

Лабораторная работа №11

Сиссе Мохамед Ламин; НММбд-01-22

¹RUDN University, Moscow, Russian Federation

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;
 - `-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 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).

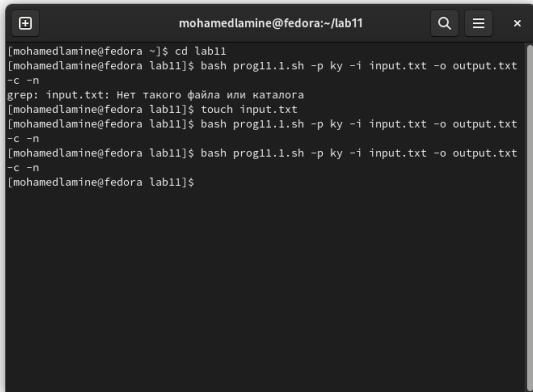
Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: - оболочка Борна (Bourne shell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

- 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).

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;
 - `-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 `-p`. (рис. (fig:001?; fig:002?; fig:003?; fig:004?))

Выполнение лабораторной работы

A terminal window titled 'mohamedlamine@fedora:~/lab11' with search, menu, and close icons. The terminal shows the following commands and output:

```
[mohamedlamine@fedora ~]$ cd lab11
[mohamedlamine@fedora lab11]$ bash prog11.1.sh -p ky -i input.txt -o output.txt
-c -n
grep: input.txt: Нет такого файла или каталога
[mohamedlamine@fedora lab11]$ touch input.txt
[mohamedlamine@fedora lab11]$ bash prog11.1.sh -p ky -i input.txt -o output.txt
-c -n
[mohamedlamine@fedora lab11]$ bash prog11.1.sh -p ky -i input.txt -o output.txt
-c -n
[mohamedlamine@fedora lab11]$
```

Рис. 2: Вызов программы в терминале

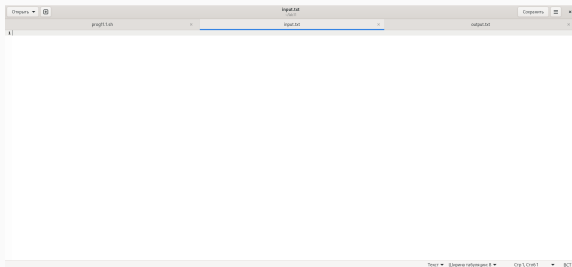


Рис. 3: Результат

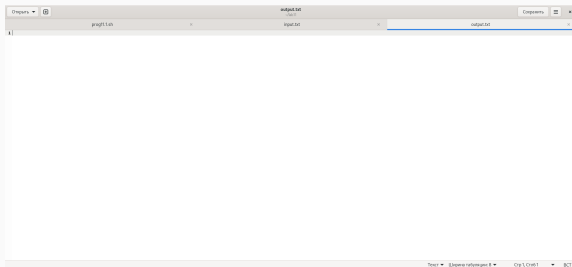
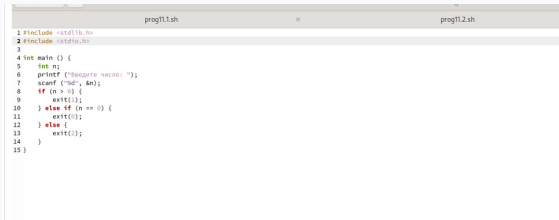


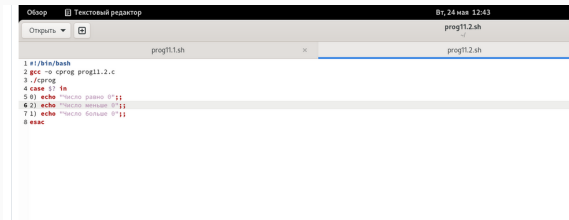
Рис. 4: Результат

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. (fig:005?; fig:006?; fig:007?))



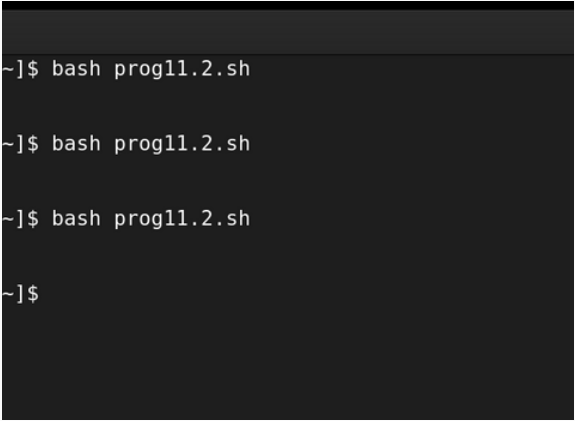
```
1 #include <stdlib.h>
2 #include <stdin.h>
3
4 int main () {
5     int n;
6     printf ("Введите число: ");
7     scanf ("%d", &n);
8     if (n > 0) {
9         exit();
10    } else if (n == 0) {
11        exit();
12    } else {
13        exit();
14    }
15 }
```

Рис. 5: Вторая программа



```
Обзор | Текстовый редактор Вт, 24 мая 12:43
Открыть ▼
prog11.1.sh x prog11.2.sh
1 #!/bin/bash
2 gcc -o cprog prog11.2.c
3 ./cprog
4 case $? in
5 0) echo "число равно 0" ;;
6 2) echo "число меньше 0" ;;
7 1) echo "число больше 0" ;;
8 esac
```

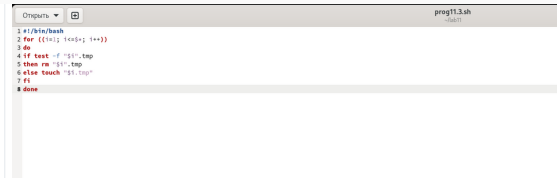
Рис. 6: Вторая программа



```
~]$ bash prog11.2.sh  
  
~]$ bash prog11.2.sh  
  
~]$ bash prog11.2.sh  
  
~]$
```

Рис. 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. (fig:008?; fig:009?))



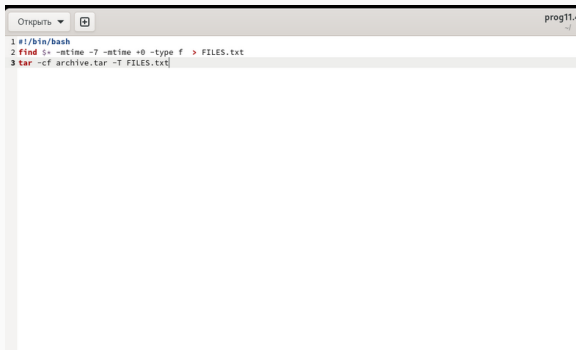
```
1 #!/bin/bash
2 for ((i=1; i<=5; i++))
3 do
4 if test -f "$i".tmp
5 then rm "$i".tmp
6 else touch "$i".tmp"
7 fi
8 done
```

Рис. 8: Третья программа

```
ra lab11]$ bash prog11.3.sh 3
ra lab11]$ ls
  prog11.1.sh  prog11.2.sh
  prog11.2     prog11.3.sh
  prog11.2.c   prog11.4.sh
ra lab11]$ bash prog11.3.sh 3
ra lab11]$ ls
out.txt  prog11.2     prog11.2.sh  prog11.4.sh
prog11.1.sh prog11.2.c   prog11.3.sh
ra lab11]$ bash prog11.3.sh 5
ra lab11]$ ls
out.txt  prog11.2     prog11.3.sh
out.txt  prog11.2.c   prog11.4.sh
prog11.1.sh prog11.2.sh
ra lab11]$ bash prog11.3.sh 5
ra lab11]$
```

Рис. 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. (fig:010?; fig:011?; fig:012?))



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

Рис. 10: Четвертая программа

```
~]$ bash prog11.4.sh home/dmbelicheva/work  
eva/work': Нет такого файла или каталога  
~]$ bash prog11.4.sh /home/dmbelicheva/work  
ьный '/' из имен объектов  
ьные '/' из целей жестких ссылок  
~]$
```

Рис. 11: Вызов программы в терминале

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!