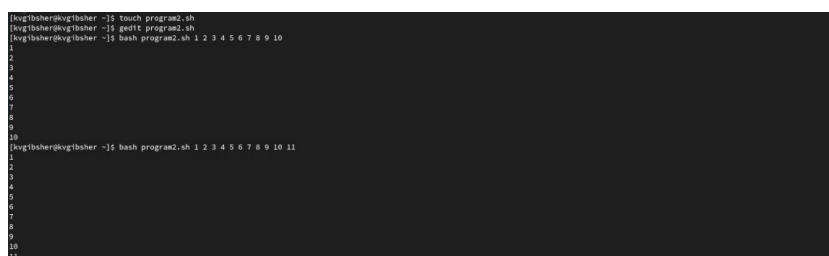


Рис. 4.5: Запуск и проверка работоспособности скрипта

6. Используем приложенный к лабораторной работе скрипт в разделе условного оператора `if`, отредактируем и получим следующий скрипт, который сделает из исполняемого файла некоторый аналог команды `ls`. Файл будет выдывать информацию о нужном каталоге и выведет информацию о возможностях доступа к файлам данного каталога. (рис. [4.6])

```
1 #!/bin/bash
2 for i in *
3 do
4     if test -d $i
5     then
6         echo $i is a directory
7     else
8         echo -n $i is a file and
9         if test -w $i
10        then
11            echo writable
12        elif test -r $i
13        then
14            echo readable
15        else
16            echo neither readable nor writable
17        fi
18    fi
19 done
```

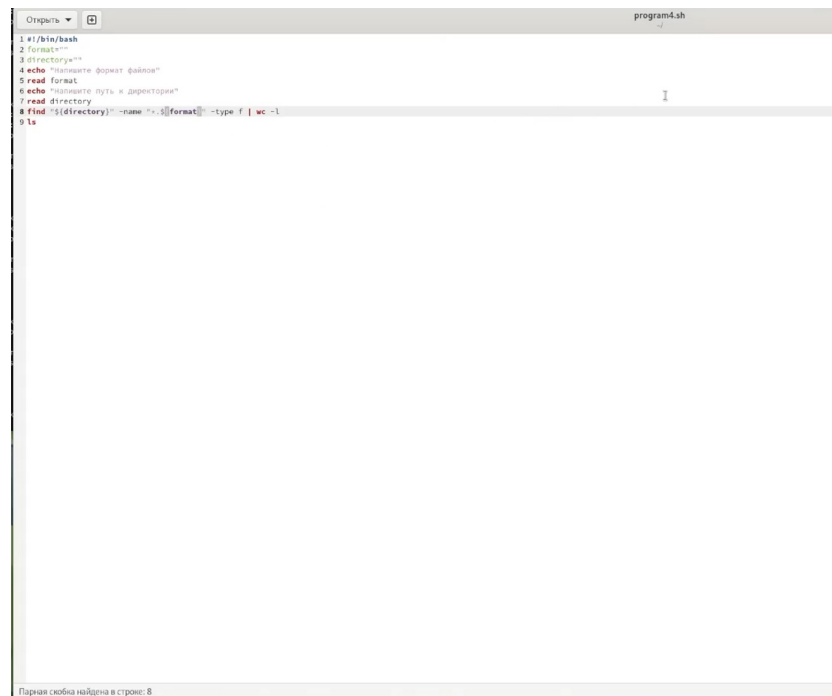
Рис. 4.6: Текст третьего скрипта

7. Запусти исполняемый файл в домашнем каталоге и проверим работоспособность скрипта (рис. [4.7])

```
lvgibshergvgibsh@ ~$ touch program3.sh
lvgibshergvgibsh@ ~$ chmod 755 program3.sh
lvgibshergvgibsh@ ~$ ./program3.sh
australia: is a directory
backups: is a directory
bin: is a directory
equipt1t2: is a file andrwriteable
readable
feathers: is a file andrwriteable
readable
fun: is a directory
lab07.sh: is a file andrwriteable
readable
lab07.sh=: is a file andrwriteable
readable
map: is a file andrwriteable
readable
monthly: is a directory
nv.sh: is a file andpandoc-2.19.2: is a directory
pandoc-2.19.2-linux-umd64.tar.gz: is a file andrwriteable
readable
play: is a directory
program1.sh: is a file andrwriteable
readable
program2.sh: is a file andrwriteable
readable
program3.sh: is a file andrwriteable
readable
reports: is a directory
ski.places: is a directory
TASKMKK: is a directory
text.txt: is a file andrwriteable
readable
text.txt: is a file andrwriteable
readable
```

Рис. 4.7: Запуск и проверка работоспособности скрипта

8. Напишем текст четвертого скрипта , благодаря которому исполняемый файл будет настроен на подсчет файлов желаемого формата. Командная строка запросит ввести искомый формат файлов и путь к деректории в которой будет производить поиск и подсчет.(рис. [4.8])

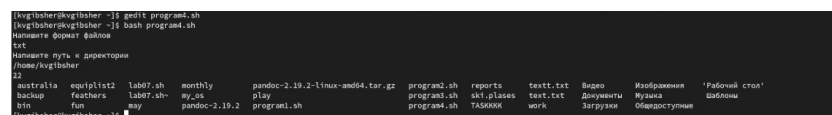


```
1 #!/bin/bash
2 format=""
3 directory=""
4 echo "Напишите формат файлов"
5 read format
6 echo "Напишите путь к деректории"
7 read directory
8 find $(directory) -name "*.${format}" -type f | wc -l
9 ls
```

Парная скобка найдена в строке: 8

Рис. 4.8: Текст четвертого скрипта

9. Запустим исполняемый файл , укажем формат файлов txt и каталогом для подсчета и поиска выберем домашнюю папку пользователя kvgibsher (рис. [4.9])



```
kvgibsher@kvgibsher ~$ gedit program4.sh
kvgibsher@kvgibsher ~$ ./program4.sh
Напишите формат файлов
txt
Напишите путь к деректории
/home/kvgibsher
22
australia  equiplist2  lab07.sh  monthly  pandoc-2.19.2-linux-amd64.tar.gz  program1.sh  reports  testt.txt  Видео  Изображения  'Рабочий стол'
backup      feathers    lab07.sh  my_os    play      program1.sh  program1.sh  testt.txt  Документы  Музыка  Шаблоны
bin         fun        my       pandoc-2.19.2  program1.sh  program1.sh  TASKBOOK  work      Загрузки  Обои/утиль
kvgibsher@kvgibsher ~$
```

Рис. 4.9: Запуск и проверка работоспособности скрипта

## 5 Вывод

Я изучил основы программирования в оболочке ОС UNIX/Linux. Научился писать небольшие командные файлы.