

Лабораторная работа № 11.

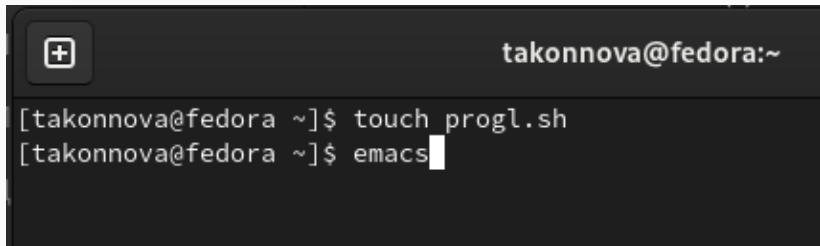
Коннова Татьяна Алексеевна

2023

RUDN, Москва, Россия

Программирование в командном
процессоре ОС UNIX. Ветвления и
циклы.

Создание первого файла для скрипта



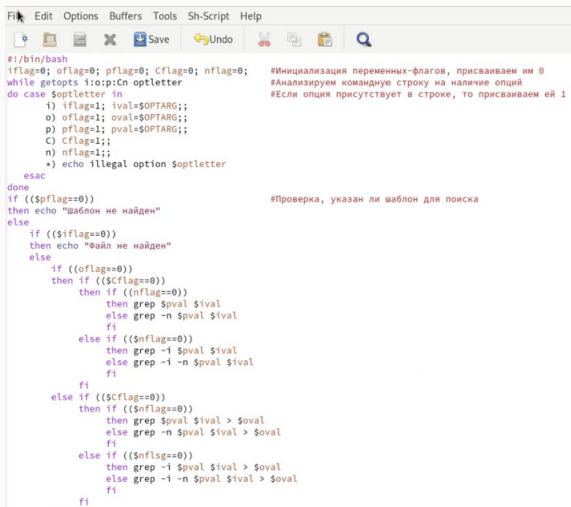
A terminal window with a dark background. The title bar at the top shows a window icon on the left and the text "takonnova@fedora:~" on the right. The terminal content shows two lines of text: the first line is "[takonnova@fedora ~]\$ touch progl.sh" and the second line is "[takonnova@fedora ~]\$ emacs" followed by a white cursor block.

```
takonnova@fedora:~  
[takonnova@fedora ~]$ touch progl.sh  
[takonnova@fedora ~]$ emacs
```

Рис. 1: Создание нового файла для скрипта

Написать скрипт, 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 i:op:Cn optletter                 #Анализируем командную строку на наличие опций
do case $optletter in                           #Если опция присутствует в строке, то присваиваем ей 1
  i) iflag=1; ival=$OPTARG;;
  o) oflag=1;  oval=$OPTARG;;
  p) pflag=1;  pval=$OPTARG;;
  C) cflag=1;;
  n) nflag=1;;
  *) echo illegal option $optletter
  esac
done
if (($pflag==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
fi
```

Рис. 2: Написание первого скрипта

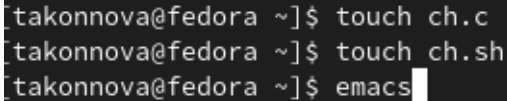
Запуск и проверка выполнения первого скрипта

```
progl.sh
program.cpp
ski.plases
teers.txt
work
Архитектура
Видео
Документы
Загрузки
Изображения
Музыка
Общедоступные
'Рабочий стол'
Шаблоны
[takonnova@fedora ~]$ mc

[takonnova@fedora ~]$ cat a1.txt
planet you rat[takonnova@fedora ~]$ ./progl.sh -i a1.txt -o a2.txt -p rat -n
bash: ./progl.sh: Отказано в доступе
[takonnova@fedora ~]$ ./progl.sh -i a1.txt -o a2.txt -p rat -n
bash: ./progl.sh: Отказано в доступе
[takonnova@fedora ~]$ ./progl.sh -i a1.txt -C -n
bash: ./progl.sh: Отказано в доступе
[takonnova@fedora ~]$
```

Рис. 3: Право на выполнение, запуск файла и проверка

Создание файлов для второго задания

A terminal window with a dark background and light gray text. It shows three lines of commands being executed by a user named takonnova on a fedora system. The first two lines use the 'touch' command to create files 'ch.c' and 'ch.sh'. The third line uses the 'emacs' command to open the Emacs editor, with a white cursor visible at the end of the command.

```
takonnova@fedora ~]$ touch ch.c  
takonnova@fedora ~]$ touch ch.sh  
takonnova@fedora ~]$ emacs
```

Рис. 4: Создание двух файлов и открытие emacs

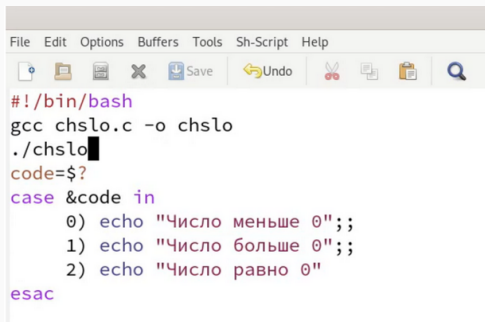
Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку.


```
#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

Написать командный файл, который должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.

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. The editor contains a shell script with the following content:

```
#!/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
```

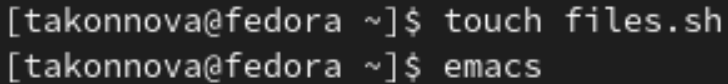
Рис. 6: Написание командного файла для второго задания

Право на выполнение и последующая проверка



```
takonnova@fedora:~  
[takonnova@fedora ~]$ touch ch.c  
[takonnova@fedora ~]$ touch ch.sh  
[takonnova@fedora ~]$ emacs  
bash: emacs: команда не найдена...  
^[[АПакеты, предоставляющие этот файл:  
'emacs'  
'emacs-lucid'  
'emacs-nox'  
[takonnova@fedora ~]$ chmod +x ch.sh  
[takonnova@fedora ~]$ ./ch.sh  
[takonnova@fedora ~]$ emacs  
bash: emacs: команда не найдена...  
^[[АПакеты, предоставляющие этот файл:  
'emacs'  
'emacs-lucid'  
'emacs-nox'  
[takonnova@fedora ~]$ ./ch.sh  
[takonnova@fedora ~]$
```

Рис. 7: Право на выполнение, запуск файла

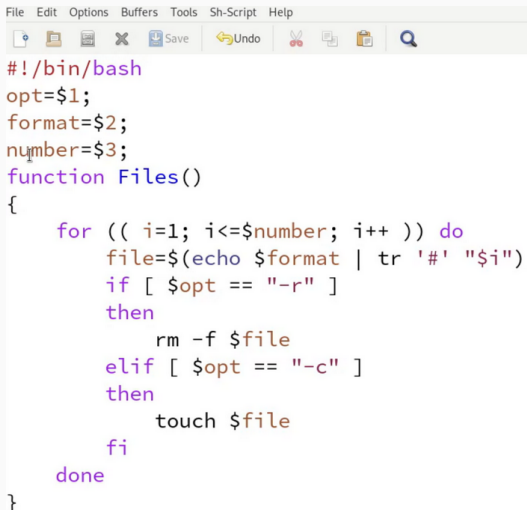


```
[takonnova@fedora ~]$ touch files.sh  
[takonnova@fedora ~]$ emacs
```

Рис. 8: Создание третьего файла

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).

Написание третьего скрипта



The image shows a screenshot of a text editor window titled "Sh-Script". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Sh-Script", and "Help". The toolbar contains icons for opening, saving, undo, redo, and search. The script content is as follows:

```
#!/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
}
```

Рис. 9: Написание третьего скрипта

Право на выполнение и запуск файла

a1.txt	backup	chslo.c~	files.sh~	lab10_1.sh~	lab10_3.sh~	prog1.sh~	Загрузки	'Рабочий стол'
a2.txt	bin	chslo.sh	'#lab07.sh#'	lab10_2.sh	lab10_4.sh	work	Изображения	Шаблоны
#abc1#'	chslo	chslo.sh~	lab07.sh	lab10_2.sh~	lab10_4.sh~	Видео	Музыка	
abc1	chslo.c	files.sh	lab10_1.sh	lab10_3.sh	prog1.sh	Документы	Общедоступные	

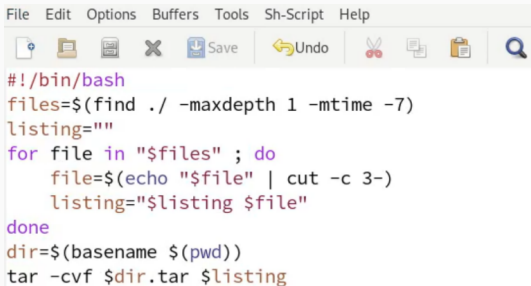
Рис. 10: Право на выполнение, запуск файла

```
[takonnova@fedora ~]$ touch program4.sh  
[takonnova@fedora ~]$ emacs
```

Рис. 11: Создание четвёртого файла

Написание четвёртого скрипта

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`).

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. The editor contains a shell script that uses the `find` command to locate files modified within the last 7 days and then uses `tar` to archive them.

```
#!/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

a1.txt	bin	chslo.sh	lab07.sh	lab10_3.sh	prog1.sh~	Документы	Рабочий
a2.txt	Catalog1	chslo.sh~	lab10_1.sh	lab10_3.sh~	prog4.sh	Загрузки	Шаблоны
#abc1#'	chslo	files.sh	lab10_1.sh~	lab10_4.sh	prog4.sh~	Изображения	
abc1	chslo.c	files.sh~	lab10_2.sh	lab10_4.sh~	work	Музыка	
backup	chslo.c~	'#lab07.sh#'	lab10_2.sh~	prog1.sh	Видео	Общедоступные	

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

Проверка

Выводы

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!