

# Лабораторная работа №11. Программирование в командном процессоре ОС UNIX. Ветвления и циклы

---

Подготовил:

Королев Адам Маратович

Группа: НПИбд-02-21

Студенческий билет: № 1032217060

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

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 оболочке 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 C 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.

POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке `bash`. В других оболочках большинство команд будет совпадать с описанными ниже.

Выполнение лабораторной работы:

---

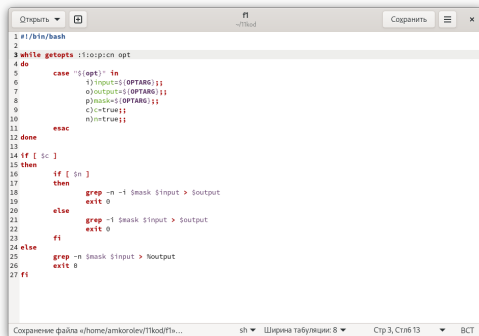
1. Используя команды `getopts` `grep`,  
написать командный файл, который  
анализирует командную строку с  
ключами:

---



– -iinputfile — прочитать данные из указанного файла; – -ooutputfile — вывести данные в указанный файл; – -ршаблон — указать шаблон для поиска; – -С — различать большие и малые буквы; – -п — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом -р.

---



```
1 #!/bin/bash
2
3 while getopts :ioip:cn opt
4 do
5     case "$opt" in
6         i)input=${OPTARG};;
7         o)output=${OPTARG};;
8         p)mask=${OPTARG};;
9         c)c=true;;
10        n)n=true;;
11    esac
12 done
13
14 if [ $c ]
15 then
16     if [ $n ]
17     then
18         grep -n -i $mask $input > $output
19         exit 0
20     else
21         grep -i $mask $input > $output
22         exit 0
23     fi
24 else
25     grep -n $mask $input > $output
26     exit 0
27 fi
```

Сохранение файла «/home/amikorelev/tikod/t1»... sh Шрифт таблицы: 8 Стр 3, Стб 13 ВСТ

Figure 1: Пишем скрипт

# Выполняем скрипт

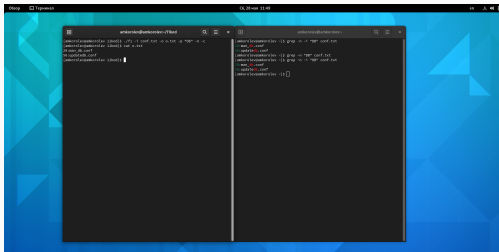


Figure 2: Выполняем скрипт

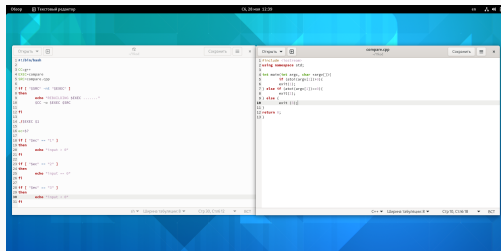
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку.

---

Командный файл должен вызывать эту программу и, проанализировав с помощью команды \$?, выдать сообщение о том, какое число было введено.

---

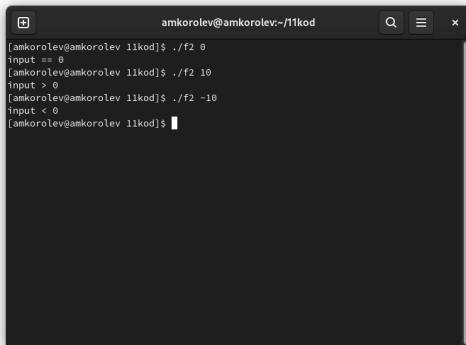
# Пишем скрипт.



```
main.cpp
1 #include <iostream>
2 #include <QApplication>
3 #include <mainwindow.h>
4
5 int main(int argc, char *argv[])
6 {
7     QApplication app(argc, argv);
8     MainWindow w;
9     w.show();
10    return app.exec();
11 }
```

```
mainwindow.cpp
1 #include <mainwindow.h>
2 #include <QPainter>
3
4 MainWindow::MainWindow(QWidget *parent)
5 : QMainWindow(parent)
6 {
7     setWindowTitle(tr("MainWindow"));
8 }
9
10 void MainWindow::paintEvent(QPaintEvent *)
11 {
12     QPainter painter(this);
13     painter.fillRect(rect(), Qt::red);
14 }
```

Figure 3: Пишем скрипт

A terminal window with a dark background and light text. The title bar at the top reads 'amkorolev@amkorolev:~/11kod'. The terminal content shows a series of commands and their outputs. The prompt is '[amkorolev@amkorolev 11kod]\$'. The commands and outputs are: './f2 0' followed by 'input == 0', './f2 10' followed by 'input > 0', and './f2 -10' followed by 'input < 0'. The final prompt is '[amkorolev@amkorolev 11kod]\$' with a cursor. The window has standard macOS-style window controls (red, yellow, green buttons) and a search icon on the right.

```
amkorolev@amkorolev:~/11kod
[amkorolev@amkorolev 11kod]$ ./f2 0
input == 0
[amkorolev@amkorolev 11kod]$ ./f2 10
input > 0
[amkorolev@amkorolev 11kod]$ ./f2 -10
input < 0
[amkorolev@amkorolev 11kod]$
```

Figure 4: Выполняем скрипт

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.).

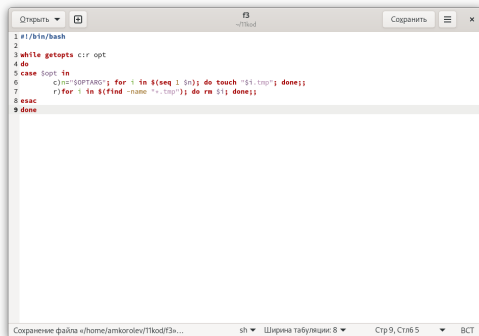
---



Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).

---

# Пишем скрипт.

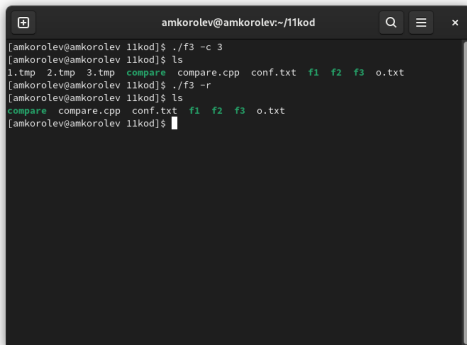


The image shows a screenshot of a text editor window. The title bar at the top contains the text "Открыть" (Open), a file icon, the filename "~Ttkod", and buttons for "Сохранить" (Save), a menu icon, and a close button. The editor area contains a shell script with the following lines:

```
1 #!/bin/bash
2
3 while getopts c:r opt
4 do
5 case $opt in
6     c) nc="$OPTARG"; for i in $(seq 1 $n); do touch "$i. tmp"; done;;
7     r) for i in $(find -name "$*.tmp"); do rm $i; done;;
8 esac
9 done
```

The status bar at the bottom of the window displays the file path "Сохранение файла «/home/amkorolev/Ttkod/T3a...», the shell "sh", the tab width "Ширина табуляции: 8", and the current position "Стр 9, Стлб 5" along with a "ВСТ" button.

Figure 5: Пишем скрипт

A terminal window with a dark background and light text. The title bar shows 'amkorolev@amkorolev:~/11kod'. The terminal content shows a series of commands and their outputs. The command './f3 -c 3' is executed, followed by 'ls', which lists files including 'compare', 'compare.cpp', 'conf.txt', 'f1', 'f2', 'f3', and 'o.txt'. Then './f3 -r' is executed, followed by another 'ls' command, which shows the same files. The prompt returns to the shell.

```
amkorolev@amkorolev:~/11kod
[amkorolev@amkorolev 11kod]$ ./f3 -c 3
[amkorolev@amkorolev 11kod]$ ls
1.tmp 2.tmp 3.tmp compare compare.cpp conf.txt f1 f2 f3 o.txt
[amkorolev@amkorolev 11kod]$ ./f3 -r
[amkorolev@amkorolev 11kod]$ ls
compare compare.cpp conf.txt f1 f2 f3 o.txt
[amkorolev@amkorolev 11kod]$
```

Figure 6: Выполняем скрипт

4. Написать командный файл,  
который с помощью команды tar  
запаковывает в архив

---

#### 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).

# Пишем скрипт.



```
1 #!/bin/bash
2
3 while getopts :d: opt; do
4     case $opt in
5         d) dir="$OPTARG";;
6     esac
7 done
8
9 find $dir -mtime -7 -mtime +0 -type f > fortar.txt
10
11 tar -cvf archive.tar -T fortar.txt
```

The screenshot shows a terminal window titled 'f4' with a path '~Ttkod'. The script contains a bash shebang, a while loop to process command-line options (looking for a directory 'd'), a find command to locate files in that directory modified within the last 7 days, and a tar command to create an archive from the found files. The status bar at the bottom indicates the file path, shell type (sh), tab width (8), and page information (11/35).

Figure 7: Пишем скрипт

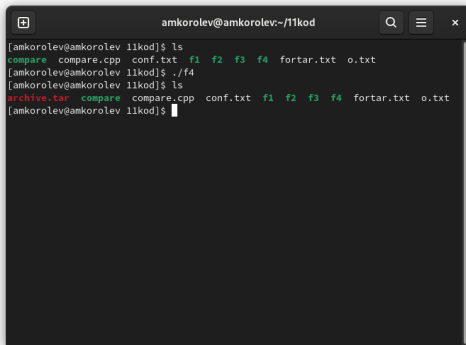
A terminal window with a dark background and light text. The title bar shows 'amkorolev@amkorolev:~/11kod'. The terminal content shows a series of commands and their outputs. The first command is 'ls', which lists files: 'compare', 'compare.cpp', 'conf.txt', 'f1', 'f2', 'f3', 'f4', 'fortar.txt', and 'o.txt'. The second command is './f4'. The third command is 'ls', which lists files: 'archive.tar', 'compare', 'compare.cpp', 'conf.txt', 'f1', 'f2', 'f3', 'f4', 'fortar.txt', and 'o.txt'. The prompt is '[amkorolev@amkorolev 11kod]\$'.

Figure 8: Выполняем скрипт

**Выводы:**

---



- В процессе выполнения работы научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.