

Лабораторная работа № 11

Ариоке Габриэль Одафе

18 Апрель 2023

Российский университет дружбы народов, Москва, Россия

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

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`.

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).

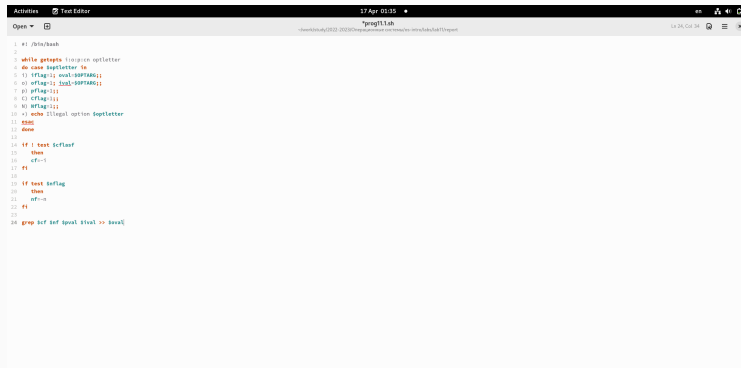
Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: - оболочка Борна (Bourne shell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

- C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку C, 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.

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, 2, 3, 4)

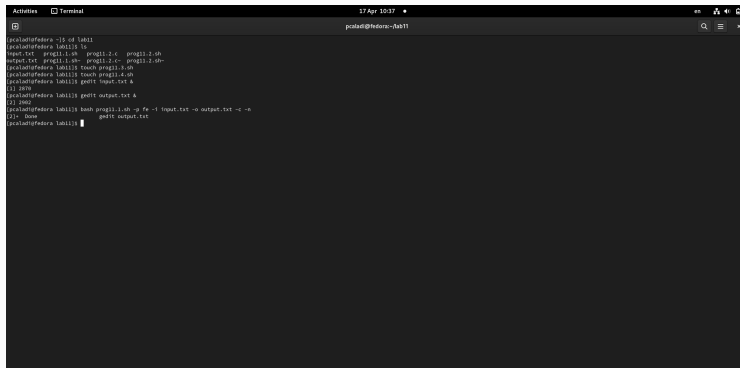
Выполнение лабораторной работы



```
1 #! /bin/bash
2
3 while getopts :roaighn optletter
4 do case $optletter in
5   r) sflag=1; eval-${OPTARG};;
6   o) oflag=1; eval-${OPTARG};;
7   p) pflag=1;;
8   c) cflag=1;;
9   h) hflag=1;;
10  *) echo "illegal option $optletter"
11     exit
12  esac
13
14 if ! test $cflag
15 then
16   cfc=5
17 fi
18
19 if test $sflag
20 then
21   ofc=a
22 fi
23
24 grep $cf $of $pval $ival >> $aval
```

Figure 1: Первая программа

Выполнение лабораторной работы



```
pcaladi@fedora ~$ cd lab11
pcaladi@fedora lab11$ ls
input.txt  prog11.1.sh  prog11.2.c  prog11.2.sh
output.txt prog11.1.sh~ prog11.2.c~  prog11.2.sh~
pcaladi@fedora lab11$ touch prog11.3.sh
pcaladi@fedora lab11$ touch prog11.4.sh
pcaladi@fedora lab11$ gedit input.txt &
[1] 2670
pcaladi@fedora lab11$ gedit output.txt &
[2] 2662
pcaladi@fedora lab11$ bash prog11.1.sh -p fe -i input.txt -o output.txt -c -n
[2]+  Done                  gedit output.txt
pcaladi@fedora lab11$
```

Figure 2: Вызов программы в терминале

Выполнение лабораторной работы

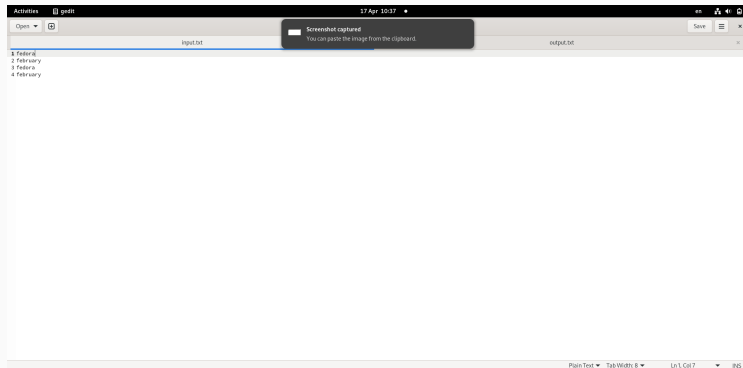


Figure 3: Результат

Выполнение лабораторной работы

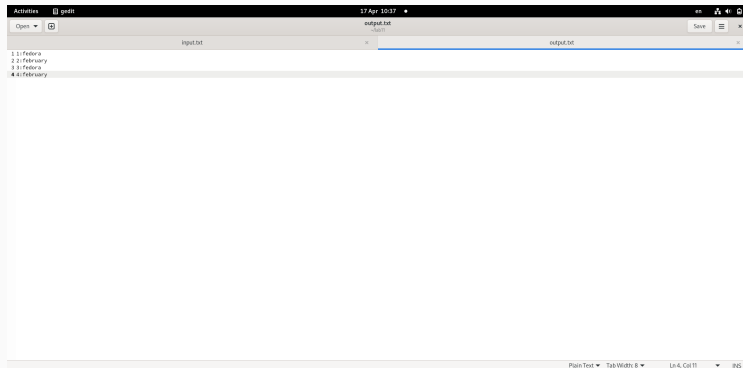
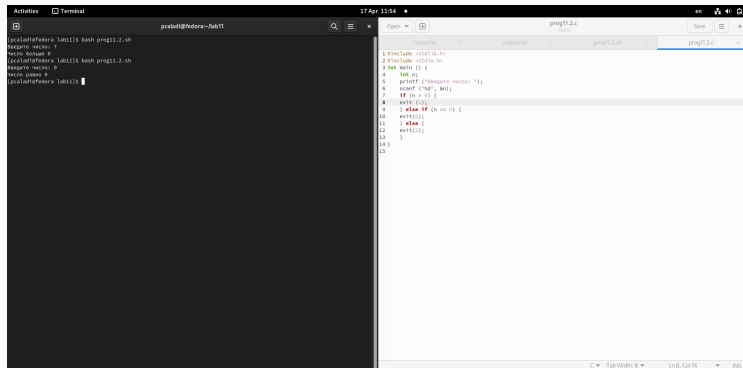


Figure 4: Результат

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. 5, 6, 7)

Выполнение лабораторной работы

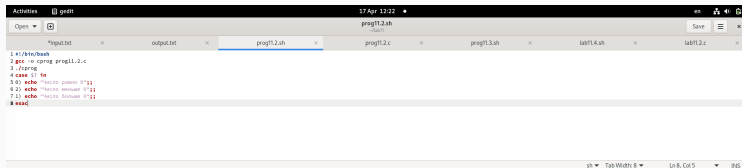


The screenshot displays a Linux desktop environment. On the left, a terminal window titled 'Terminal' shows the user 'pcabadi@fedora ~/lab11' running a program named 'prog11.2.sh'. The program prompts for 'Введите значение: 7', 'Введите значение: 0', and 'Введите значение: 8', with the user inputting '7', '0', and '8' respectively. On the right, a code editor window titled 'prog11.2.c' shows the source code of the program. The code includes standard headers, defines a 'main' function, and uses a loop to read and process input values. The code is as follows:

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

Figure 5: Вторая программа

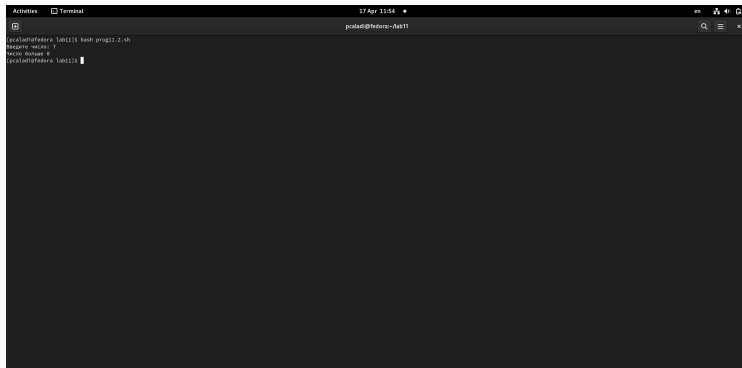
Выполнение лабораторной работы



```
1 #!/bin/bash
2 gcc -o cprog prog11.2.c
3 ./cprog
4 case $? in
5 0) echo "маленько больше 0" ;;
6 2) echo "маленько меньше 0" ;;
7 1) echo "маленько больше 0" ;;
8 esac
```

Figure 6: Вторая программа

Выполнение лабораторной работы

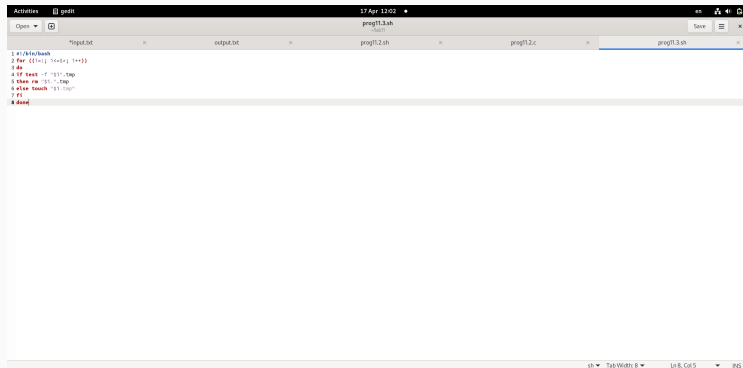
A terminal window titled 'Terminal' with a date and time of '17 Apr 11:54'. The window shows a series of commands and their outputs in Russian. The first command is 'bash prog11.2.sh', which outputs 'Результат: success! 1'. The second command is 'echo success! 0', which outputs 'success! 0'. The prompt is '[pcal@fedora ~]\$' followed by a cursor.

```
Activities Terminal 17 Apr 11:54 pcal@fedora: ~ - lab11
[pcal@fedora ~]$ bash prog11.2.sh
Результат: success! 1
success! 0
[pcal@fedora ~]$
```

Figure 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. 8, 9)

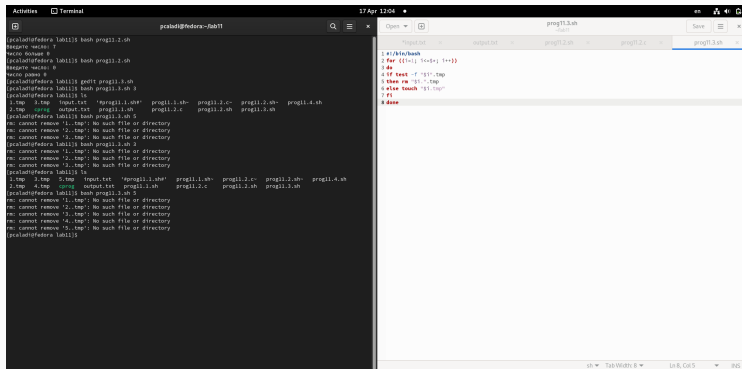
Выполнение лабораторной работы



```
1 #!/bin/bash
2 for ((i=0; i<${#} i++))
3 do
4 if test -f "11111"
5 then rm "11111"
6 else touch "11111"
7 fi
8 done
```

Figure 8: Третья программа

Выполнение лабораторной работы



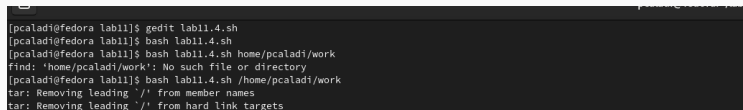
```
pcaladi@fedora:~$ bash prog11.3.sh
$egate -case: 1
$echo output 0
[pcaladi@fedora ~]$ bash prog11.3.sh
$egate -case: 0
$echo pause 0
[pcaladi@fedora ~]$ gedit prog11.3.sh
[pcaladi@fedora ~]$ bash prog11.3.sh 3
[pcaladi@fedora ~]$ ls
3.tmp 3.tmp  input.txt  'prog11.1.sh'  prog11.1.sh-  prog11.2.c-  prog11.2.sh-  prog11.4.sh
2.tmp 4.tmp  output.txt  prog11.1.sh  prog11.2.c  prog11.2.sh  prog11.3.sh
[pcaladi@fedora ~]$ bash prog11.3.sh 5
rm: cannot remove '1..tmp': No such file or directory
rm: cannot remove '2..tmp': No such file or directory
rm: cannot remove '3..tmp': No such file or directory
[pcaladi@fedora ~]$ bash prog11.3.sh 5
rm: cannot remove '1..tmp': No such file or directory
rm: cannot remove '2..tmp': No such file or directory
rm: cannot remove '3..tmp': No such file or directory
[pcaladi@fedora ~]$ ls
3.tmp 3.tmp 5.tmp  input.txt  'prog11.1.sh'  prog11.1.sh-  prog11.2.c-  prog11.2.sh-  prog11.4.sh
2.tmp 4.tmp 5.tmp  output.txt  prog11.1.sh  prog11.2.c  prog11.2.sh  prog11.3.sh
[pcaladi@fedora ~]$ bash prog11.3.sh 5
rm: cannot remove '1..tmp': No such file or directory
rm: cannot remove '2..tmp': No such file or directory
rm: cannot remove '3..tmp': No such file or directory
rm: cannot remove '4..tmp': No such file or directory
rm: cannot remove '5..tmp': No such file or directory
[pcaladi@fedora ~]$
```

Figure 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. 10, 11, 12)

```
Activities gedit 17 Apr 12:24
lab11.4.sh
~lab11
Save
*input.txt x output.txt x prog11.2.sh x prog11.2.c x prog11.3.sh x lab11.4.sh x lab11.2.c x
1 #!/bin/bash
2
3 find -x -mtime -7 -mtime +0 -type f > FILES.txt
4 tar -cf archive.tar -T FILES.txt
sh Tab Width: 8 Ln 4, Col 33 BBS
```

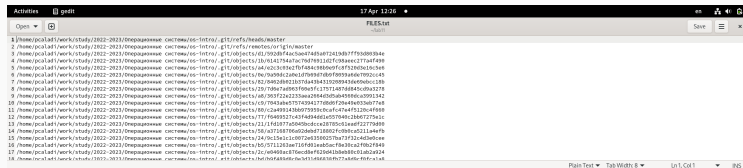
Figure 10: Четвертая программа



```
[pcaladi@fedora lab11]$ gedit lab11.4.sh
[pcaladi@fedora lab11]$ bash lab11.4.sh
[pcaladi@fedora lab11]$ bash lab11.4.sh home/pcaladi/work
find: 'home/pcaladi/work': No such file or directory
[pcaladi@fedora lab11]$ bash lab11.4.sh /home/pcaladi/work
tar: Removing leading '/' from member names
tar: Removing leading '/' from hard link targets
```

Figure 11: Вызов программы в терминале

Выполнение лабораторной работы



```
Activities | gedit 17 Apr 12:26
FILES.txt
~sub11
Save
1 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/refs/heads/master
2 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/refs/remotes/origin/master
3 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/61/7992b9f4ac5a647485a6724193b7ff9368934e
4 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/10/614179487ac7bd7812d2f03a8ec277aaf498
5 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/5a/2c53c3242bf4841989e90f1c87528d3a1c6c48
6 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/9e/9a99d12a6c1d7993e7d80f9959a6e7092c645
7 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/82/8a92d8821b7b4a318419208943d609b0cc18b
8 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/2b/7b9e74d983f96a971175714874844c5d9a3278
9 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/a8/793f22a2255a82864c05ab45886ca991342
10 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/c9/7043ab65757439417708d9f29449c03ab77a8
11 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/80/c2a4912436677933c0ca4c4744f5120c4f680
12 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/77/f6409527c43f4a94661e55704812b66727561c
13 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/21/1fd077ad040bdcce28705c1eae9f22779480
14 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/50/a571087760d246b97138027c08fca2114e4fb
15 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/24/9c15e1c1c0072a3560257a73f32c4d3e0cae
16 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/b0/3711212a61f6f81eab5dcf8a38ca2f0b2f949
17 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/2c/cb4081ac570cdd8f4216413b8b8c01ab2a214
18 /home/pcaladi/work/study/2022-2023/Операционные системы/ос-интра/.git/objects/1d/b4f485682a54314868167c73a6d616fca1a8
```

Figure 12: Результат

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.

1. Лабораторная работа № 10. Программирование в командном процессоре ОС UNIX. Командные файлы [Электронный ресурс]. URL: <https://esystem.rudn.ru/>.

спасибо за внимание!