

Отчёт к лабораторной работе №12

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

Кекишева Анастасия Дмитриевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
3.1	Выполнение 1-го пункта задания	9
3.2	Выполнение 2-го пункта задания	11
3.3	Выполнение 3-го пункта задания	13
3.4	Выполнение 4-го пункта задания	14
4	Вывод	17
5	Библиография	18

Список таблиц

Список иллюстраций

3.1	Командный файл, который анализирует командную строку с ключами	9
3.2	Текстовый файл	10
3.3	Результат выполнения командного файла	10
3.4	Программа на языке Си: cod02.c	11
3.5	Командный файл, вызывающий программу cod02.c	12
3.6	Результат работы программ	12
3.7	Командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N	13
3.8	Результат работы программы: создание и удаление	14
3.9	Командный файл, архивирует файлы в указанной директории . .	14
3.10	Вызов командного файла	15
3.11	Результат работы командного файла	15

1 Цель работы

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

2 Задание

Выполнить данные пункты и ответить на вопросы:

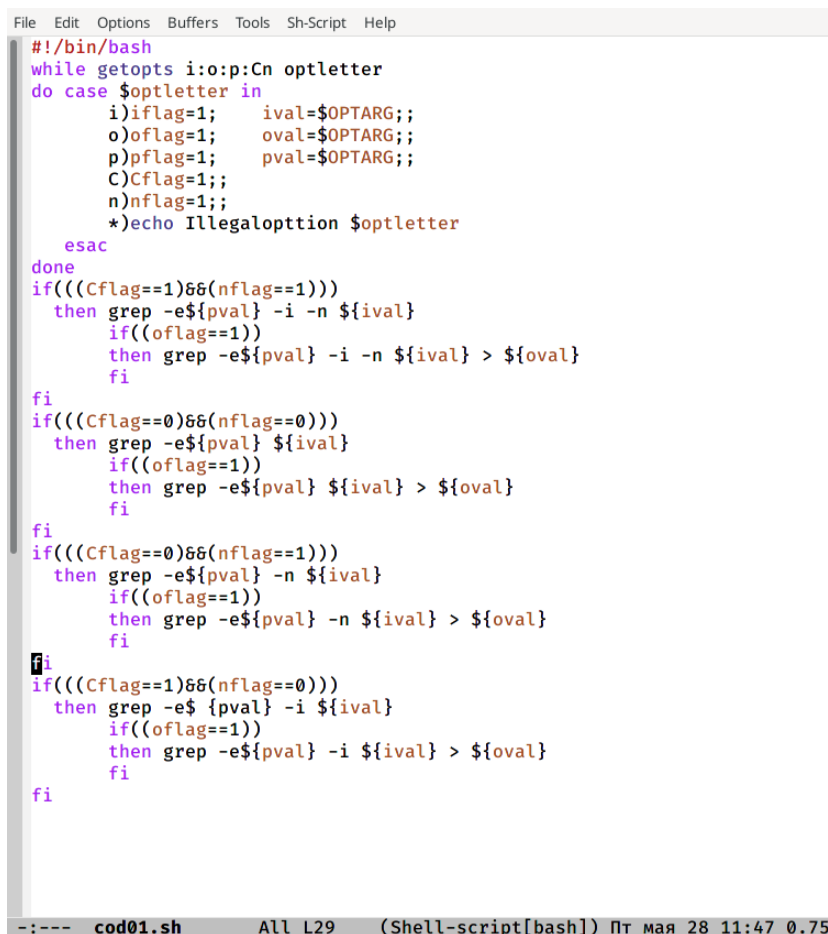
1. Используя команды `grep`, написать командный файл, который анализирует командную строку с ключами:
 - `-i` `inputfile`— прочитать данные из указанного файла;
 - `-o` `outputfile`— вывести данные в указанный файл;
 - `-r` `шаблон`— указать шаблон для поиска;
 - `-C`— различать большие и малые буквы;
 - `-n`— выдавать номера строк, а затем ищет в указанном файле нужные строки, определяемые ключом `-r`.
2. Написать на языке Си программу, которая вводит число и определяет, является 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 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

Перед выполнением лабораторной работы я хорошо ознакомилась с теоритическим материалом для её выполнения Ссылка 1

3.1 Выполнение 1-го пункта задания



```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
while getopts i:op:Cn optletter
do case $optletter in
    i)iflag=1;    ival=$OPTARG;;
    o)oflag=1;    oval=$OPTARG;;
    p)pflag=1;    pval=$OPTARG;;
    C)Cflag=1;;
    n)nflag=1;;
    *)echo Illegaloption $optletter
    esac
done
if(((Cflag==1)&&(nflag==1)))
then grep -e${pval} -i -n ${ival}
    if((oflag==1))
    then grep -e${pval} -i -n ${ival} > ${oval}
    fi
fi
if(((Cflag==0)&&(nflag==0)))
then grep -e${pval} ${ival}
    if((oflag==1))
    then grep -e${pval} ${ival} > ${oval}
    fi
fi
if(((Cflag==0)&&(nflag==1)))
then grep -e${pval} -n ${ival}
    if((oflag==1))
    then grep -e${pval} -n ${ival} > ${oval}
    fi
fi
if(((Cflag==1)&&(nflag==0)))
then grep -e$ {pval} -i ${ival}
    if((oflag==1))
    then grep -e${pval} -i ${ival} > ${oval}
    fi
fi
fi

-:--- cod01.sh All L29 (Shell-script[bash]) Пт мая 28 11:47 0.75
```

Рис. 3.1: Командный файл, который анализирует командную строку с ключами

Используя команды `getopts` `grep`, который анализирует командную строку с ключами. Во-первых, в цикле `while` применила команду `getopts`, которая будет распознавать аргумент, и если это так вернёт истину. Также для флагов, которые ожидают дополнительное значение, пишем `OPTARG`, который будет устанавливаться в значение этого аргумента. И в цикле с помощью оператора `case` я расписала операции, которые необходимо будет выполнить. Также для того чтобы выводить в указанном файле нужные строки, определяемые ключом `-p`, я проверяла условия, то есть чему в результате работы `case`, были равны флаги и после это командой `grep`, выбирала нужные строки, при этом если `Cflag=1` в `grep`

используем опцию -i, а если nflag=1 -n. (рис. @fig:001)

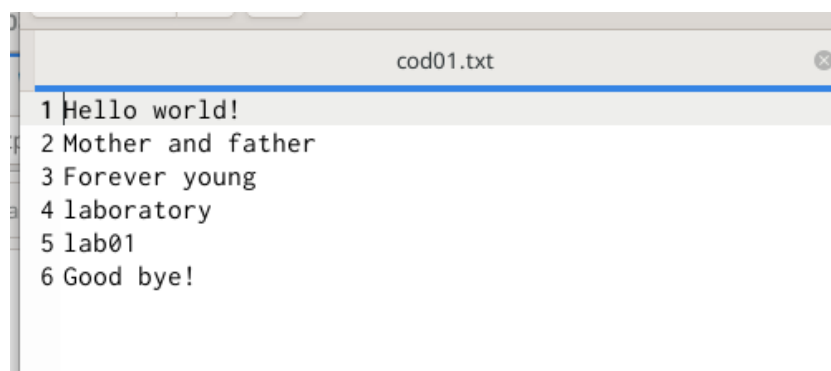


Рис. 3.2: Текстовый файл

```
adkeki sheva@dk8n76 ~ $ bash cod01.sh -icod01.txt -ocod.txt -plab -Cn
4:laboratory
5:lab01
adkeki sheva@dk8n76 ~ $ bash cod01.sh -icod01.txt -ocod.txt -plab
laboratory
lab01
adkeki sheva@dk8n76 ~ $ bash cod01.sh -icod01.txt -ocod.txt -pbye -n
6:Good bye!
```

Рис. 3.3: Результат выполнения командного файла

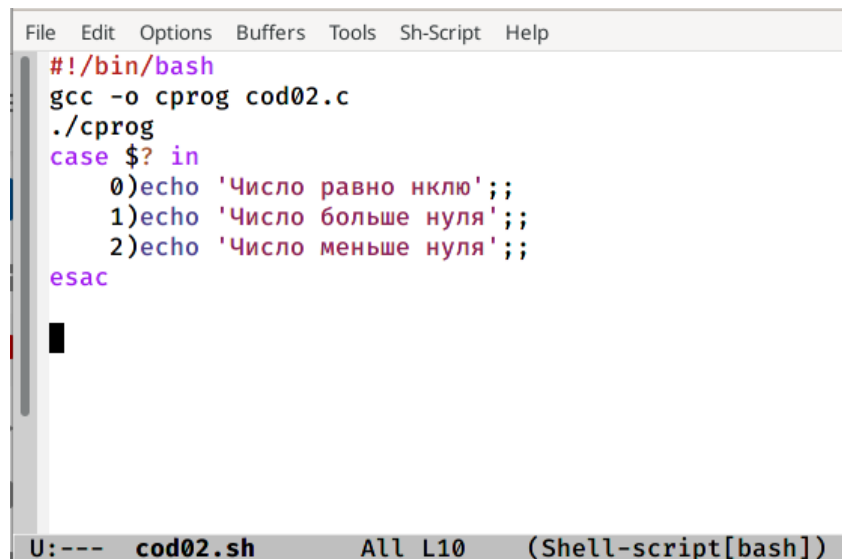
После написания командного файла, текстовый файл (рис. @fig:002) , после чего вызвала командный файл командой bash (командный файл) (новый файл) и далее желаемые опции+название текстового файла(например, -icod01.txt). Я искала в своём тексте lab и bye. (рис. @fig:003)

3.2 Выполнение 2-го пункта задания

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     int number;
6     printf("Введите число: ");
7     scanf("%i",&number);
8     if (number==0) exit(0);
9     else if(number>0) exit(1);
10    else exit(2);
11    return (3);
12 }
```

Рис. 3.4: Программа на языке Си: cod02.c

Написала 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. Если число равно нулю `exit(0)`, если больше нуля - `exit(1)`, если меньше - `exit(2)`. (рис. @fig:004)

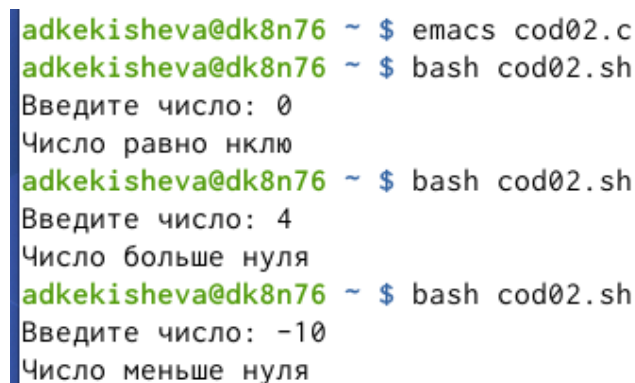


```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
gcc -o cprog cod02.c
./cprog
case $? in
  0)echo 'Число равно нулю';;
  1)echo 'Число больше нуля';;
  2)echo 'Число меньше нуля';;
esac

U:--- cod02.sh All L10 (Shell-script[bash])
```

Рис. 3.5: Командный файл, вызывающий программу cod02.c

После написала командный файл, который вызывает эту программу и, анализируя с помощью команды \$?, выдаёт сообщение о том, какое число было введено.(рис. @fig:005)

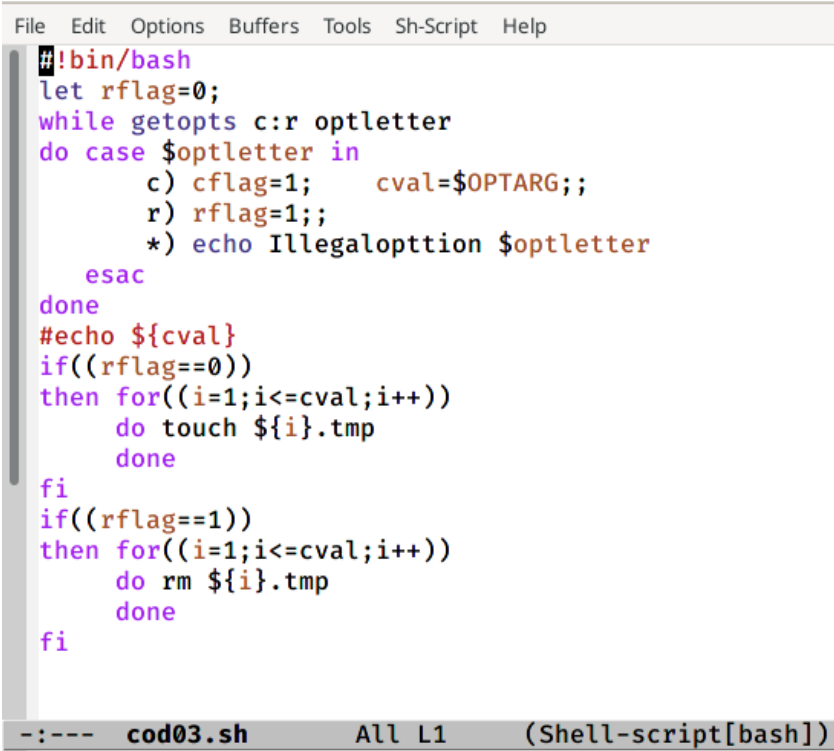


```
adkekisheva@dk8n76 ~ $ emacs cod02.c
adkekisheva@dk8n76 ~ $ bash cod02.sh
Введите число: 0
Число равно нулю
adkekisheva@dk8n76 ~ $ bash cod02.sh
Введите число: 4
Число больше нуля
adkekisheva@dk8n76 ~ $ bash cod02.sh
Введите число: -10
Число меньше нуля
```

Рис. 3.6: Результат работы программ

После, проверила выполнение командного файла, вызвав его командой bash.
(рис. @fig:006)

3.3 Выполнение 3-го пункта задания



```
#!/bin/bash
let rflag=0;
while getopts c:r optletter
do case $optletter in
    c) cflag=1;    cval=$OPTARG;;
    r) rflag=1;;
    *) echo Illegaloption $optletter
    esac
done
#echo ${cval}
if((rflag==0))
then for((i=1;i<=cval;i++))
do touch ${i}.tmp
done
fi
if((rflag==1))
then for((i=1;i<=cval;i++))
do rm ${i}.tmp
done
fi
```

--- cod03.sh All L1 (Shell-script[bash])

Рис. 3.7: Командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N

Написала 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. Для этого я воспользовалась командой `getopts` и у меня получилось два флага `c` - создание и `r` - удаление. Напротив `cflag` пишем `OPTARG`, так как ожидаются дополнительные значения, так как мы будем создавать файлы. И далее, в зависимости от того, какой файл мы применим, то есть какой будет равен 1 в результате работы `case`. Если флаг удаление равен нулю мы создаём файлы, количество которых регулируется циклом, там же задаётся расширение, а если флаг удаления равен 1, то удаляем эти файлы. (рис. @fig:007)

```

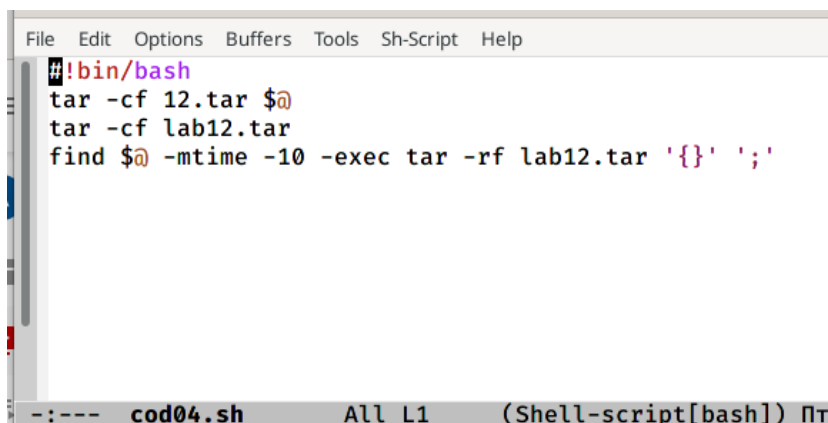
adkekisheva@dk8n76 ~ $ emacs cod03.sh
adkekisheva@dk8n76 ~ $ bash cod03.sh -c5
adkekisheva@dk8n76 ~ $ ls
1. tmp      cod03.sh      feathers      public        v
2. tmp      cod03.sh~    file.txt     public_html   Видео
3. tmp      '#cod2.sh#'  getopt.sh    script01.sh~  Документы
4. tmp      cod2.sh      getopt.sh~   script02.sh   Загрузки
5. tmp      cod2.sh~    GNUstep     script02.sh~  Изображения
cod01.sh    cod3.sh      lab11.c     script03.sh   Музыка
cod01.sh~  cod3.sh~    lab11.c~    script03.sh~  Общедоступные
cod01.txt  cod4.sh      laboratory  script04.sh   ОС
cod01.txt~ cod4.sh~    Makefile    script04.sh~  'Рабочий стол'
cod02.c    conf.txt     may         ser.sage      Шаблоны
cod02.c~   cprog       my_os       skr.sh~
cod02.sh   euler2.sage prog.ccp     text.txt
cod02.sh~  euler.sage  Programma   tmp
adkekisheva@dk8n76 ~ $ bash cod03.sh -c5 -r
adkekisheva@dk8n76 ~ $ ls
cod01.sh    cod2.sh~    getopt.sh~   script01.sh~  Видео
cod01.sh~  cod3.sh     GNUstep     script02.sh   Документы
cod01.txt  cod3.sh~    lab11.c     script02.sh~  Загрузки
cod01.txt~ cod4.sh     lab11.c~    script03.sh   Изображения
cod02.c    cod4.sh~    laboratory  script03.sh~  Музыка
cod02.c~   conf.txt   Makefile    script04.sh   Общедоступные
cod02.sh   cprog      may         script04.sh~  ОС
cod02.sh~  euler2.sage my_os       ser.sage      'Рабочий стол'
cod03.sh   euler.sage prog.ccp     skr.sh~       Шаблоны
cod03.sh~  feathers   Programma   text.txt
'#cod2.sh#' file.txt    public      tmp
cod2.sh    getopt.sh  public_html v

```

Рис. 3.8: Результат работы программы: создание и удаление

Вызвала командный файл и попробовала создать 5 файлов, используя для этого опции -c5. И после удалила их той же командой + используя опцию -r. (рис. @fig:008)

3.4 Выполнение 4-го пункта задания



```

File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
tar -cf 12.tar $@
tar -cf lab12.tar
find $@ -mtime -10 -exec tar -rf lab12.tar '{}' ';'

-:--- cod04.sh All L1 (Shell-script[bash]) Пт

```

Рис. 3.9: Командный файл, архивирует файлы в указанной директории

Написала командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. и модифицировала его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад, для этого использовала команду find. Когда я попробовала осуществить это архив создавался пустым, поэтому я решила сделать 10 дней. Во-первых я создаю архив tar -cf и для выбора директории пишу \$@. Все файлы будут архивированы в lab12.tar. После применяю команду find, где указываем директорию, затем -mtime n*24, указываю вместо n=10, далее идёт опция -exec '{} ' '; которая выполняет true, если верно условие на дни и запускает указанную команду (tar -f использовать файл или устройство АРХИВ и -r добавление файлов в конец архива) для выбранных файлов. (рис. @fig:009)

```
adkekisheva@dk8n76 ~ $ bash cod04.sh laboratory
tar: 12.tar: Функция open завершилась с ошибкой: Превышена дисковая квота
tar: Error is not recoverable: exiting now
tar: Робкий отказ от создания пустого архива
Попробуйте «tar --help» или «tar --usage» для
получения более подробного описания.
tar: lab12.tar: Функция read завершилась с ошибкой: Неправильный дескриптор файла
tar: Начало ленты, завершение работы
tar: Error is not recoverable: exiting now
```

Рис. 3.10: Вызов командного файла

Запустила командный файл, на этом моменте мне выдалось много ошибок, но в итоге архив создался. (рис. @fig:010)

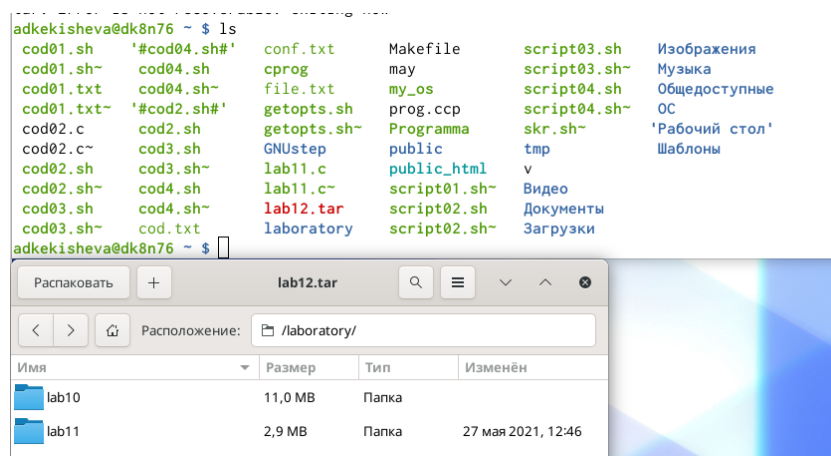


Рис. 3.11: Результат работы командного файла

Результат работы командного файла. (рис. @fig:011)

4 Вывод

Я продолжила изучать основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

5 Библиография

1. Ссылка 1

Контрольные вопросы:

1. Каково предназначение команды `getopts`?

Команда `getopts` осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]`

2. Какое отношение метасимволы имеют к генерации имён файлов?

Метасимволы используют при генерации имен, например:

- – произвольная (возможно пустая) последовательность символов;
- ? один произвольный символ;
- [...] любой из символов, указанных в скобках перечислением и/или с указанием диапазона;
- `cat f*` выдаст все файлы каталога, начинающиеся с “f”;
- `cat f` выдаст все файлы, содержащие “f”;
- `cat program.?` выдаст файлы данного каталога с однобуквенными расширениями, скажем “program.c” и “program.o”, но не выдаст “program.com”;

- `cat [a-d]*` выдаст файлы, которые начинаются с “a”, “b”, “c”, “d”. Аналогичный эффект дадут и команды “`cat [abcd]`” и “`cat [bdac]`”.

3. Какие операторы управления действиями вы знаете? <, >,

Операторы `&&` и `||` являются управляющими операторами. Если в командной строке стоит `command1 && command2`, то `command2` выполняется в том, и только в том случае, если статус выхода из команды `command1` равен нулю, что говорит об успешном ее завершении. Аналогично, если командная строка имеет вид `command1 || command2`, то команда `command2` выполняется тогда, и только тогда, когда статус выхода из команды `command1` отличен от нуля.

4. Какие операторы используются для прерывания цикла?

Для прерывания цикла используются операторы `break`, `continue`:

Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях

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

Команда `true` всегда возвращает ноль в качестве выходного статуса для индикации успеха. Команда `false` всегда возвращает не-ноль в качестве выходного статуса для индикации неудачи. Во всех управляющих конструкциях в качестве логического значения используется код возврата из программы, указанной в качестве условия. Код возврата 0 – истина, любое другое значение – ложь. Программа `true` – всегда завершается с кодом 0, `false` – всегда завершается с кодом 1.

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

Введенная строка означает условие существования файла `man s/i.$s`

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

Цикл `While` выполняется до тех пор, пока указанное в нем условие истинно. Когда указанное условие становится ложным - цикл завершается. Цикл `Until` выполняется до тех пор, пока указанное в нем условие ложно