

Отчёт по лабораторной работе №10

**Программирование в командном процессоре ОС UNIX. Ветвления и
циклы**

Тагиев Павел Фаикович

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	7
3.1	Задание 1	7
3.2	Задание 2	8
3.3	Задание 3	10
3.4	Задание 4	11
4	Контрольные вопросы	14
5	Выводы	17
	Список литературы	18

Список иллюстраций

3.1	Результаты поиска	7
3.2	Вывод второго скрипта	10
3.3	Создание последовательности файлов	11
3.4	Создание архива	12

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов [1].

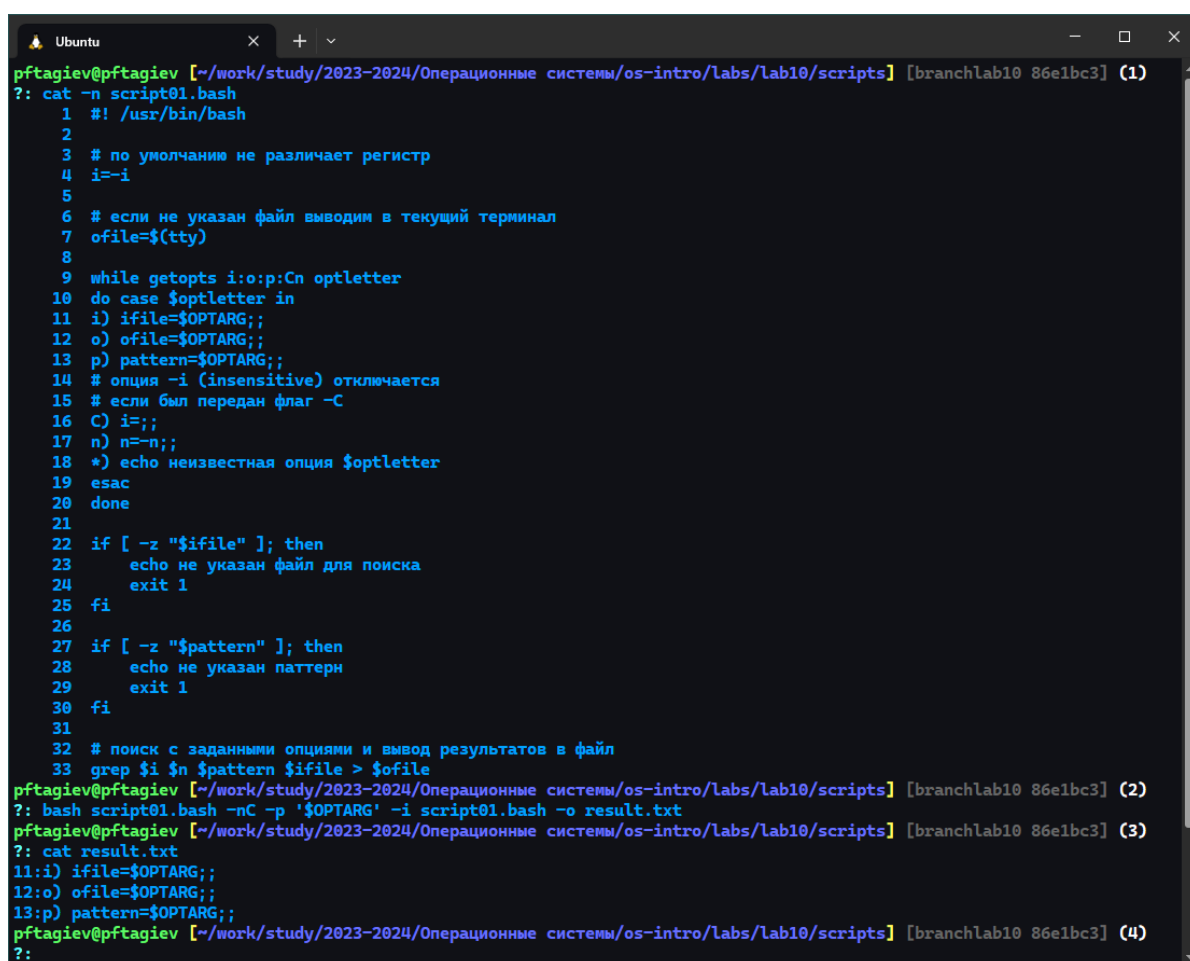
2 Задание

1. Используя команды `getopts`, `grep`, написать командный файл, который анализирует командную строку с ключами:
 - `-iinputfile` — прочитать данные из указанного файла;
 - `-ooutputfile` — вывести данные в указанный файл;
 - `-ршаблон` — указать шаблон для поиска;
 - `-C` — различать большие и малые буквы;
 - `-n` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-р`.
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (например `1.tmp`, `2.tmp`, `3.tmp`, `4.tmp` и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).
4. Написать командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее

недели тому назад (использовать команду [find](#))

3 Выполнение лабораторной работы

3.1 Задание 1



```
Ubuntu
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 86e1bc3] (1)
?: cat -n script01.bash
 1  #!/usr/bin/bash
 2
 3  # по умолчанию не различает регистр
 4  i=-i
 5
 6  # если не указан файл выводим в текущий терминал
 7  ofile=$(tty)
 8
 9  while getopts i:o:p:Cn optletter
10  do case $optletter in
11  i) ifile=$OPTARG;;
12  o) ofile=$OPTARG;;
13  p) pattern=$OPTARG;;
14  # опция -i (insensitive) отключается
15  # если был передан флаг -C
16  C) i=;;
17  n) n=-n;;
18  *) echo неизвестная опция $optletter
19  esac
20  done
21
22  if [ -z "$ifile" ]; then
23      echo не указан файл для поиска
24      exit 1
25  fi
26
27  if [ -z "$pattern" ]; then
28      echo не указан паттерн
29      exit 1
30  fi
31
32  # поиск с заданными опциями и вывод результатов в файл
33  grep $i $n $pattern $ifile > $ofile
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 86e1bc3] (2)
?: bash script01.bash -nC -p 'OPTARG' -i script01.bash -o result.txt
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 86e1bc3] (3)
?: cat result.txt
11:i) ifile=$OPTARG;;
12:o) ofile=$OPTARG;;
13:p) pattern=$OPTARG;;
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 86e1bc3] (4)
?:
```

Рис. 3.1: Результаты поиска

Выполним первое задание. Требуется написать программу, для поиска в файле строк содержащих указанный шаблон. Результат поиска нужно вывести в другой файл.

Код написанной программы получился достаточно длинным, поэтому я разместил его в конце раздела на лист. 3.5. В комментариях можно увидеть пояснения к коду.

Разберем результат. На рис. 3.1 в промте (1) в терминал выводится файл скрипта, для удобства каждая строка нумеруется. В промте (2) я вызываю скрипт для поиска строки `$OPTARG` в файле `script01.bash`, результаты вывожу в `result.txt`. В самом конце можно увидеть результаты поиска, и сравнить их с выводом промта (1).

3.2 Задание 2

Листинг 3.1 Программа на Си

```
#include "stdlib.h"
#include "stdio.h"

int main() {
    printf("введите число: ");
    int num;
    scanf("%d", &num);
    exit(num < 0 ? 0 : num > 0 ? 1 : 2);
    // return num < 0 ? 0 : num > 0 ? 1 : 2;
    // ^^ можно так ^^ не нужно будет подключать stdlib
}
```

Напишем на языке Си программу, которая будет принимать число и в зависимости от него возвращать разные коды завершения, через функцию `exit`. А именно, если число меньше нуля, то вернется 0, если оно больше нуля вернется 1, если же оно равно нулю, то код возврата будет 2. Код Си программы можно увидеть на лист. 3.1

Теперь напомним скрипт, который будет вызывать нашу Си программу и обрабатывать ее код возврата (лист. 3.2). Пояснения к коду, как и в предыдущем задании, находятся в комментариях.

Листинг 3.2 Скрипт обрабатывающий код возврата

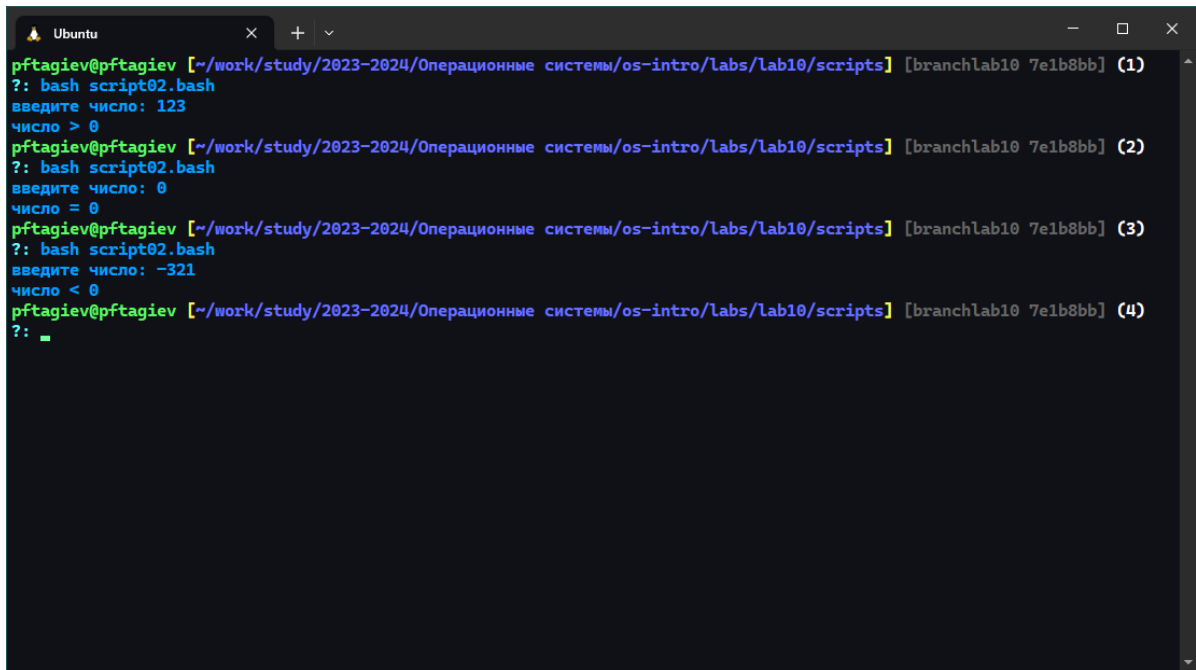
```
#!/usr/bin/bash

# собираем программу на Си если она не собрана
if ! [ -f app ]; then
    gcc app.c -o app
fi

# вызов собранной Си программы
./app

# получение кода завершения выполненной программы
# и его обработка
case $? in
0) echo 'число < 0';;
1) echo 'число > 0';;
2) echo 'число = 0';;
esac
```

Результат работы скрипта можно увидеть на рис. 3.2.



```
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (1)
?: bash script02.bash
введите число: 123
число > 0
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (2)
?: bash script02.bash
введите число: 0
число = 0
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (3)
?: bash script02.bash
введите число: -321
число < 0
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (4)
?: -
```

Рис. 3.2: Вывод второго скрипта

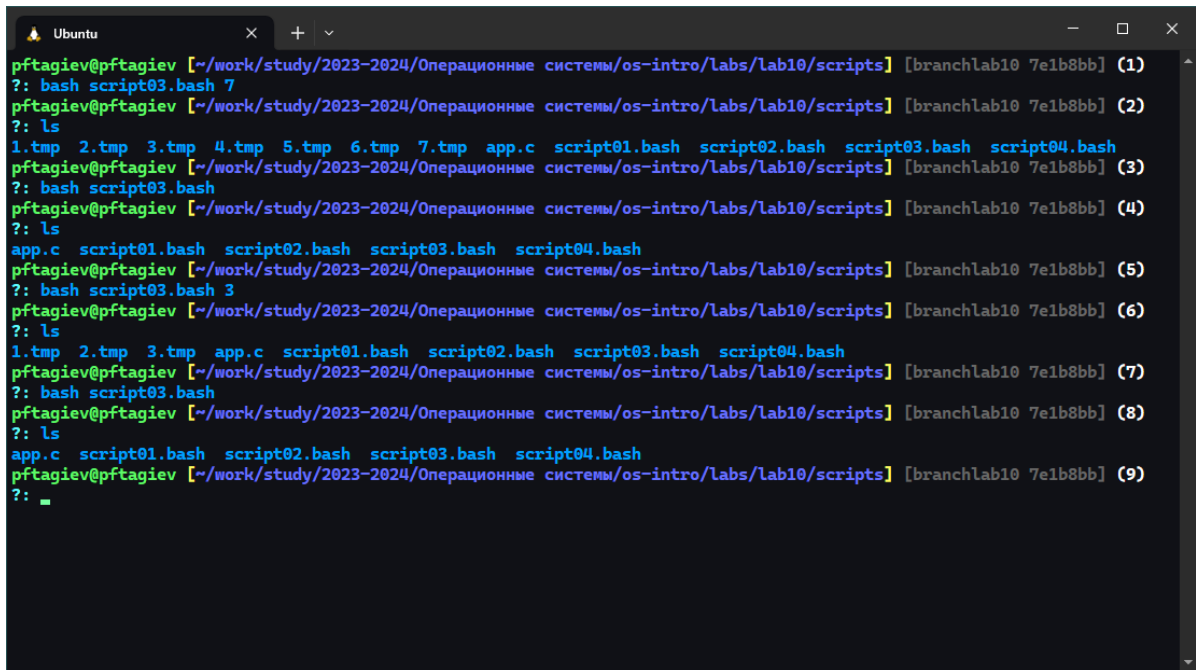
3.3 Задание 3

Листинг 3.3 Создание последовательности файлов

```
#!/usr/bin/bash

if [ -n "$1" ]; then
    # файлы создаются если передано число
    eval touch {1..$1}.tmp
    # если ничего не передано удаляются по шаблону
    else find * -maxdepth 1 -type f -regex "[1-9][0-9]*\.tmp$" -delete
fi
```

На лист. 3.3 можно увидеть выполненное третье задание. Если передать скрипту число N , он создаст последовательность файлов: 1.tmp 2.tmp ... N.tmp. Если вызвать его без аргументов командной строки, он удалит все созданные файлы. Результат работы скрипта можно увидеть на рис. 3.3



```
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (1)
?: bash script03.bash 7
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (2)
?: ls
1.tmp 2.tmp 3.tmp 4.tmp 5.tmp 6.tmp 7.tmp app.c script01.bash script02.bash script03.bash script04.bash
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (3)
?: bash script03.bash
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (4)
?: ls
app.c script01.bash script02.bash script03.bash script04.bash
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (5)
?: bash script03.bash 3
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (6)
?: ls
1.tmp 2.tmp 3.tmp app.c script01.bash script02.bash script03.bash script04.bash
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (7)
?: bash script03.bash
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (8)
?: ls
app.c script01.bash script02.bash script03.bash script04.bash
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (9)
?: -
```

Рис. 3.3: Создание последовательности файлов

3.4 Задание 4

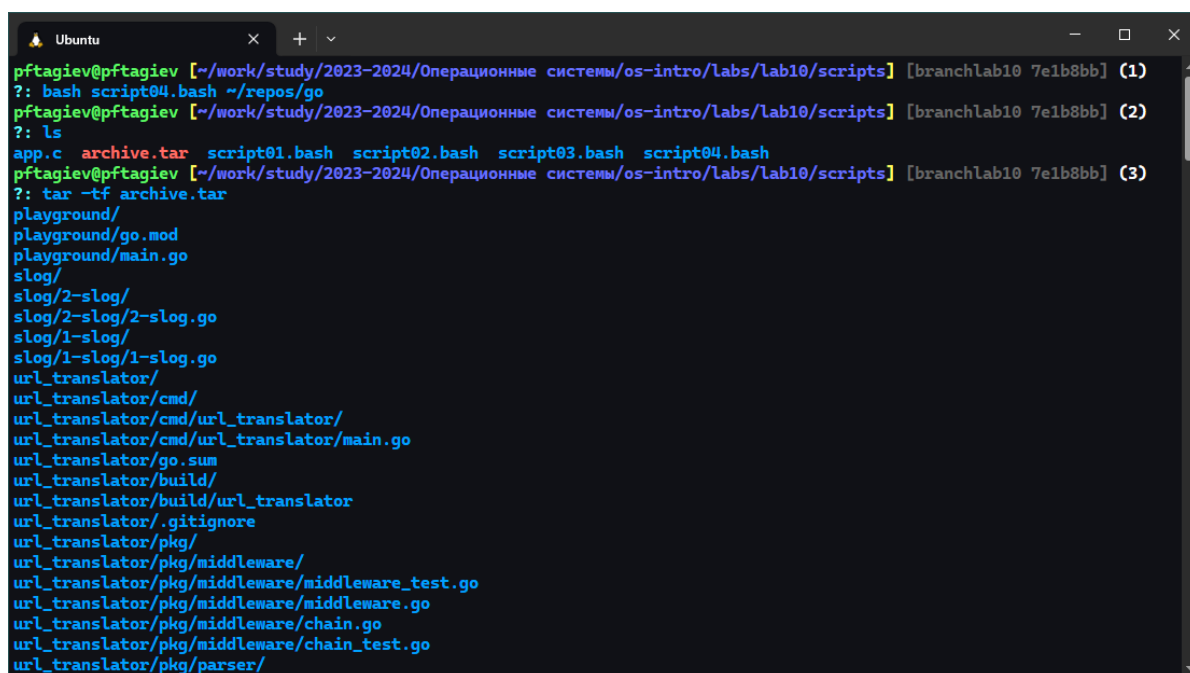
Требуется написать командный файл, который запаковывает в архив все файлы в указанной директории. И модифицировать его так чтобы запаковывались только те файлы, которые были изменены менее недели тому назад.

Листинг 3.4 Создание архива

```
#!/usr/bin/bash

# создаю пустой архив
tar -c -f archive.tar -T /dev/null
# меняю рабочий каталог
cd $1
# все файлы, которые были изменены за последние 7 дней
# добавляются в созданный ранее архив
find * -mtime -7 -exec tar -u -f "$OLDPWD/archive.tar" {} \;
```

На лист. 3.4 можно увидеть написанный командный файл. Так как модификация во второй части задания требует минимальных изменений я написал скрипт сразу с ними.



```
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (1)
?: bash script04.bash ~/repos/go
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (2)
?: ls
app.c archive.tar script01.bash script02.bash script03.bash script04.bash
pftagiev@pftagiev [~/work/study/2023-2024/Операционные системы/os-intro/labs/lab10/scripts] [branchlab10 7e1b8bb] (3)
?: tar -tf archive.tar
playground/
playground/go.mod
playground/main.go
slog/
slog/2-slog/
slog/2-slog/2-slog.go
slog/1-slog/
slog/1-slog/1-slog.go
url_translator/
url_translator/cmd/
url_translator/cmd/url_translator/
url_translator/cmd/url_translator/main.go
url_translator/go.sum
url_translator/build/
url_translator/build/url_translator
url_translator/.gitignore
url_translator/pkg/
url_translator/pkg/middleware/
url_translator/pkg/middleware/middleware_test.go
url_translator/pkg/middleware/middleware.go
url_translator/pkg/middleware/chain.go
url_translator/pkg/middleware/chain_test.go
url_translator/pkg/parser/
```

Рис. 3.4: Создание архива

Разберем результаты работы скрипта (рис. 3.4). В промте (1) вызывается командный файл, в промте (3) я вывожу содержимое созданного архива.

Листинг 3.5 Поиск по файлу

```
#!/usr/bin/bash

# по умолчанию не различает регистр
i=-i

# если не указан файл выводим в текущий терминал
ofile=$(tty)

while getopts i:op:Cn optletter
do case $optletter in
i) ifile=$OPTARG;;
o) ofile=$OPTARG;;
p) pattern=$OPTARG;;
# опция -i (insensitive) отключается
# если был передан флаг -C
C) i=;;
n) n=-n;;
*) echo неизвестная опция $optletter
esac
done

if [ -z "$ifile" ]; then
    echo не указан файл для поиска
    exit 1
fi

if [ -z "$pattern" ]; then
    echo не указан паттерн
    exit 1
fi

# поиск с заданными опциями и вывод результатов в файл
grep $i $n $pattern $ifile > $ofile
```

4 Контрольные вопросы

1. Каково предназначение команды `getopts`?

С помощью `getopts` можно достаточно легко произвести разбор флагов переданных скрипту.

2. Какое отношение метасимволы имеют к генерации имён файлов?

Метасимволы играют важную роль в генерации имен файлов, особенно в командных оболочках *Unix* и *Linux*. Эти символы представляют собой специальные символы, которые интерпретируются оболочкой для выполнения шаблонного поиска и подстановки. Например:

- `*` — соответствует любому количеству символов в имени файла.
- `?` — соответствует ровно одному символу.
- `[...]` — соответствуют любому одному символу из перечисленных в скобках.

3. Какие операторы управления действиями вы знаете?

- *Условные операторы:*
 - `if`, `elif` и `else` — позволяют выполнять команды, основываясь на условиях.
 - `case` — выбор действий в зависимости от значения переменной.
- *Циклы:*
 - `for` — выполняет команды для списка значений.
 - `while` — цикл который выполняется пока условие истинно.
 - `until` — цикл который выполняется пока условие ложно.

4. Какие операторы используются для прерывания цикла?

В Bash для прерывания циклов используются следующие операторы:

- `break` — прерывает выполнение текущего цикла и передает управление на команду, следующую за циклом. Вы можете указать `break n`, где `n` - это количество уровней цикла, которые нужно прервать.
- `continue` — пропускает оставшуюся часть тела текущего цикла и переходит к следующей итерации. Аналогично `break`, можно использовать `continue n` для пропуска итераций во вложенных циклах.

5. Для чего нужны команды `false` и `true`?

Команды `true` и `false` в Bash являются простыми утилитами, которые возвращают статус выхода. Они используются в скриптах и условных операторах для управления логикой выполнения.

- `true` всегда возвращает статус выхода 0, что означает успех. Эта команда может быть использована в местах, где требуется гарантированно успешный результат, например, в бесконечных циклах (`while true; do ... done`) или как заглушка для функции, которая еще не реализована.
- `false` всегда возвращает статус выхода 1, что означает неудачу. Эта команда может быть использована для преднамеренного вызова ошибки или как условие, которое никогда не будет выполнено.

6. Что означает строка `if test -f man$s/$i.$s`, встреченная в командном файле?

В этой строке проверяется существование файла `man$s/$i.$s`, где `$s` и `$i` переменные подставляющиеся в имя файла.

7. Объясните различия между конструкциями `while` и `until`.

- `while` — цикл который выполняется пока условие истинно.

- `until` — цикл который выполняется пока условие ложно.

5 Выводы

В этой работе мы поближе познакомились с циклами в языке *bash*. Научились писать более сложные командные файлы используя логические управляющие конструкции.

Список литературы

1. Кулябов. Операционные системы. Москва: РУДН, 2016. 118 с.