

Отчёт по лабораторной работе 11

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

Ангелина Дмитриевна Чванова

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
4	Выполнение лабораторной работы	9
5	Выводы	16
6	Контрольные вопросы	17

Список иллюстраций

4.1	Создание файла, открытие emacs в фоновом режиме	9
4.2	Скрипт программы	10
4.3	Команда <code>chmod +x</code> , которая дает право на исполнение, работа программы	10
4.4	Создание файлов , открытие emacs в фоновом режиме. Команда <code>chmod +x</code> , которая дает право на исполнение	11
4.5	Скрипт программы на С	11
4.6	Скрипт программы	12
4.7	Работа программы	12
4.8	Создание файла , открытие emacs в фоновом режиме. Команда <code>chmod +x</code> , которая дает право на исполнение	13
4.9	Скрипт программы на С	13
4.10	Работа программы	14
4.11	Работа программы	14
4.12	Создание файла , открытие emacs в фоновом режиме. Команда <code>chmod +x</code> , которая дает право на исполнение	15
4.13	Скрипт программы на С	15
4.14	Работа программы	15

Список таблиц

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

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;
- `-р`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 `-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 ∞ (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 Теоретическое введение

Весьма необходимой при программировании является команда `getopts`, которая осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных.

Флаги — это опции командной строки, обычно помеченные знаком минус; Например, для команды `ls` флагом может являться `-F`. Иногда флаги имеют аргументы, связанные с ними. Программы интерпретируют флаги, соответствующим образом изменяя своё поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, то она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введённые данные с помощью оператора `case`.

Функция `getopts` включает две специальные переменные среды — `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента (будет равна `file_in.txt` для опции `i` и `file_out.doc` для опции `o`. `OPTIND` является числовым индексом на упомянутый аргумент. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введённых пользователем данных.

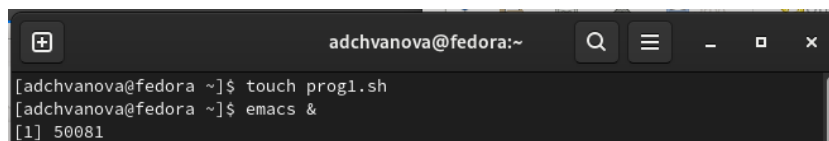
4 Выполнение лабораторной работы

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;
- `-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`.

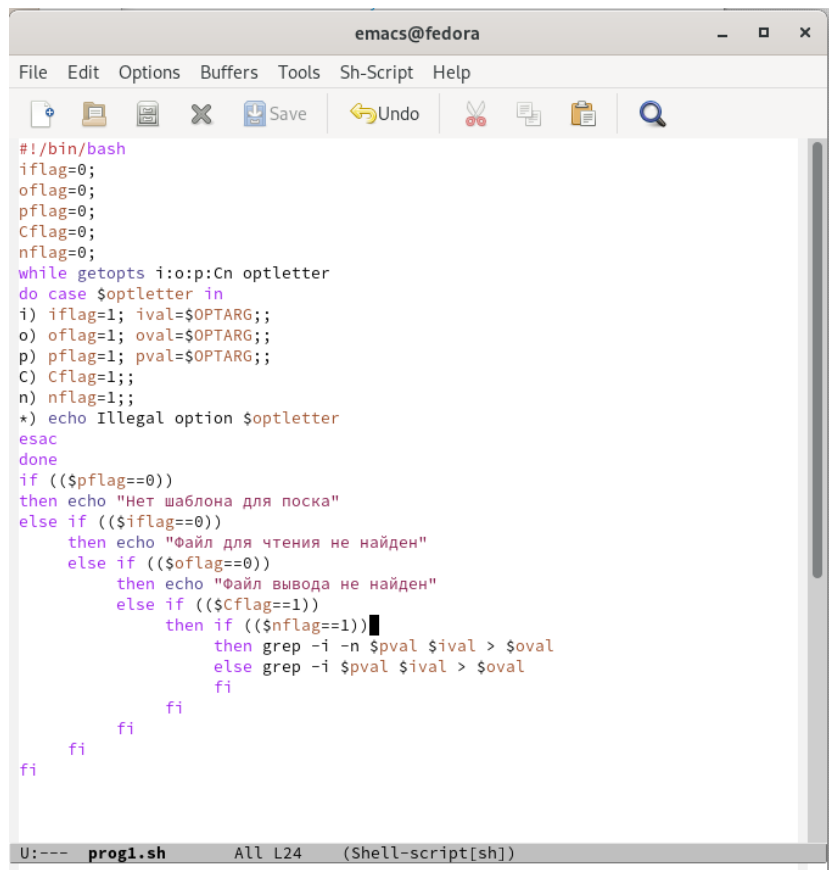
Создаем файл, а также открываем `emacs` в фоновом режиме. (рис. 4.1)



```
adchvanova@fedora:~  
[adchvanova@fedora ~]$ touch prog1.sh  
[adchvanova@fedora ~]$ emacs &  
[1] 50081
```

Рис. 4.1: Создание файла, открытие `emacs` в фоновом режиме

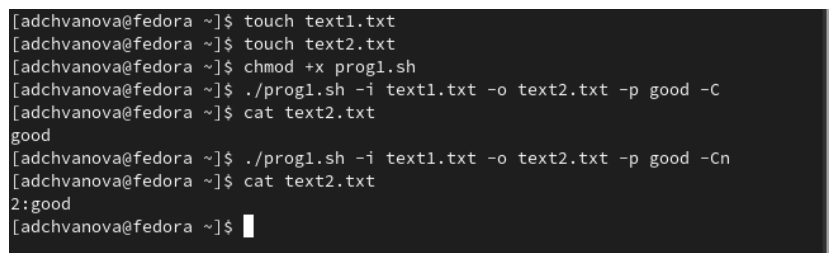
Пишем программу.(рис. 4.2)



```
#!/bin/bash
iflag=0;
oflag=0;
pflag=0;
Cflag=0;
nflag=0;
while getopts i:o:p: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 Illegal option $optletter
esac
done
if (($pflag==0))
then echo "Нет шаблона для поиска"
else if (($iflag==0))
then echo "Файл для чтения не найден"
else if (($oflag==0))
then echo "Файл вывода не найден"
else if (($Cflag==1))
then if (($nflag==1))
then grep -i -n $pval $ival > $oval
else grep -i $pval $ival > $oval
fi
fi
fi
fi
fi
fi
```

Рис. 4.2: Скрипт программы

Делаем файл исполняемым и проверяем его работу. (рис. 4.3)



```
[adchvanova@fedora ~]$ touch text1.txt
[adchvanova@fedora ~]$ touch text2.txt
[adchvanova@fedora ~]$ chmod +x prog1.sh
[adchvanova@fedora ~]$ ./prog1.sh -i text1.txt -o text2.txt -p good -C
[adchvanova@fedora ~]$ cat text2.txt
good
[adchvanova@fedora ~]$ ./prog1.sh -i text1.txt -o text2.txt -p good -Cn
[adchvanova@fedora ~]$ cat text2.txt
2:good
[adchvanova@fedora ~]$
```

Рис. 4.3: Команда `chmod +x`, которая дает право на исполнение, работа программы

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.

Создаем файлы , а также открываем emacs в фоновом режиме. Делаем их исполняемыми (рис. 4.4)

```
[adchvanova@fedora ~]$ touch n2.c
[adchvanova@fedora ~]$ emacs &
[2] 51257
[adchvanova@fedora ~]$ touch prog2.sh
[adchvanova@fedora ~]$ emacs &
[3] 51509
[adchvanova@fedora ~]$ chmod +x prog2.sh
```

Рис. 4.4: Создание файлов , открытие emacs в фоновом режиме. Команда chmod +x, которая дает право на исполнение

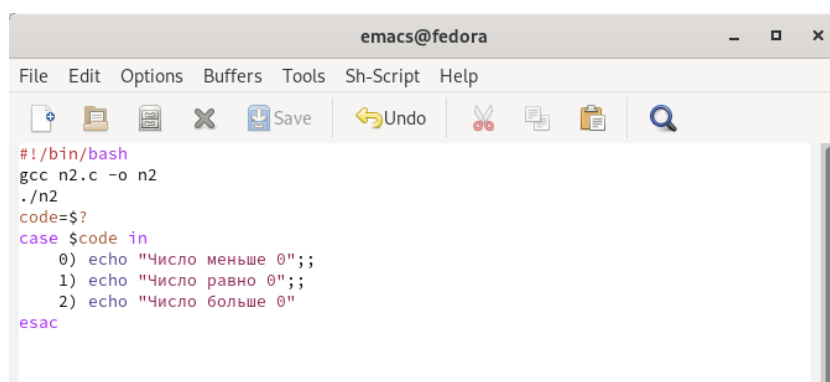
Пишем программу на языке C.(рис. 4.5)

The image shows a screenshot of the Emacs text editor window. The title bar reads 'emacs@fedora'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'C', and 'Help'. Below the menu bar is a toolbar with icons for file operations and editing. The main text area contains a C program script with the following code:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    printf("Введите число\n");
    int n;
    scanf("%d", &n);
    if(n<0) exit(0);
    if(n==0) exit(1);
    if(n>0) exit(2);
    return 0;
}
```

Рис. 4.5: Скрипт программы на C

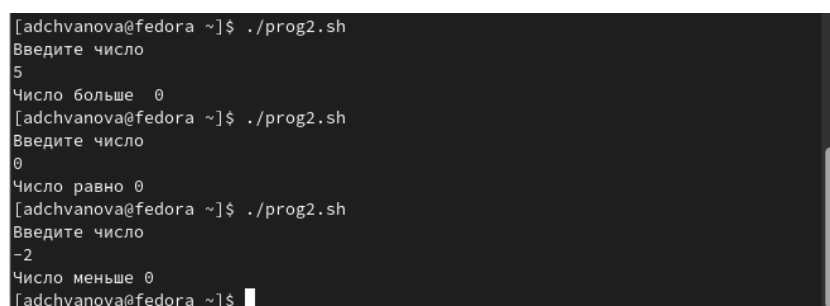
Пишем командный файл .(рис. 4.6)



```
#!/bin/bash
gcc n2.c -o n2
./n2
code=$?
case $code in
0) echo "число меньше 0";;
1) echo "число равно 0";;
2) echo "число больше 0";;
esac
```

Рис. 4.6: Скрипт программы

Работа программы. (рис. 4.7)



```
[adchvanova@fedora ~]$ ./prog2.sh
Введите число
5
Число больше 0
[adchvanova@fedora ~]$ ./prog2.sh
Введите число
0
Число равно 0
[adchvanova@fedora ~]$ ./prog2.sh
Введите число
-2
Число меньше 0
[adchvanova@fedora ~]$
```

Рис. 4.7: Работа программы

3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до ∞ (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).

Создаем файл , а также открываем emacs в фоновом режиме. Делаем их исполняемыми (рис. 4.8)

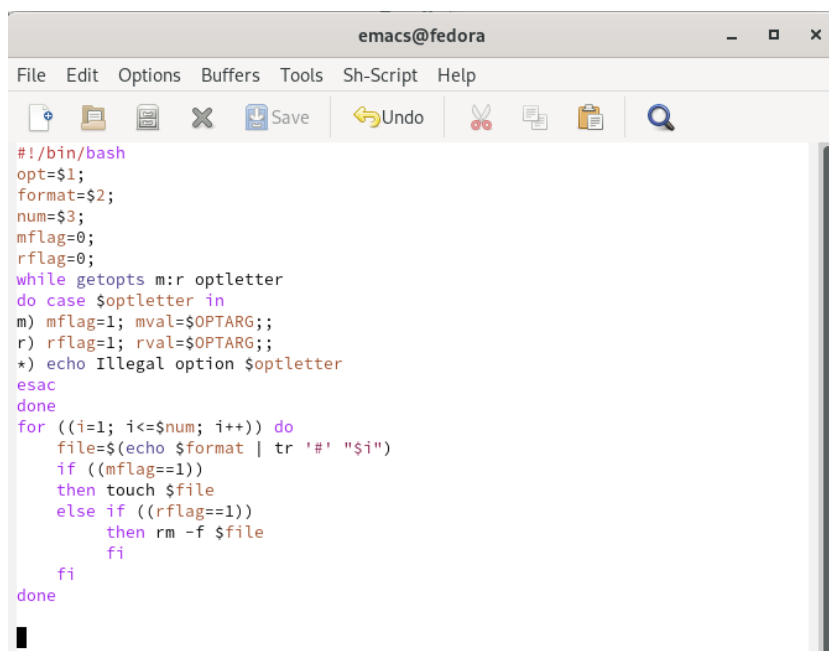
```

[adchvanova@fedora ~]$ touch prog3.sh
[2]-  Завершён      emacs
[3]+  Завершён      emacs
[adchvanova@fedora ~]$ emacs &
[2] 52545
[adchvanova@fedora ~]$ chmod +x prog3.sh

```

Рис. 4.8: Создание файла , открытие emacs в фоновом режиме. Команда `chmod +x`, которая дает право на исполнение

Пишем программу.(рис. 4.9)



```

emacs@fedora
File Edit Options Buffers Tools Sh-Script Help
+ Save Undo
#!/bin/bash
opt=$1;
format=$2;
num=$3;
mflag=0;
rflag=0;
while getopts m:r optletter
do case $optletter in
m) mflag=1; mval=$OPTARG;;
r) rflag=1; rval=$OPTARG;;
*) echo Illegal option $optletter
esac
done
for ((i=1; i<=$num; i++)) do
file=$(echo $format | tr '#' "$i")
if ((mflag==1))
then touch $file
else if ((rflag==1))
then rm -f $file
fi
fi
done

```

Рис. 4.9: Скрипт программы на С

Работа программы (рис. 4.10 - 4.11)

```
[adchvanova@fedora ~]$ ./prog3.sh -m hh#.txt 3
[adchvanova@fedora ~]$ ls
010-lab_shell_prog_1.pdf  my_os      pr.sh~
abc1                      n2         reports
australia                n2.c      ski.plases
backup                   n2.c~     snap
backup.sh~              newdir     study_2021-2022_os-intro
bin                      os         text1.txt
conf.txt                 password   text2.txt
conf.txt.               play       work
feathers                 pr10.3.sh Видео
file.txt                pr10.3.sh~ Документы
games                   prog1.sh   Загрузки
hh1.txt                 prog1.sh~  Изображения
hh2.txt                 prog2.sh   Музыка
hh3.txt                 prog2.sh~  Общедоступные
lab07.sh~              prog3.sh   'Рабочий стол'
letters                 prog3.sh~  сжатие
may                     prog4.sh~  Шаблоны
monthly                 pr.sh
```

Рис. 4.10: Работа программы

```
[adchvanova@fedora ~]$ ./prog3.sh -m hh#.txt 3
[adchvanova@fedora ~]$ ls
010-lab_shell_prog_1.pdf  my_os      pr.sh~
abc1                      n2         reports
australia                n2.c      ski.plases
backup                   n2.c~     snap
backup.sh~              newdir     study_2021-2022_os-intro
bin                      os         text1.txt
conf.txt                 password   text2.txt
conf.txt.               play       work
feathers                 pr10.3.sh Видео
file.txt                pr10.3.sh~ Документы
games                   prog1.sh   Загрузки
hh1.txt                 prog1.sh~  Изображения
hh2.txt                 prog2.sh   Музыка
hh3.txt                 prog2.sh~  Общедоступные
lab07.sh~              prog3.sh   'Рабочий стол'
letters                 prog3.sh~  сжатие
may                     prog4.sh~  Шаблоны
monthly                 pr.sh
```

Рис. 4.11: Работа программы

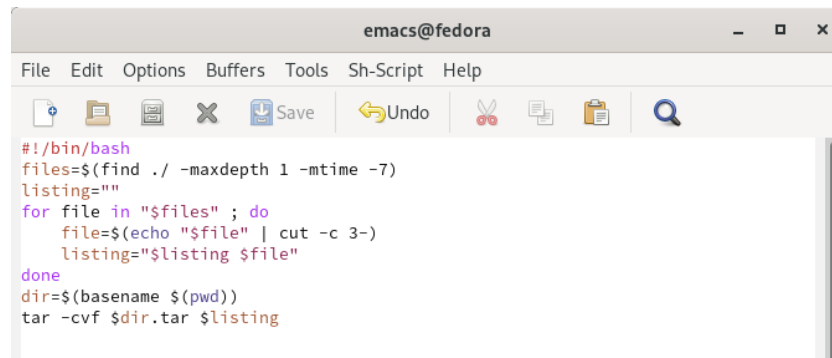
4. Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find). Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию

Создаем файл, а также открываем emacs в фоновом режиме. Делаем их исполняемыми (рис. 4.12)

```
[adchvanova@fedora ~]$ touch prog4.sh
[2]+  Завершён      emacs
[adchvanova@fedora ~]$ emacs &
[2] 53628
[adchvanova@fedora ~]$ chmod +x prog4.sh
```

Рис. 4.12: Создание файла , открытие emacs в фоновом режиме. Команда `chmod +x`, которая дает право на исполнение

Пишем программу.(рис. 4.13)



```
emacs@fedora
File Edit Options Buffers Tools Sh-Script Help
[Icons: New, Open, Save, Undo, Cut, Copy, Paste, Find]

#!/bin/bash
files=$(find ./ -maxdepth 1 -mtime -7)
listing=""
for file in "$files" ; do
    file=$(echo "$file" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

Рис. 4.13: Скрипт программы на С

Работа программы (рис. 4.14)

```
[adchvanova@fedora ~]$ ./prog4.sh
Загрузки/
Загрузки/Отчет Лабораторная работа 2.docx
Загрузки/study_2021-2022_os-intro-master.zip
Загрузки/presentation.md
Загрузки/report_lab4.md
Загрузки/28-04-2022_13-26-12.zip
Загрузки/lab3report (3).md
Загрузки/28-04-2022_13-26-12/
```

Рис. 4.14: Работа программы

5 Выводы

Мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

6 Контрольные вопросы

1).Каково предназначение команды getoptс?

Команда getoptс осуществляет синтаксический анализ командной строки, выделяя флаги, используется для объявления переменных. Синтаксис команды следующий: getoptс option-string variable [arg...] Флаги – это опции командной строки, обычно помеченные знаком минус; Например, для команды ls флагом может являться -F. Строка опций option-string – это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда getoptс может распознать аргумент, то она возвращает истину. Принято включать getoptс в цикл while и анализировать введённые данные с помощью оператора case. Функция getoptс включает две специальные переменные среды –OPTARG и OPTIND. Если ожидается дополнительное значение, то OPTARG устанавливается в значение этого аргумента. Функция getoptс также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введённых пользователем данных.

2).Какое отношение метасимволы имеют к генерации имён файлов?

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, лексикогра-

фически находящемуся между символами c1 и c2. Например, 1.1 echo – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды ls; 1.2. ls.c–выведет все файлы с последними двумя символами, совпадающими с.с. 1.3. echoproг.?–выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются prog.. 1.4.[a-z]–соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

3). Какие операторы управления действиями вы знаете?

Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути, являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.

4). Какие операторы используются для прерывания цикла

Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. 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`?

Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования `bash`: это команда `true`, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда `false`, которая всегда возвращает код завершения, неравный нулю (т.е. ложь).

6). . Что означает строка `if test -f mans/i.$$`, встречаемая в командном файле?

Строка `if test -f mans/i.s, mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).

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

Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.