

# Лабораторная работа №10

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

---

Кузнецова С. В.

22 апреля 2023

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Кузнецова София Вадимовна
- студент прикладной информатики
- Российский университет дружбы народов

Цель

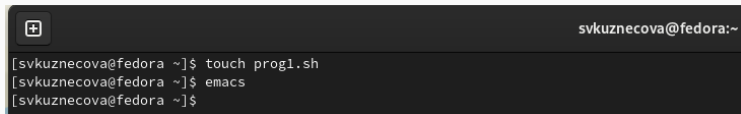
---

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

## Выполнение лабораторной работы

---

Открываем терминал и создаём в домашнем каталоге файл prog1.sh. После чего перейдём в emacs.

A terminal window with a dark background. The title bar shows a window icon and the text 'svkuznecova@fedora:~'. The terminal content shows three lines of commands and their prompts: '[svkuznecova@fedora ~]\$ touch prog1.sh', '[svkuznecova@fedora ~]\$ emacs', and '[svkuznecova@fedora ~]\$'.

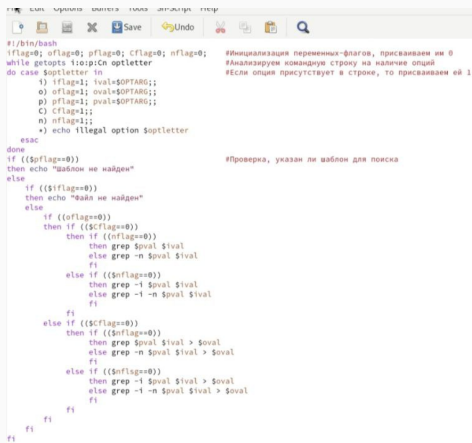
```
[svkuznecova@fedora ~]$ touch prog1.sh
[svkuznecova@fedora ~]$ emacs
[svkuznecova@fedora ~]$
```

Рис. 1: Создание нового файла для скрипта

В emacs откроем созданный файл `prog1.sh` и приступим к написанию скрипта, который анализирует командную строку с определёнными ключами, а затем ищет в указанном файле нужные строки, определяемые ключом `p`.



# Написание первого скрипта



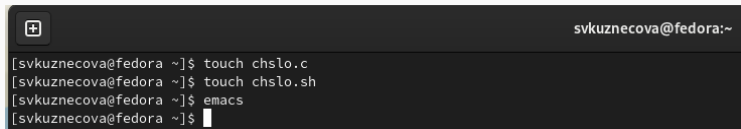
```
#!/bin/bash
iflag=0; oflag=0; pflag=0; cflag=0; nflag=0; #Инициализация переменных-флагов, присваиваем им 0
while getopts 1:0:p:c:n optletter          #Анализируем командную строку на наличие опций
do case $optletter in
  1) iflag=1; ival=$OPTARG;;
  o) oflag=1; oval=$OPTARG;;
  p) pflag=1; pval=$OPTARG;;
  C) cflag=1;;
  n) nflag=1;;
  *) echo illegal option $optletter
  esac
done
if (($flag==0))                           #Проверка, указан ли шаблон для поиска
then echo "Шаблон не найден"
else
  if (($iflag==0))
  then echo "Файл не найден"
  else
    if ((oflag==0))
    then if (($cflag==0))
        then if ((nflag==0))
            then grep $pval $ival
            else grep -n $pval $ival
            fi
        else if (($nflag==0))
            then grep -i $pval $ival
            else grep -i -n $pval $ival
            fi
        fi
    else if (($cflag==0))
    then if (($nflag==0))
        then grep $pval $ival > $oval
        else grep -n $pval $ival > $oval
        fi
    else if (($nflag==0))
    then grep -i $pval $ival > $oval
    else grep -i -n $pval $ival > $oval
    fi
  fi
fi
fi
```

Рис. 2: Написание первого скрипта



## Создание файлов для второго задания

Теперь открываем в терминале два файла(chslo.c и chslo.sh), для второго задания.  
Открываем emacs.

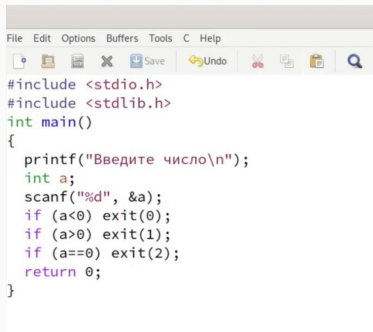


```
svkuznecova@fedora:~  
[svkuznecova@fedora ~]$ touch chslo.c  
[svkuznecova@fedora ~]$ touch chslo.sh  
[svkuznecova@fedora ~]$ emacs  
[svkuznecova@fedora ~]$
```

Рис. 4: Создание двух файлов и открытие emacs

## Написание скрипта на языке программирования Си

Открываем файл chslo.c и начинаем писать 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.

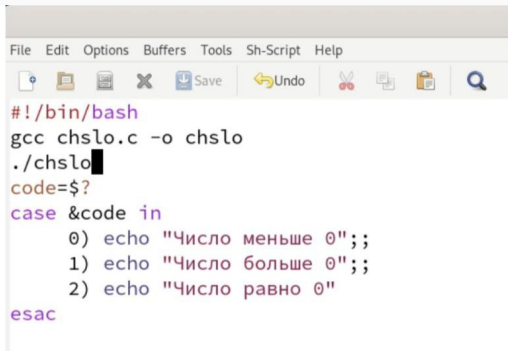
A screenshot of a text editor window showing a C program. The window has a menu bar with 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'C', and 'Help'. Below the menu bar is a toolbar with icons for file operations and editing. The code is written in a monospaced font with syntax highlighting: preprocessor directives are in blue, keywords in green, identifiers in blue, strings in red, and control flow in purple. The program includes `<stdio.h>` and `<stdlib.h>`, defines a `main` function, prints a prompt, reads an integer, and uses `exit` with different codes based on the input's sign.

```
File Edit Options Buffers Tools C Help
#include <stdio.h>
#include <stdlib.h>
int main()
{
    printf("Введите число\n");
    int a;
    scanf("%d", &a);
    if (a<0) exit(0);
    if (a>0) exit(1);
    if (a==0) exit(2);
    return 0;
}
```

Рис. 5: Написание программы на языке Си

## Написание скрипта для второго задания

После программы на Си, в файле chslo.sh пишем командный файл, который должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было выведено.

A screenshot of a text editor window with a menu bar (File, Edit, Options, Buffers, Tools, Sh-Script, Help) and a toolbar with icons for file operations and editing. The editor contains a shell script for chslo.sh. The script starts with a shebang line, followed by compilation and execution commands, then uses the exit status of the last command to print a message in Russian based on whether the number was less than, greater than, or equal to zero.

```
#!/bin/bash
gcc chslo.c -o chslo
./chslo
code=$?
case &code in
    0) echo "Число меньше 0";;
    1) echo "Число больше 0";;
    2) echo "Число равно 0"
esac
```

Рис. 6: Написание командного файла для второго задания

## Право на выполнение и последующая проверка

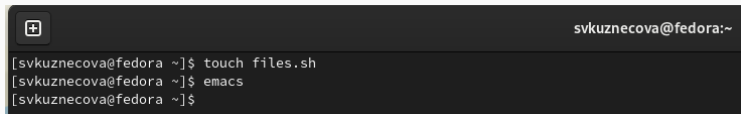
Сохраняем файл и также даём в терминале право на выполнение для файла chslo.sh.

Запускаем файл chslo.sh.

```
[ismahorin@fedora ~]$ touch chslo.c
[ismahorin@fedora ~]$ touch chslo.sh
[ismahorin@fedora ~]$ emacs
[ismahorin@fedora ~]$ emacs
[ismahorin@fedora ~]$ chmod +x chslo.sh
[ismahorin@fedora ~]$ ./chslo.sh
Введите число
0
./chslo.sh: строка 5: синтаксическая ошибка рядом с неожиданным маркером «&»
./chslo.sh: строка 5: `case &code in'
[ismahorin@fedora ~]$ emacs
[ismahorin@fedora ~]$ ./chslo.sh
Введите число
0
Число равно 0
[ismahorin@fedora ~]$ ./chslo.sh
Введите число
0
Число равно 0
[ismahorin@fedora ~]$ ./chslo.sh
Введите число
-1
Число меньше 0
[ismahorin@fedora ~]$ ./chslo.sh
Введите число
99
Число больше 0
[ismahorin@fedora ~]$ clear
```

Рис. 7: Право на выполнение, запуск файла

Снова в домашнем каталоге создаём файл, но уже для третьего задания. Запускаем emacs.

A terminal window with a dark background. The title bar shows a window icon and the text 'svkuznecova@fedora:~'. The terminal content shows three lines of commands and their prompts: '[svkuznecova@fedora ~]\$ touch files.sh', '[svkuznecova@fedora ~]\$ emacs', and '[svkuznecova@fedora ~]\$'.

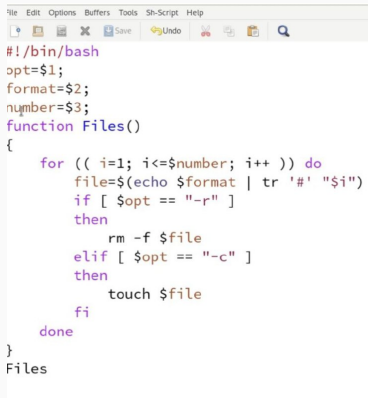
```
svkuznecova@fedora:~  
[svkuznecova@fedora ~]$ touch files.sh  
[svkuznecova@fedora ~]$ emacs  
[svkuznecova@fedora ~]$
```

Рис. 8: Создание третьего файла

После открытия файла `files.sh` напишем 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).



## Написание третьего скрипта

A screenshot of a text editor window titled 'Sh-Script'. The window has a menu bar with 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Sh-Script', and 'Help'. Below the menu bar is a toolbar with icons for file operations (open, save, print, delete, copy, paste, undo, redo) and a search icon. The main text area contains a Bash script. The script starts with a shebang line, followed by variable assignments for 'opt', 'format', and 'number'. It then defines a function 'Files()' which contains a 'for' loop. Inside the loop, it constructs a filename, checks for the '-r' option to remove the file or the '-c' option to create the file, and then either removes or creates the file. The function ends with a closing brace, and the script concludes with a call to the 'Files' function.

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

Рис. 9: Написание третьего скрипта

# Право на выполнение и запуск файла

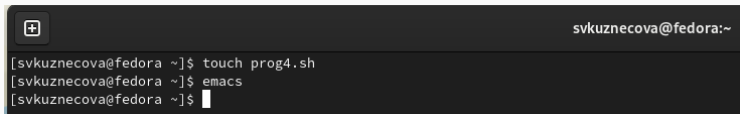
Сохраняем наш скрипт и даём право на выполнение. Запускаем файл и создаём три текстовых файла. Удаляем эти три файла.

```
ismahorin@fedora ~$ touch files.sh
ismahorin@fedora ~$ emacs
ismahorin@fedora ~$ chmod +x files.sh
ismahorin@fedora ~$ ls
a1.txt  backup  chslo.c  files.sh  lab19_1.sh  lab19_3.sh  progr1.sh  Загрузки  'Рабочий стол'
a2.txt  bin     chslo.sh  'lab07.sh'  lab19_2.sh  lab19_4.sh  work      Избранное  Шаблоны
#abc1#  chslo  chslo.sh  lab07.sh  lab19_2.sh  lab19_4.sh  Видео     Музыка
abc1    chslo.c  files.sh  lab19_1.sh  lab19_3.sh  progr1.sh  Документы  Общедоступные
ismahorin@fedora ~$ ./files.sh -c abc1.txt 3
ismahorin@fedora ~$ ls
a1.txt  abc2.txt  chslo.c  files.sh  lab19_2.sh  lab19_4.sh  Документы  'Рабочий стол'
a2.txt  abc2.txt  chslo.c  'lab07.sh'  lab19_2.sh  lab19_4.sh  progr1.sh  Загрузки  Шаблоны
#abc1#  backup  chslo.sh  lab07.sh  lab19_3.sh  progr1.sh  Избранное
abc1    bin     chslo.sh  lab19_1.sh  lab19_3.sh  work      Музыка
abc1.txt  chslo  files.sh  lab19_1.sh  lab19_4.sh  Видео     Общедоступные
ismahorin@fedora ~$ ./files.sh -r abc1.txt 3
ismahorin@fedora ~$ ls
a1.txt  backup  chslo.c  files.sh  lab19_1.sh  lab19_3.sh  progr1.sh  Загрузки  'Рабочий стол'
a2.txt  bin     chslo.sh  'lab07.sh'  lab19_2.sh  lab19_4.sh  work      Избранное  Шаблоны
#abc1#  chslo  chslo.sh  lab07.sh  lab19_2.sh  lab19_4.sh  Видео     Музыка
abc1    chslo.c  files.sh  lab19_1.sh  lab19_3.sh  progr1.sh  Документы  Общедоступные
ismahorin@fedora ~$ clear
```

Рис. 10: Право на выполнение, запуск файла

## Создание файла для четвёртого скрипта

Создаём последний файл для четвёртого скрипта. Запускаем emacs.



```
[svkuznecova@fedora ~]$ touch prog4.sh
[svkuznecova@fedora ~]$ emacs
[svkuznecova@fedora ~]$
```

Рис. 11: Создание четвёртого скрипта

## Написание четвёртого скрипта

В четвёртом файле напишем 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`).

A screenshot of a text editor window with a menu bar (File, Edit, Options, Buffers, Tools, Sh-Script, Help) and a toolbar. The editor contains a shell script for archiving files modified within the last week. The script uses `find` to locate files, `cut` to format the output, and `tar` to create the archive.

```
#!/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
```

Рис. 12: Написание четвёртого скрипта

## Право на выполнение и запуск файла для нужного каталога

Сохраним файл и выйдем из emacs. Как делали ранее, дадим файлу право на выполнение и запустим его для каталога Catalog1 (который мы создали ранее в терминале и в который перенесли некоторые файлы из домашнего каталога).

```
ismaherin@fedora ~]$ touch prog4.sh
ismaherin@fedora ~]$ emacs
ismaherin@fedora ~]$ chmod +x prog4.sh
ismaherin@fedora ~]$ mkdir Catalog1
ismaherin@fedora ~]$ ls
a1.txt  vim      chslo.sh  lab07.sh  lab10_3.sh  prog1.sh  Документы  'Рабочий стол'
a2.txt  emacslog chslo.sh  lab10_1.sh  lab10_3.sh-  prog4.sh  Загрузки   Рабочий стол
'abc12' chslo    files.sh  lab10_1.sh-  lab10_4.sh  prog4.sh-  Избранное  Рабочий стол
abc1    chslo.c  'files.sh'  lab10_2.sh  lab10_4.sh-  prog4.sh-  Печата     Рабочий стол
backlog chslo.c- 'lab07.sh'  lab10_2.sh-  prog1.sh    prog4.sh-  Избранное  Рабочий стол

ismaherin@fedora ~]$ cd ~/Catalog1
ismaherin@fedora Catalog1]$ ls
ismaherin@fedora Catalog1]$ ls
a1.txt  a2.txt  abc1  chslo.c  chslo.sh  files.sh  prog1.sh
ismaherin@fedora Catalog1]$ ls -l
-rwxr-xr-x 1 ismaherin ismaherin 4096 102216 a1.txt 102217 a2.txt 102218 abc1 102219 chslo.c 102220 chslo.sh 102221 files.sh 102222 prog1.sh
ismaherin@fedora Catalog1]$ sudo ./prog4.sh
[sudo] пароль для ismaherin:
a1.txt
a2.txt
chslo.c
chslo.sh
files.sh
prog1.sh
ismaherin@fedora Catalog1]$ tar -tf Catalog1.tar
a1.txt
a2.txt
chslo.c
chslo.sh
files.sh
prog1.sh
ismaherin@fedora Catalog1]$
ismaherin@fedora Catalog1]$
```

Рис. 13: Право на выполнение, запуск файла для каталога Catalog1

Перейдём в файл и выполним проверку архивации.



Рис. 14: Проверка

## Вывод

---

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



Спасибо за внимание!