

# **Отчёт по лабораторной работе №9:**

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

Касакьянц Владислав Сергеевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>4</b>	<b>Контрольные вопросы</b>	<b>11</b>
<b>5</b>	<b>Выводы</b>	<b>15</b>
	<b>Список литературы</b>	<b>16</b>

## Список иллюстраций

3.1	Скрипт для создания архивной копии . . . . .	6
3.2	Результат работы скрипта №1 . . . . .	7
3.3	Скрипт вывода аргументов командной строки . . . . .	7
3.4	Результат работы скрипта №2 . . . . .	8
3.5	Скрипт аналога команды ls . . . . .	8
3.6	Результат работы скрипта №3 . . . . .	9
3.7	Скрипт подсчета количества файлов нужного формата . . . . .	10
3.8	Результат работы скрипта №4 . . . . .	10

# 1 Цель работы

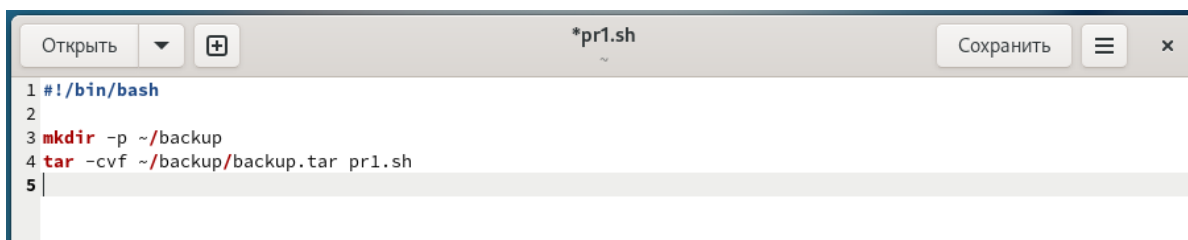
Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

## 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 Выполнение лабораторной работы

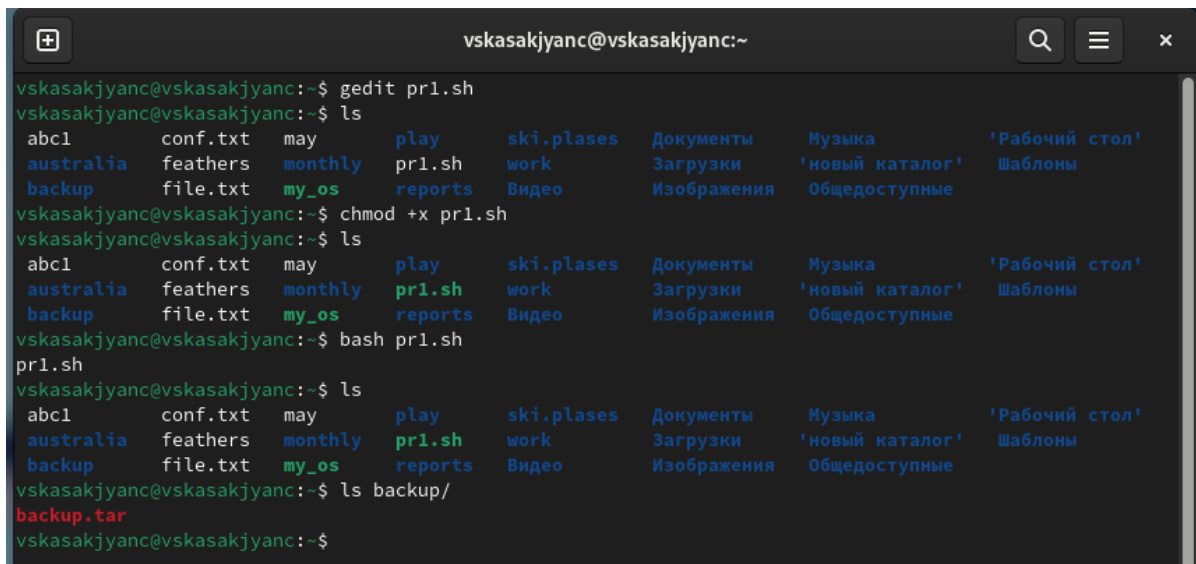
1. Напишем скрипт, который при запуске будет делать резервную копию самого себя в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar (рис. 3.1):



```
1 #!/bin/bash
2
3 mkdir -p ~/backup
4 tar -cvf ~/backup/backup.tar pr1.sh
5
```

Рис. 3.1: Скрипт для создания архивной копии

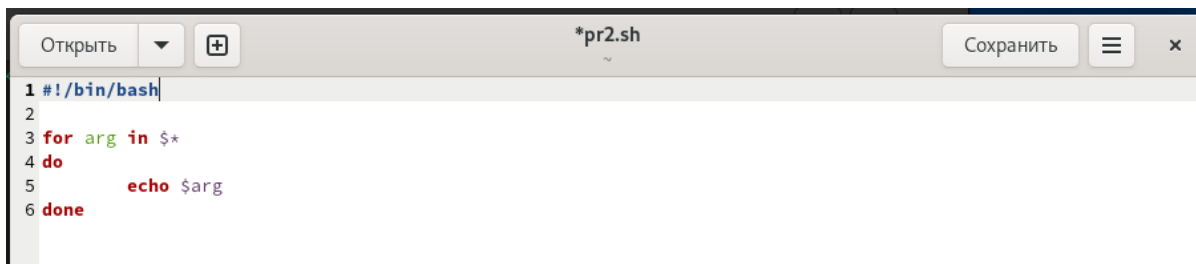
Сделаем файл исполняемым с помощью и проверяем, сработал ли скрипт. На скриншоте можно увидеть, что все работает (рис. 3.2).



```
vskasakjyanc@vskasakjyanc:~$ gedit pr1.sh
vskasakjyanc@vskasakjyanc:~$ ls
abcl      conf.txt  may      play      ski.plases  Документы  Музыка  'Рабочий стол'
australia feathers  monthly  pr1.sh     work        Загрузки  'новый каталог'  Шаблоны
backup    file.txt  my_os    reports    Видео        Изображения  Общедоступные
vskasakjyanc@vskasakjyanc:~$ chmod +x pr1.sh
vskasakjyanc@vskasakjyanc:~$ ls
abcl      conf.txt  may      play      ski.plases  Документы  Музыка  'Рабочий стол'
australia feathers  monthly  pr1.sh     work        Загрузки  'новый каталог'  Шаблоны
backup    file.txt  my_os    reports    Видео        Изображения  Общедоступные
vskasakjyanc@vskasakjyanc:~$ bash pr1.sh
pr1.sh
vskasakjyanc@vskasakjyanc:~$ ls
abcl      conf.txt  may      play      ski.plases  Документы  Музыка  'Рабочий стол'
australia feathers  monthly  pr1.sh     work        Загрузки  'новый каталог'  Шаблоны
backup    file.txt  my_os    reports    Видео        Изображения  Общедоступные
vskasakjyanc@vskasakjyanc:~$ ls backup/
backup.tar
vskasakjyanc@vskasakjyanc:~$
```

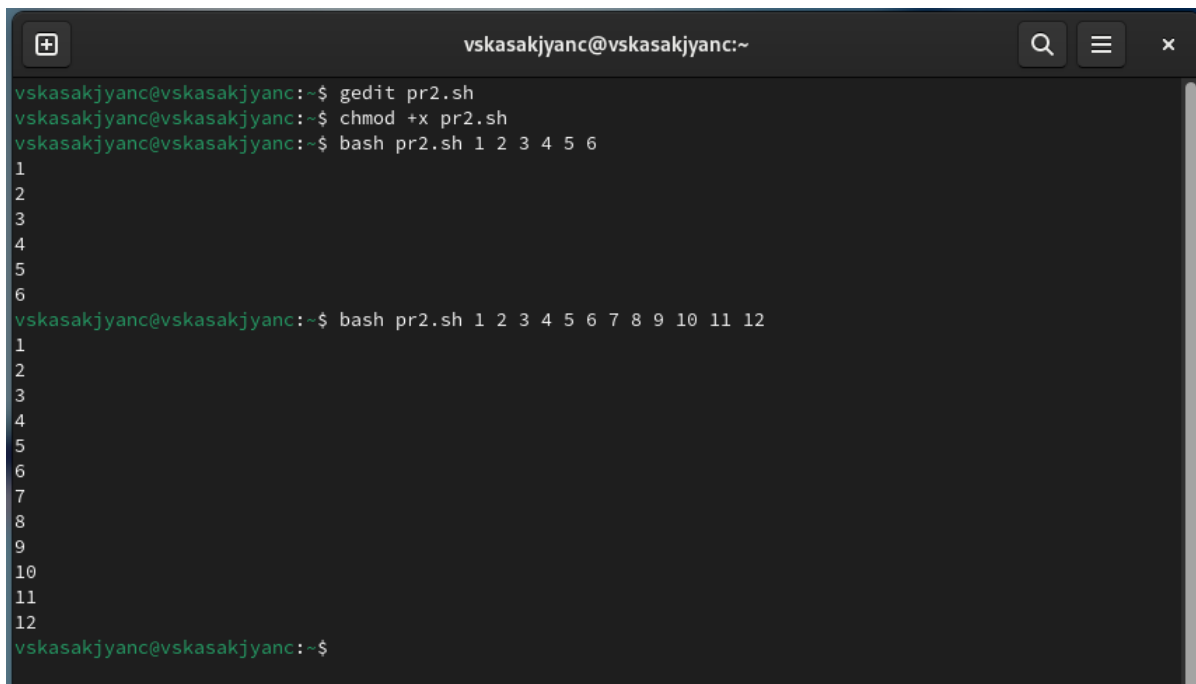
Рис. 3.2: Результат работы скрипта №1

2. Напишем пример командного файла, обрабатывающего любое произвольное число аргументов командной строки (рис. 3.3), (рис. 3.4).:



```
*pr2.sh
1 #!/bin/bash
2
3 for arg in $*
4 do
5     echo $arg
6 done
```

Рис. 3.3: Скрипт вывода аргументов командной строки



```
vskasakjyanc@vskasakjyanc:~$ gedit pr2.sh
vskasakjyanc@vskasakjyanc:~$ chmod +x pr2.sh
vskasakjyanc@vskasakjyanc:~$ bash pr2.sh 1 2 3 4 5 6
1
2
3
4
5
6
vskasakjyanc@vskasakjyanc:~$ bash pr2.sh 1 2 3 4 5 6 7 8 9 10 11 12
1
2
3
4
5
6
7
8
9
10
11
12
vskasakjyanc@vskasakjyanc:~$
```

Рис. 3.4: Результат работы скрипта №2

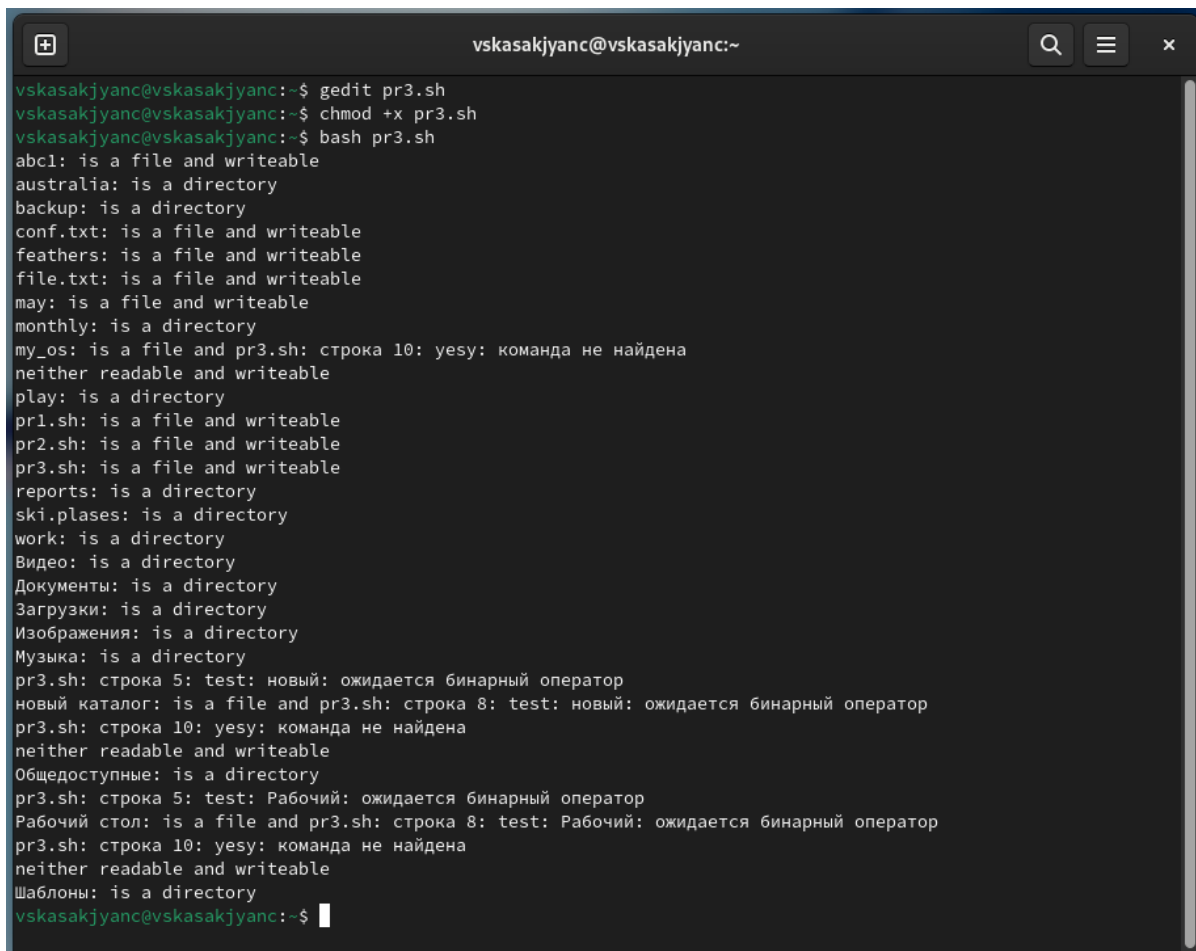
3. Напишем командный файл — аналог команды `ls` (без использования самой этой команды и команды `dir`), который выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога (рис. 3.5), (рис. 3.6):



```
Открыть  *pr3.sh  Сохранить
1 #!/bin/bash
2
3 for A in *
4 do
5 if test -d $A
6 then echo "$A: is a directory"
7 else echo -n "$A: is a file and "
8 if test -w $A
9 then echo writeable
10 elif test -r $A
11 then echo readable
12 else
13 echo neither readable and writeable
14 fi
15 fi
16 done
```

Рис. 3.5: Скрипт аналога команды `ls`

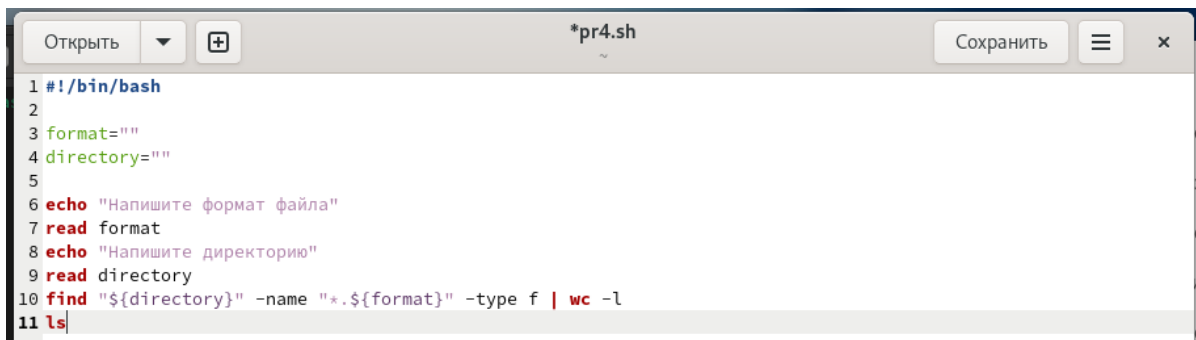




```
vskasakjyanc@vskasakjyanc:~$ gedit pr3.sh
vskasakjyanc@vskasakjyanc:~$ chmod +x pr3.sh
vskasakjyanc@vskasakjyanc:~$ bash pr3.sh
abc1: is a file and writeable
australia: is a directory
backup: is a directory
conf.txt: is a file and writeable
feathers: is a file and writeable
file.txt: is a file and writeable
may: is a file and writeable
monthly: is a directory
my_os: is a file and pr3.sh: строка 10: yesy: команда не найдена
neither readable and writeable
play: is a directory
pr1.sh: is a file and writeable
pr2.sh: is a file and writeable
pr3.sh: is a file and writeable
reports: is a directory
ski.places: is a directory
work: is a directory
Видео: is a directory
Документы: is a directory
Загрузки: is a directory
Изображения: is a directory
Музыка: is a directory
pr3.sh: строка 5: test: новый: ожидается бинарный оператор
новый каталог: is a file and pr3.sh: строка 8: test: новый: ожидается бинарный оператор
pr3.sh: строка 10: yesy: команда не найдена
neither readable and writeable
Общедоступные: is a directory
pr3.sh: строка 5: test: Рабочий: ожидается бинарный оператор
Рабочий стол: is a file and pr3.sh: строка 8: test: Рабочий: ожидается бинарный оператор
pr3.sh: строка 10: yesy: команда не найдена
neither readable and writeable
Шаблоны: is a directory
vskasakjyanc@vskasakjyanc:~$
```

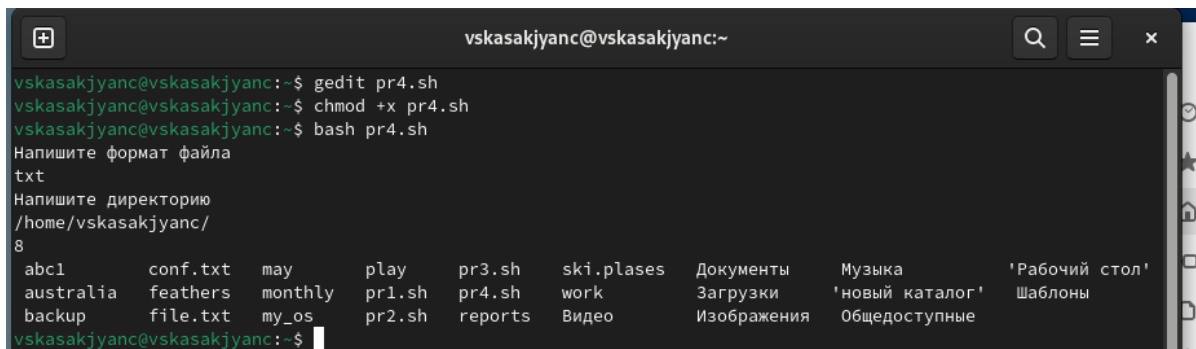
Рис. 3.6: Результат работы скрипта №3

4. 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. (рис. 3.7), (рис. 3.8):



```
1 #!/bin/bash
2
3 format=""
4 directory=""
5
6 echo "Напишите формат файла"
7 read format
8 echo "Напишите директорию"
9 read directory
10 find "${directory}" -name ".*${format}" -type f | wc -l
11 ls
```

Рис. 3.7: Скрипт подсчета количества файлов нужного формата



```
vskasakjyanc@vskasakjyanc:~$ gedit pr4.sh
vskasakjyanc@vskasakjyanc:~$ chmod +x pr4.sh
vskasakjyanc@vskasakjyanc:~$ bash pr4.sh
Напишите формат файла
txt
Напишите директорию
/home/vskasakjyanc/
8
abc1      conf.txt  may       play      pr3.sh    ski.plases  Документы  Музыка    'Рабочий стол'
australia feathers  monthly   pr1.sh    pr4.sh    work       Загрузки   'новый каталог'  Шаблоны
backup    file.txt  my_os     pr2.sh    reports   Видео       Изображения  Общедоступные
vskasakjyanc@vskasakjyanc:~$
```

Рис. 3.8: Результат работы скрипта №4

## 4 Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?

**Командная оболочка** – это интерфейс между пользователем и операционной системой, который позволяет пользователю взаимодействовать с операционной системой путем ввода текстовых команд. Примеры командных оболочек включают Bash (Bourne Again Shell), Zsh (Z Shell), Fish (Friendly Interactive Shell) и другие. Они отличаются по своим возможностям, синтаксису, встроенным функциям и поддерживаемым расширениям.

2. Что такое POSIX?

**POSIX** (Portable Operating System Interface) – это семейство стандартов, разработанных для обеспечения совместимости между различными операционными системами Unix. Он определяет общие интерфейсы для программирования на языке C, командной строки и управления файлами.

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

В языке программирования bash переменные определяются путем присваивания значений их именам. Например:

- Переменные: `variable_name=value`
- Массивы: `array_name[index]=value`

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

**Оператор `let`** используется для выполнения арифметических выражений в `bash`. **Оператор `read`** используется для считывания значений из стандартного ввода и присваивания их переменным.

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

В языке программирования `bash` можно применять стандартные арифметические операции, такие как сложение, вычитание, умножение и деление.

#### 6. Что означает операция `(( ))`?

Операция `(( ))` в `bash` используется для выполнения арифметических вычислений.

#### 7. Какие стандартные имена переменных Вам известны?

Некоторые стандартные имена переменных в `bash`:

- `HOME`: домашний каталог текущего пользователя.
- `PWD`: текущий рабочий каталог.
- `PATH`: список каталогов, в которых операционная система ищет исполняемые файлы.
- `USER`: имя текущего пользователя.

#### 8. Что такое метасимволы?

**Метасимволы** – это символы, которые имеют специальное значение в контексте командной строки или шаблонов файлов. Некоторые примеры метасимволов включают `*`, `?`, `[ ]`, `{ }`, `|`, `;` и `&`.

#### 9. Как экранировать метасимволы?

Для экранирования метасимволов в `bash` используется обратная косая черта `\`. Например, чтобы использовать символ `*` как обычный символ, его можно экранировать так: `\*`.

#### 10. Как создавать и запускать командные файлы?

Для создания и запуска командных файлов в `bash` можно использовать текстовый редактор для создания файла с расширением `.sh`, затем присвоить ему права на выполнение с помощью команды `chmod +x filename.sh`, и, наконец, запустить файл с помощью команды `./filename.sh`.

#### 11. Как определяются функции в языке программирования `bash`?

Функции в языке программирования `bash` определяются с использованием ключевого слова `function` или просто с именем функции, после чего идет блок кода. Например:

```
function my_function {  
    # Код функции  
}
```

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

Для определения, является ли файл каталогом или обычным файлом, можно использовать команду `test`. Например:

- Проверка на каталог: `test -d filename`
- Проверка на обычный файл: `test -f filename`

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

Команды `set`, `typeset` и `unset` используются для работы с переменными в `bash`:

- `set`: устанавливает значения и флаги для параметров командной строки.
- `typeset`: используется для объявления переменных с определенными свойствами, такими как `readonly` или `integer`.
- `unset`: удаляет значения переменных.

#### 14. Как передаются параметры в командные файлы?

Параметры передаются в командные файлы в виде аргументов командной строки. Они доступны внутри скрипта через специальные переменные `$1`, `$2`, `$3` и так далее, где `$1` содержит первый аргумент, `$2` – второй и т.д.

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

Некоторые специальные переменные языка `bash` и их назначение:

- `$0`: имя текущей выполняемой программы.
- `$#`: количество аргументов, переданных скрипту.
- `$?`: код возврата последней выполненной команды.
- `$$`: PID (идентификатор процесса) текущего скрипта.
- `$_`: PID последнего запущенного фонового процесса.

## **5 Выводы**

В данной лабораторной работе мы изучили основы программирования в оболочке ОС UNIX/Linux, а также научились писать небольшие командные файлы.

## **Список литературы**