

Лабораторная работа №11: отчет.

Программирование в командном процессоре ОС UNIX. Ветвления и циклы.

Евдокимов Максим Михайлович. Группа - НФИбд-01-20.

Содержание

Цель работы	4
Задание	5
Выполнение лабораторной работы	6
Задание 1	6
Код 1	7
Задание 2	8
Код 2	10
Задание 3	11
Код 3	11
Задание 4	12
Код 4	13
Контрольные вопросы	14
Выводы	17
Список литературы	18

Список иллюстраций

1	Текстовый файл откуда берётся текст	6
2	Код программы 1	7
3	Файл результат	7
4	Код на языке Си	9
5	Код программы 2	9
6	Результат работы 2	9
7	Код программы 3	11
8	Результат работы 3	11
9	Код программы 4	12
10	Полученный архив tar	13

Цель работы

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

Задание

1. Изучение правила написания циклов.
2. Изучить правила написание ветвление.

Выполнение лабораторной работы

Задание 1

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`.

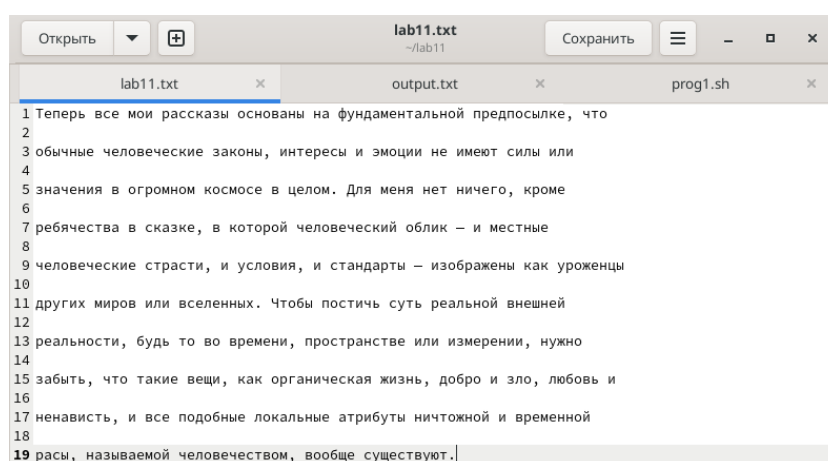
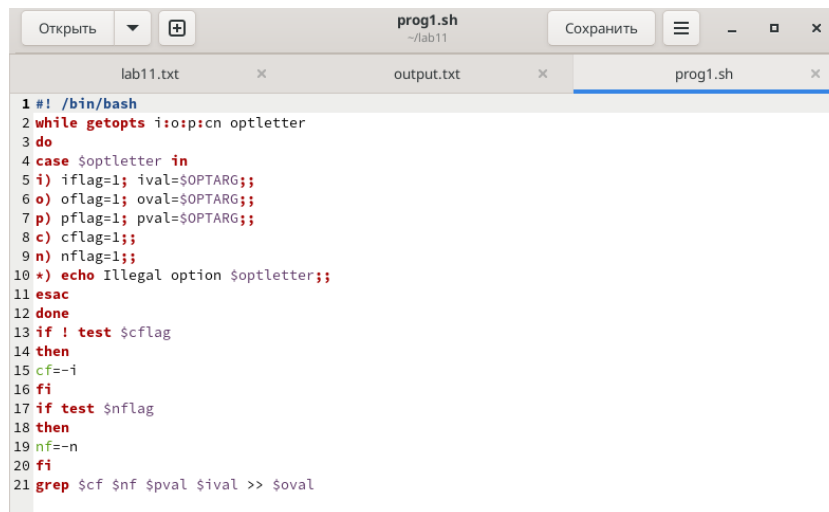


Рис. 1: Текстовый файл откуда берётся текст



```
1 #! /bin/bash
2 while getopts i:o:p:cn optletter
3 do
4 case $optletter in
5 i) iflag=1; ival=$OPTARG;;
6 o) oflag=1; oval=$OPTARG;;
7 p) pflag=1; pval=$OPTARG;;
8 c) cflag=1;;
9 n) nflag=1;;
10 *) echo Illegal option $optletter;;
11 esac
12 done
13 if ! test $cflag
14 then
15 cf=-i
16 fi
17 if test $nflag
18 then
19 nf=-n
20 fi
21 grep $cf $nf $pval $ival >> $oval
```

Рис. 2: Код программы 1

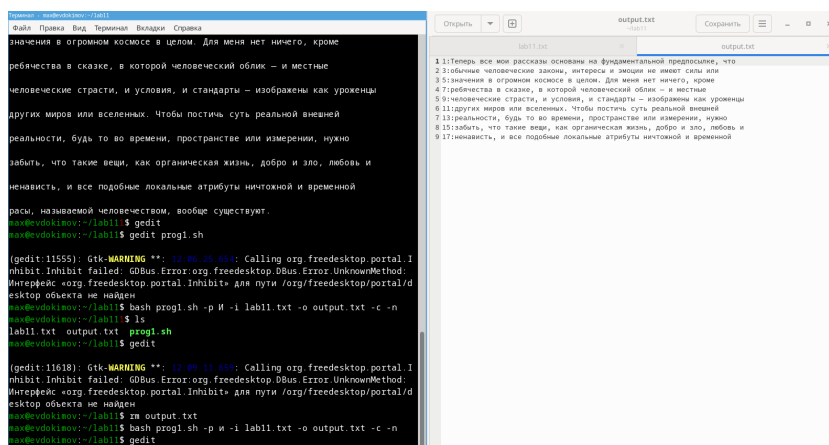


Рис. 3: Файл результат

Код 1

```
#!/bin/bash
while getopts i:o:p:cn optletter
do
case $optletter in
i) iflag=1; ival=$OPTARG;;
```

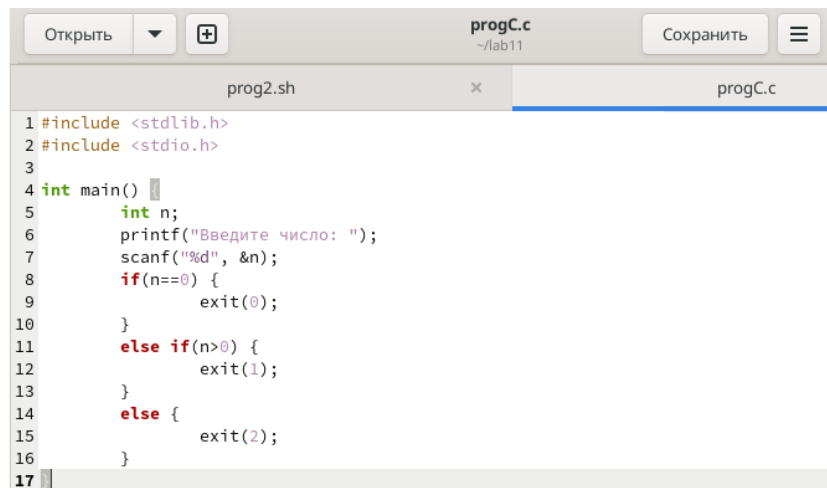
```

o) oflag=1; oval=$OPTARG;;
p) pflag=1; pval=$OPTARG;;
c) cflag=1;;
n) nflag=1;;
*) echo Illegal option $optletter;;
esac
done
if ! test $cflag
then
cf=-i
fi
if test $nflag
then
nf=-n
fi
grep $cf $nf $pval $ival >> $oval

```

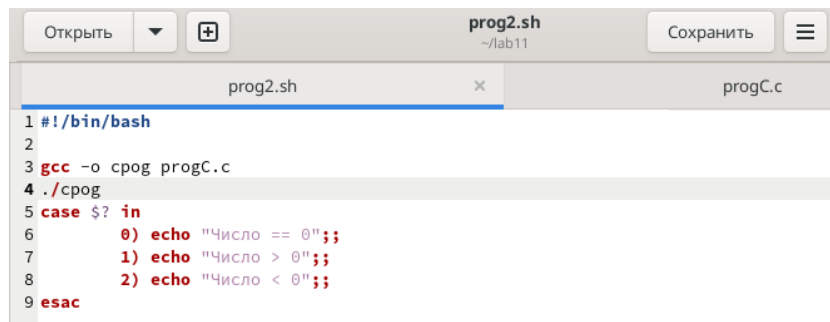
Задание 2

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.



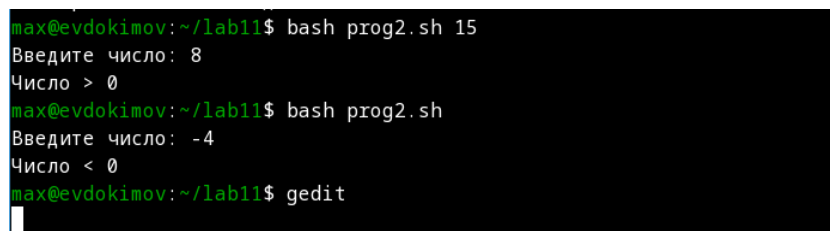
```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int main() {
5     int n;
6     printf("Введите число: ");
7     scanf("%d", &n);
8     if(n==0) {
9         exit(0);
10    }
11    else if(n>0) {
12        exit(1);
13    }
14    else {
15        exit(2);
16    }
17 }
```

Рис. 4: Код на языке Си



```
1 #!/bin/bash
2
3 gcc -o cpog progC.c
4 ./cpog
5 case $? in
6     0) echo "Число == 0";;
7     1) echo "Число > 0";;
8     2) echo "Число < 0";;
9 esac
```

Рис. 5: Код программы 2



```
max@evdokimov:~/lab11$ bash prog2.sh 15
Введите число: 8
Число > 0
max@evdokimov:~/lab11$ bash prog2.sh
Введите число: -4
Число < 0
max@evdokimov:~/lab11$ gedit
```

Рис. 6: Результат работы 2

Код 2

```
gcc -o cprog progC.c
./cprog
case $? in
    0) echo "Число == 0";;
    1) echo "Число > 0";;
    2) echo "Число < 0";;
esac

#include <stdlib.h>
#include <stdio.h>

int main() {
    int n;
    printf("Введите число: ");
    scanf("%d", &n);
    if(n==0) {
        exit(0);
    }
    else if(n>0) {
        exit(1);
    }
    else {
        exit(2);
    }
}
```

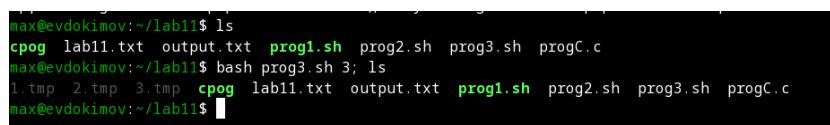
Задание 3

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).



```
1 #!/bin/bash
2 for((i=1; i<=$*; i++))
3 do
4 if test -f "$i".tmp
5 then rm "$i".tmp
6 else touch "$i.tmp"
7 fi
8 done
```

Рис. 7: Код программы 3



```
max@evdokimov:~/lab11$ ls
cpog lab11.txt output.txt prog1.sh prog2.sh prog3.sh progC.c
max@evdokimov:~/lab11$ bash prog3.sh 3; ls
1.tmp 2.tmp 3.tmp cpog lab11.txt output.txt prog1.sh prog2.sh prog3.sh progC.c
max@evdokimov:~/lab11$
```

Рис. 8: Результат работы 3

Код 3

```
#!/bin/bash
for((i=1; i<=$*; i++))
do
if test -f "$i".tmp
then rm "$i".tmp
else touch "$i.tmp"
```

fi

done

Задание 4

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`).



```
1 #!/bin/bash
2 find $* -mtime -7 -mtime +0 -type f > FILES.txt
3 tar -cf archive.tar -T FILES.txt
```

Рис. 9: Код программы 4

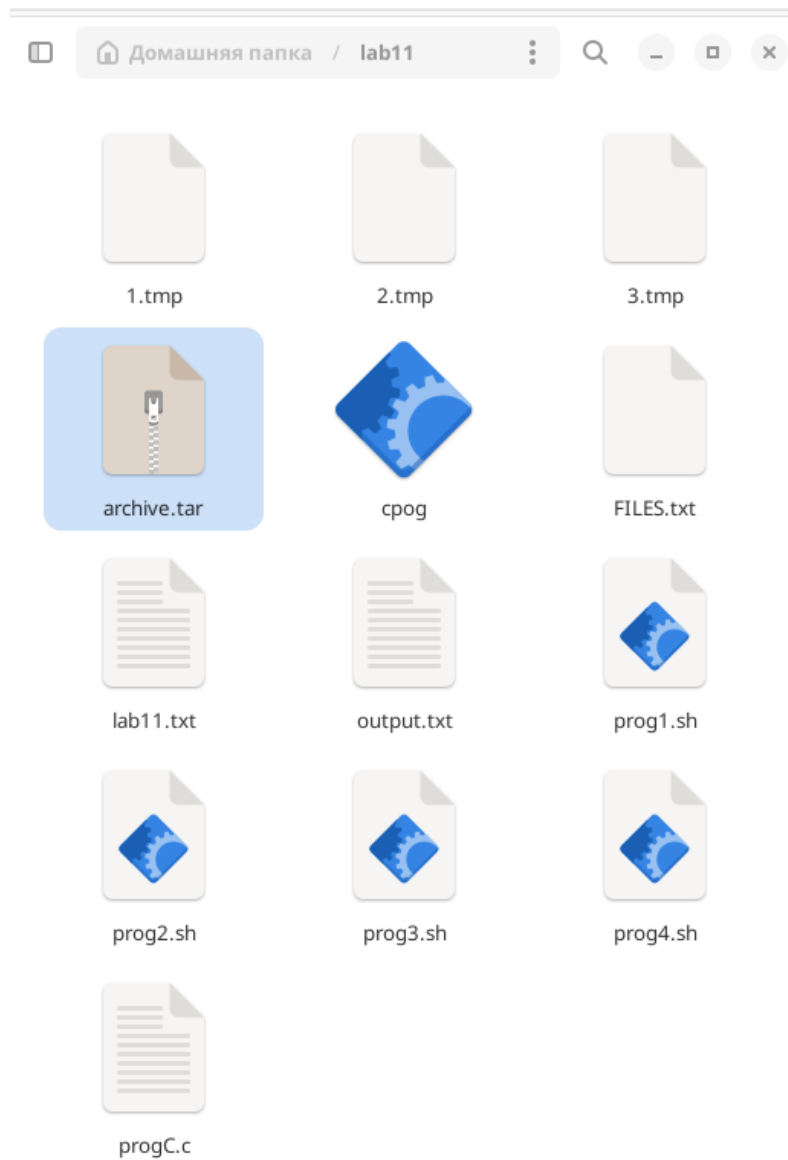


Рис. 10: Полученный архив tar

Код 4

```
#!/bin/bash
find $* -mtime -7 -mtime +0 -type f > FILES.txt
tar -cf archive.tar -T FILES.txt
```

Контрольные вопросы

1. Каково предназначение команды `getopts`?

Осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]` Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие.

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, 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.

3. Какие операторы управления действиями вы знаете?

Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости отрезультатов 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.

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, `== 0` (истина), и команда `false`, которая всегда возвращает код завершения, `!= 0` (ложь).

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

Строка “if test -f mans/i.\$s” проверяет является ли этот “объект” обычным файлом, если же 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 возвращает нулевой код завершения цикл (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 сделавшей всей цепочке команд 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.

Выводы

В ходе выполнения лабораторной работы были изучены способы создания ветвления и циклов и работа с си файлами.

Список литературы

1. Лабораторная работа №11