

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

Leysan R. Abdullina

NEC-2022, 25 May

RUDN University, Moscow, Russian Federation

Лабораторная работа No 11.

Программирование в командном
процессоре ОС UNIX. Ветвления и
циклы

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; - C-оболочка (или csh)—надстройка на оболочке Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;

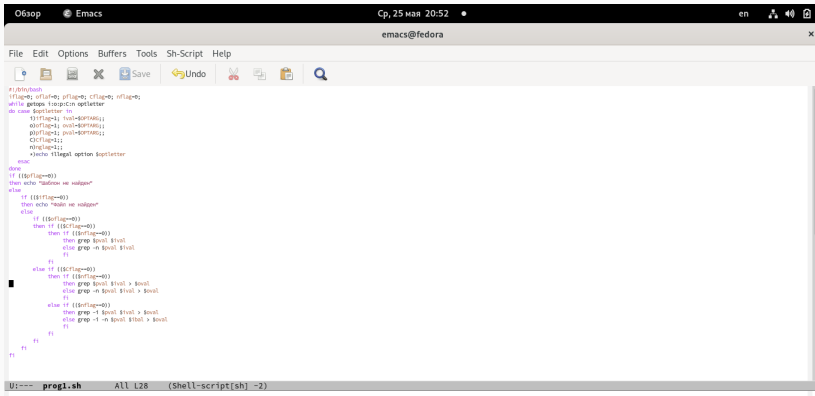
- оболочка Корна (или ksh)—напоминает оболочку C,но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна),в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation). POSIX (Portable 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.

Используя команды `getopts` `grep`, напомним командный файл, который анализирует командную строку с ключами:

- `-i inputfile` —прочитать данные из указанного файла;
- `-o outputfile` —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 `-r`.

Выполнение работы

Для этого мы создаем файл prog1.sh, после чего открываем emacs в фоновом и режиме и начинаем работу. (скриншот 1)

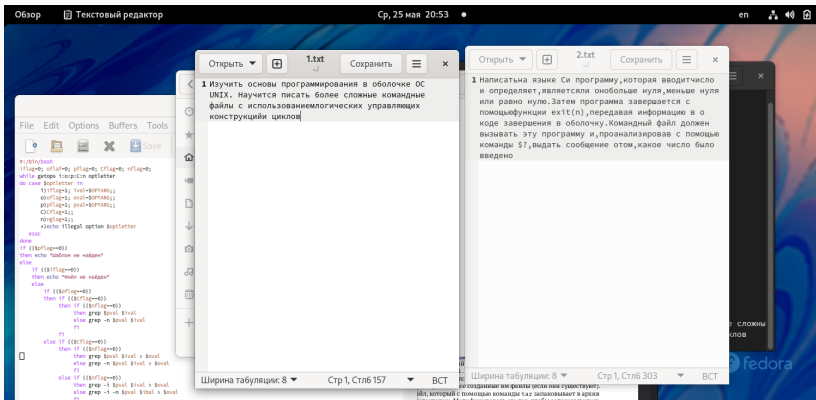


```
#!/bin/bash
iflag=0; oflag=0; pflag=0; cflag=0; rflag=0;
while getopts 'trpocn' optletter
do case $optletter in
  t)iflag=1; test=${OPTARG};
  o)oflag=1; test=${OPTARG};
  p)pflag=1; pval=${OPTARG};
  c)cflag=1;
  r)rflag=1;
  *)echo "illegal option $optletter"
  esac
done
if [[iflag=0]]
then echo "nothing to halp"
else
  if [[iflag=0]]
  then echo "make me halp"
  else
    if [[iflag=0]]
    then if [[cflag=0]]
    then if [[iflag=0]]
    then grep $val $val
    else grep -n $val $val
    fi
    fi
    else if [[cflag=0]]
    then if [[iflag=0]]
    then grep $val $val > $val
    else grep -n $val $val > $val
    fi
    else if [[iflag=0]]
    then grep -r $val $val > $val
    else grep -r -n $val $val > $val
    fi
    fi
  fi
fi
```

Figure 1: Написанная программа

Выполнение работы

Для проверки работы нашей программы предварительно создадим файлы 1.txt и 2.txt и внесем туда информацию. А также не забудем сделать наш файл с программой исполняемым через команду `chmod +x prog1.sh`. После этого вводим команды в консоли `cat ~/1.txt` и `./prog1.sh -i ~/1.txt -o ~/2.txt -p "" -C -t` и смотрим на результат. (скриншоты 2, 3)



Выполнение работы



```
Обзор Терминал Ср, 25 мая 20:55 en
lrabduullina@fedora:~/lab11

[lrabduullina@fedora ~]$ cd mkdir
bash: cd: mkdir: Нет такого файла или каталога
[lrabduullina@fedora ~]$ mkdir lab11
[lrabduullina@fedora ~]$ cd lab11
[lrabduullina@fedora lab11]$ touch prog1
[lrabduullina@fedora lab11]$ emacs &
[1] 2355
[lrabduullina@fedora lab11]$ rm -r prog1
[1]+  Завершено      emacs
[lrabduullina@fedora lab11]$ touch prog1.sh
[lrabduullina@fedora lab11]$ emacs &
[1] 2457
[lrabduullina@fedora lab11]$ touch 1.txt 2.txt
[lrabduullina@fedora lab11]$ ls
1.txt  2.txt  '#prog1#'  '#prog1.sh#'  prog1.sh
[lrabduullina@fedora lab11]$ chmod +x prog1.sh
[lrabduullina@fedora lab11]$ chmod +x prog1.sh
[lrabduullina@fedora lab11]$ cat ~/1.txt
cat: /home/lrabduullina/1.txt: Нет такого файла или каталога
[lrabduullina@fedora lab11]$ cat lab11/1.txt
cat: lab11/1.txt: Нет такого файла или каталога
[lrabduullina@fedora lab11]$ cat ~/1.txt
Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованиемлогических управляющих конструкций циклов
[lrabduullina@fedora lab11]$ ./prog1.sh -i ~/1.txt -o ~/2.txt -p файл -C -n
./prog1.sh: строка 3: getopts: команда не найдена
шаблон не найден
[lrabduullina@fedora lab11]$
```

Figure 3: Проверка - работает успешно

Напишем 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 будет вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.

Выполнение работы

Для этого мы создаем 2 файла: prog2.c, prog2.sh. В них мы пишем код программы на Си и bash соответственно. (скриншоты 4, 5)

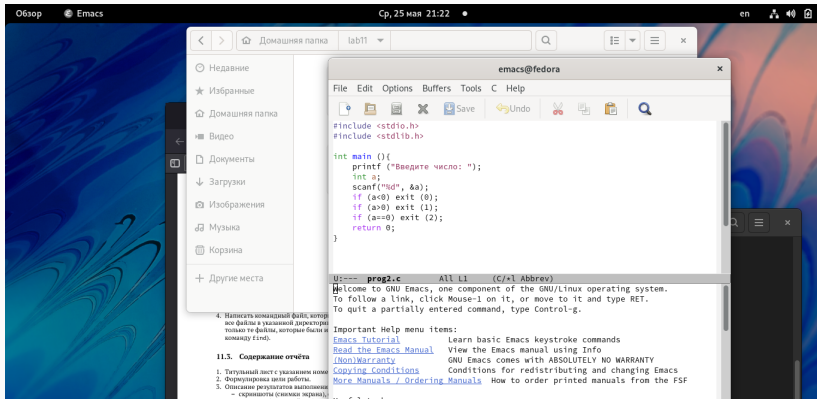


Figure 4: Написанная программа на Си

Выполнение работы

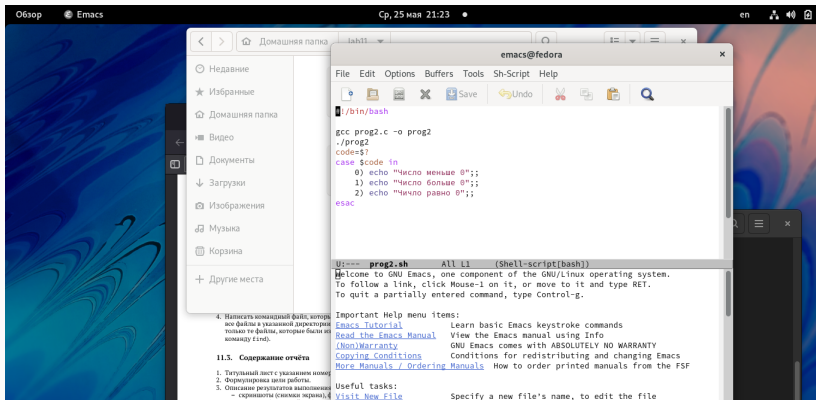


Figure 5: Написанная программа на bash

Выполнение работы

Сделаем файл исполняемым через команду `chmod +x prog2.sh`. И проверим его работу. Задаем случайные числа и смотрим, правильный ли ответ. (скриншот 6)



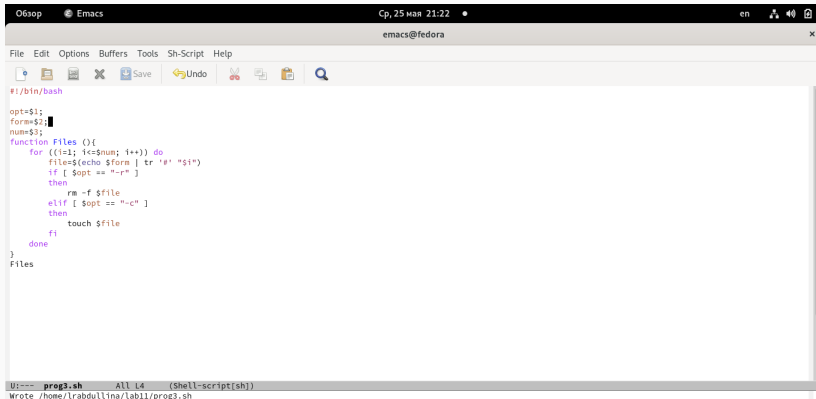
```
Обзор Терминал Ср, 25 мая 21:13 en
lrabdullina@fedora:~/lab11

[lrabdullina@fedora lab11]$ touch prog2.c
[lrabdullina@fedora lab11]$ emacs &
[1] 2898
[lrabdullina@fedora lab11]$ chmod +x p[1]+  Завершён      emacs
prog
chmod: невозможно получить доступ к 'prog': Нет такого файла или каталога
[lrabdullina@fedora lab11]$ chmod +x prog2.
prog2.c  prog2.c-  prog2.sh  prog2.sh-
[lrabdullina@fedora lab11]$ chmod +x prog2.
prog2.c  prog2.c-  prog2.sh  prog2.sh-
[lrabdullina@fedora lab11]$ chmod +x prog2.sh
[lrabdullina@fedora lab11]$ ./prog2.sh
prog2.c:11:10: fatal error: iostream: Нет такого файла или каталога
   1 | #include <iostream>
     |           ^~~~~~
компиляция прервана.
./prog2.sh: строка 4: ./prog2: Нет такого файла или каталога
[lrabdullina@fedora lab11]$ ./prog2.sh
Введите число: 14
Число больше 0
[lrabdullina@fedora lab11]$ ./prog2.sh
Введите число: -5
Число меньше 0
[lrabdullina@fedora lab11]$ ./prog2.sh
Введите число: 0
Число равно 0
[lrabdullina@fedora lab11]$
```

Figure 6: Проверка - работает успешно

3. Напишем командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до `%1` (например `1.tmp, 2.tmp, 3.tmp, 4.tmp` ит.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).

Для начала создадим новый файл prog3.sh и напомним программу.
(скриншот 7)



```
#!/bin/bash

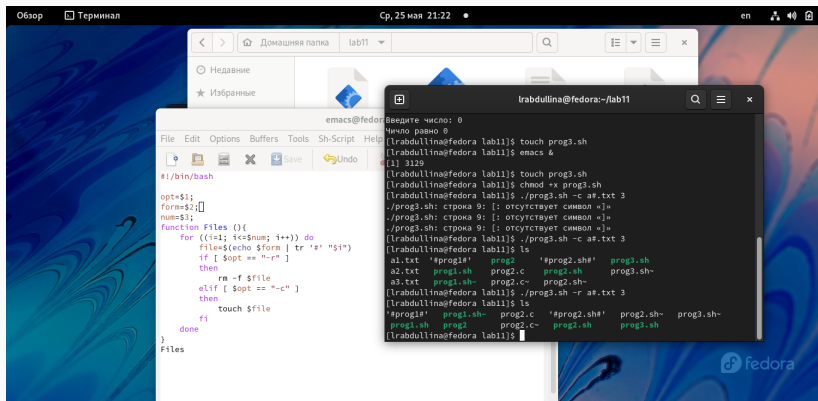
opt=$1;
form=$2;
num=$3;
function Files (){
  for ((i=1; i<=num; i++)) do
    file=$(echo $form | tr '0' "1")
    if [ $opt == "-r" ]
    then
      rm -f $file
    elif [ $opt == "-c" ]
    then
      touch $file
    fi
  done
}
Files
```

U:--- prog3.sh All L4 (Shell-script[sh])
Wrote /home/lrabdullina/lab11/prog3.sh

Figure 7: Написанная программа

Выполнение работы

Сделаем файл исполняемым через команду `chmod +x prog3.sh`. И проверим его работу. Создадим через команду `./prog3.sh -c a#.txt 3 3` новых файла. А затем через команду `./prog3.sh -r a#.txt 3` удалим их (скриншот 8)



```
emacs@fedora:~/lab11$ touch prog3.sh
[lab11]$ emacs &
[1] 3129
[lab11]$ touch prog3.sh
[lab11]$ chmod +x prog3.sh
[lab11]$ ./prog3.sh -c a#.txt 3
./prog3.sh: строка 9: [: отсутствует символ «|»
./prog3.sh: строка 9: [: отсутствует символ «|»
[lab11]$ ./prog3.sh -c a#.txt 3
[lab11]$ ls
a1.txt  'prog1#'  prog2  'prog2.sh#'  prog3.sh
a2.txt  prog1.sh  prog2.c  prog2.sh  prog3.sh-
a3.txt  prog1.sh-  prog2.c-  prog2.sh-
[lab11]$ ./prog3.sh -r a#.txt 3
[lab11]$ ls
'prog1#'  prog1.sh-  prog2.c  'prog2.sh#'  prog2.sh-  prog3.sh-
prog1.sh  prog2     prog2.c-  prog2.sh     prog3.sh
[lab11]$
```

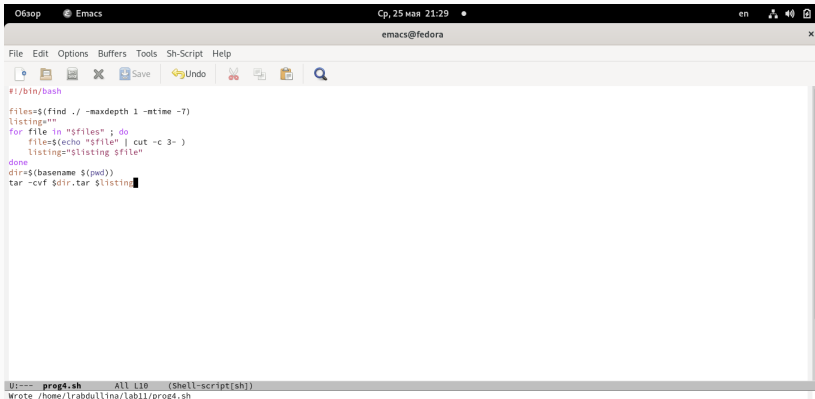
```
#!/bin/bash
opt=$1;
form=$2;
num=$3;
function Files () {
    for ((i=1; i<=$num; i++)) do
        file=$(echo $form | tr 's' '$i')
        if [ $opt == "-r" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
Files
```

Figure 8: Проверка - работает успешно

4. Напишем командный файл, который с помощью команды `tar` запаковывает архив все файлы в указанной директории. Модифицируем его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду `find`).

Выполнение работы

Для начала создадим новый файл prog4.sh и напомним программу.
(скриншот 9)



```
Обзор Emacs                               Ср, 25 мая 21:29 • en
emacs@fedora
File Edit Options Buffers Tools Sh-Script Help
Save Undo
#!/bin/bash

files=$(find ./ -maxdepth 1 -mtime -7)
listing=""
for file in "$files" ; do
    file=$(echo "$file" | cut -c 3- )
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

U:--- prog4.sh All L10 (Shell-script[sh])
Wrote /home/lrabdullina/lab11/prog4.sh

Figure 9: Написанная программа

Выполнение работы

Вводим команду `sudo ~/lab11/prog4.sh` и смотрим на результат.

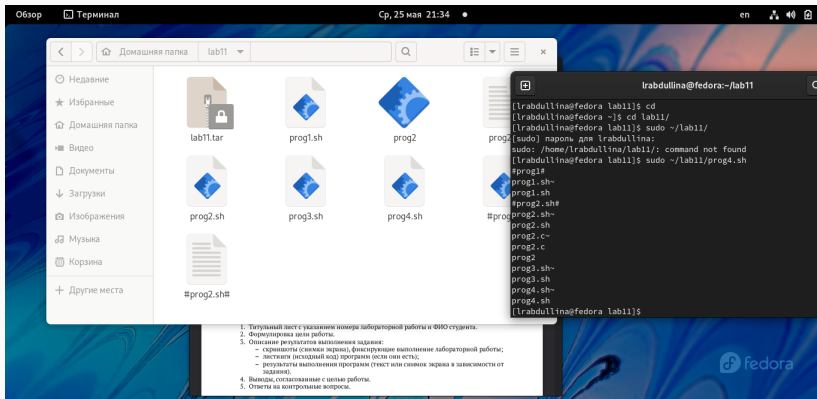


Figure 10: Проверка - работает успешно

В ходе лабораторной работы мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.