

# Отчёт по лабораторной работе №11

Программирование в командном процессоре ОС UNIX. Ветвления и  
циклы

Ганина Таисия Сергеевна, НКАбд-01-22

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Листинги	15
6	Контрольные вопросы	17
7	Выводы	20
	Список литературы	21

## Список иллюстраций

4.1	Файл . . . . .	8
4.2	Текст программы 1 . . . . .	9
4.3	Файл, в котором выполнялась программа . . . . .	10
4.4	Результат . . . . .	10
4.5	Программа на Си . . . . .	11
4.6	Программный файл . . . . .	11
4.7	Результат . . . . .	12
4.8	Программный файл . . . . .	12
4.9	Результат . . . . .	13
4.10	Файл и запуск . . . . .	13
4.11	Текст программы 4 . . . . .	14
4.12	Созданный архив и файл . . . . .	14
4.13	FILES.txt . . . . .	14

## 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; • `-rшаблон` — 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 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 Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: - оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;

- С-оболочка (или csh) — надстройка на оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

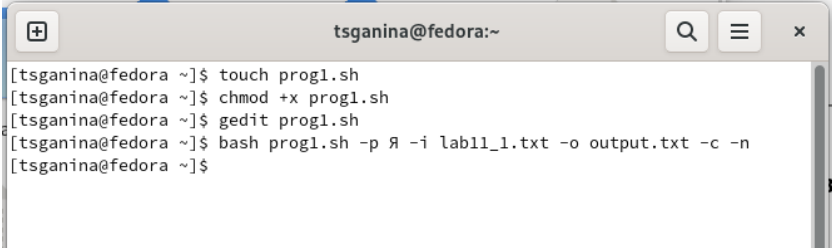
POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода.

## 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;
- `-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. 4.1, 4.2, 4.3, 4.4)



```
tsganina@fedora:~  
[tsganina@fedora ~]$ touch prog1.sh  
[tsganina@fedora ~]$ chmod +x prog1.sh  
[tsganina@fedora ~]$ gedit prog1.sh  
[tsganina@fedora ~]$ bash prog1.sh -p Я -i lab11_1.txt -o output.txt -c -n  
[tsganina@fedora ~]$
```

Рис. 4.1: Файл



```
1  #! /bin/bash
2  while getopts 1:o:p:cn optletter
3  do
4  case $optletter in
5      1) iflag=1; ival=$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=-1
16 fi
17 if test $nflag
18 then
19     nf=-n
20 fi
21 grep $cf $nf $pval $ival >> $oval
```

Рис. 4.2: Текст программы 1

```

1 В динамиках лишь звенящая тишина,
2
3 Шелест волн и, бог знает, какой прибор...
4
5 Эта история больше совсем никому не нужна:
6
7 Лишь я остаюсь в постоянном контакте с тобой.
8
9
10 Я упрямо слушаю этот извечный шум,
11
12 Сквозь помехи ловя одной мне приоткрытый ритм.
13
14 Голова ломается от извечных тяжелых дум,
15
16 Хочу на дно уйти, добраться до тех глубин.
17
18
19 52 Герца, странно и чуждо всем.

```

Рис. 4.3: Файл, в котором выполнялась программа

```

1 6:Я упрямо слушаю этот извечный шум,
2 13:Я тоню в океане своих и чужих проблем,
3 21:Я рисую графики, крепче держу гидрофон.
4 22:Я готова кричать: я тоже здесь, приходи.
5 29:Я надеюсь, что ты вернешься ко мне невредим.
6 33:Я поймала сигнал: нервы тянут не хуже каната.
7 6:Я упрямо слушаю этот извечный шум,
8 13:Я тоню в океане своих и чужих проблем,
9 21:Я рисую графики, крепче держу гидрофон.
10 22:Я готова кричать: я тоже здесь, приходи.
11 29:Я надеюсь, что ты вернешься ко мне невредим.
12 33:Я поймала сигнал: нервы тянут не хуже каната.

```

Рис. 4.4: Результат

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 (рис. 4.5, 4.6, 4.7).

```

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(1);
10     }
11     else if (n==0) {
12         exit(0);
13     }
14     else {
15         exit(2);
16     }
17 }

```

Рис. 4.5: Программа на Си

```

1  #! /bin/bash
2
3  gcc -o cprog lab11_2.c
4  ./cprog
5  case $? in
6      0) echo "Число равно нулю";;
7      1) echo "Число больше нуля";;
8      2) echo "Число меньше нуля";;
9  esac

```

Рис. 4.6: Программный файл

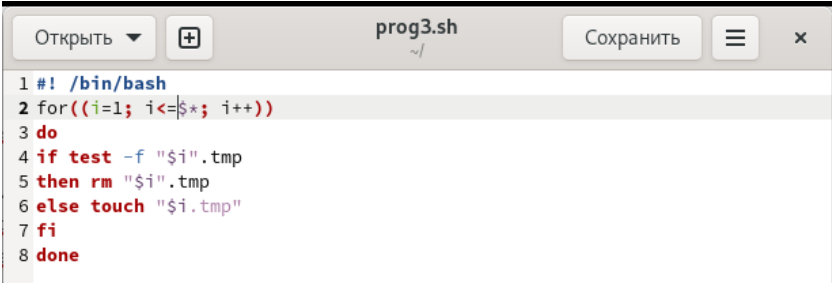
```

[tsganina@fedora ~]$ gedit prog2.sh
[tsganina@fedora ~]$ gedit lab11_2.c
[tsganina@fedora ~]$ bash prog2.sh 12
Введите число: 12
Число больше нуля
[tsganina@fedora ~]$ bash prog2.sh
Введите число: 0
Число равно нулю
[tsganina@fedora ~]$ bash prog2.sh
Введите число: -123
Число меньше нуля
[tsganina@fedora ~]$

```

Рис. 4.7: Результат

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) (рис. 4.8, 4.9).



```

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. 4.8: Программный файл

```

[tsганина@fedora ~]$ bash prog3.sh 5
[tsганина@fedora ~]$ ls
1.tmp      conf.txt    monthly     ski.plases    КомпАл
2.tmp      cprog      my_os       tmp05_lab     лаб10_файлы
3.tmp      feathers   output.txt  work          Музыка
4.tmp      file.txt   play        'Без названия.ipynb'  Общедоступные
5.tmp      lab07.sh   prog1.sh    'Библиотека calibre' 'Рабочий стол'
abc1      lab07.sh~  prog2.sh    Видео         Шаблоны
australia lab11_1.txt prog3.sh     Документы
backup    lab11_2.c  -r          Загрузки
bin       may        reports     Изображения

[tsганина@fedora ~]$ bash prog3.sh 5
[tsганина@fedora ~]$ ls
abc1      lab07.sh   play        work          лаб10_файлы
australia lab07.sh~  prog1.sh    'Без названия.ipynb'  Музыка
backup    lab11_1.txt prog2.sh    'Библиотека calibre'  Общедоступные
bin       lab11_2.c  prog3.sh    Видео         'Рабочий стол'
conf.txt   may        -r          Документы     Шаблоны
cprog      monthly   reports     Загрузки
feathers   my_os     ski.plases  Изображения
file.txt   output.txt tmp05_lab    КомпАл

[tsганина@fedora ~]$

```

Рис. 4.9: Результат

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) (рис. 4.10, 4.11, 4.12, 4.13).

```

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

```

Рис. 4.10: Файл и запуск

```

[tsганина@fedora ~]$ touch prog4.sh
[tsганина@fedora ~]$ gedit prog4.sh
[tsганина@fedora ~]$ pwd
/home/tsганина
[tsганина@fedora ~]$ ^C
[tsганина@fedora ~]$ bash prog4.sh /home/tsганина
find: '/home/tsганина/monthly': Отказано в доступе
tar: Удаляется начальный '/' из имен объектов
tar: Удаляются начальные '/' из целей жестких ссылок
[tsганина@fedora ~]$ gedit prog4.sh

```

Рис. 4.11: Текст программы 4

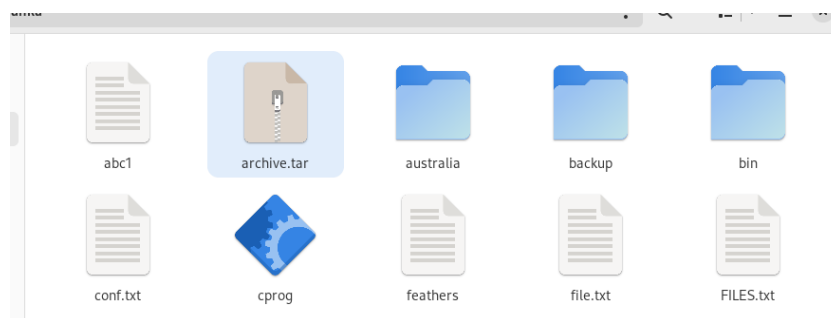


Рис. 4.12: Созданный архив и файл

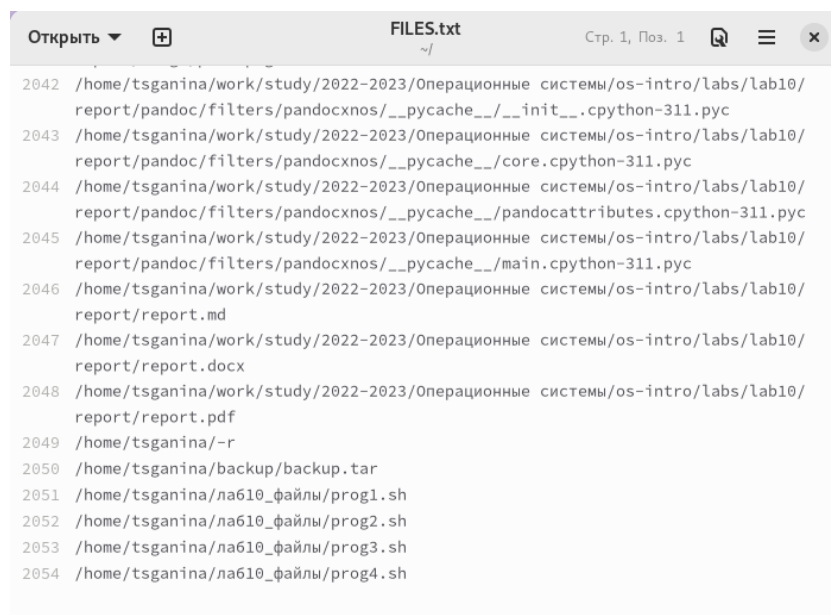


Рис. 4.13: FILES.txt

## 5 ЛИСТИНГИ

### 1. Программа 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
```

## 2. Программа 2

```
#!/bin/bash

gcc -o cprog lab11_2.c
./cprog
case $? in
0) echo "Число равно нулю";;
1) echo "Число больше нуля";;
2) echo "Число меньше нуля";;
esac
```

## 3. Программа 3

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

## 4. Программа 4

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



## 6 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. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`.
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. 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, не равный нулю (т. е. ложь).

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

## 7 Выводы

В процессе выполнения данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# Список литературы

Руководство к лабораторной работе