

Лабораторная работа №13

Операционные системы

Павлова Т. Ю.

Российский университет дружбы народов, Москва, Россия

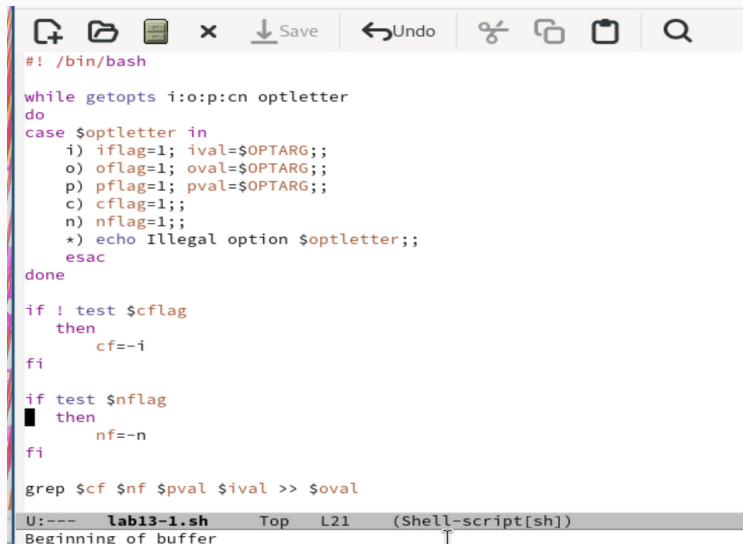
Целью данной лабораторной работы является изучение основ программирования в оболочке ОС UNIX, а также научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-i` input file — прочитать данные из указанного файла; `-o` output file — 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`.
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 `n` 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 ??? (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).

Bash (Bourne Again Shell) — это мощная командная оболочка Unix, которая используется для выполнения различных задач в терминале. Bash предоставляет интерактивный интерфейс, в котором пользователи могут вводить команды, а затем получать результаты. Она также поддерживает скрипты оболочки, которые представляют собой текстовые файлы, содержащие последовательность команд Bash для автоматизации задач. Bash широко используется в средах Unix и Linux, а также поддерживается Windows с помощью подсистемы Windows для Linux (WSL). Перечислим основные возможности этой оболочки. Обработка команд. Bash может обрабатывать как простые, так и сложные команды. Простые состоят из одного действия и, возможно, некоторых аргументов. Сложные команды могут содержать несколько простых, объединенных с помощью операторов конвейера (`|`), перенаправления ввода и вывода (`<`, `>`, `>>`), условных операторов (`if/else`, `case/esac`, `while/do`). О них мы расскажем позже. Расширенный ввод, редактирование строк. Оболочка предоставляет функции расширенного ввода, такие как автодополнение, которое предлагает возможные варианты завершения команд и имен файлов по мере их ввода. Она также поддерживает

Выполнение лабораторной работы

Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-i` input file — прочитать данные из указанного файла; `-o` output file — 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` (рис. 1), (рис. 2), (рис. 3), (рис. 4).



```
#!/bin/bash

while getopts i:o:p:cn optletter
do
case $optletter in
    i) iflag=1; ival=$OPTARG;;
    o) oflag=1; oval=$OPTARG;;
    p) pflag=1; pval=$OPTARG;;
    c) cflag=1;;
    n) nflag=1;;
    *) echo Illegal option $optletter;;
    esac
done

if ! test $cflag
then
    cf=-i
fi

if test $nflag
then
    nf=-n
fi

grep $cf $nf $pval $ival >> $oval

U:--- lab13-1.sh Top L21 (Shell-script[sh])
Beginning of buffer
```

```
tanya@tatyana Pavlova files_for_lab13]$ ls
input.txt lab13-1.sh lab13-1.sh~ output.txt
tanya@tatyana Pavlova files_for_lab13]$ chmod +x lab13-1.sh
tanya@tatyana Pavlova files_for_lab13]$ bash lab13-1.sh -p должен -i input.txt -o output.txt -c -n
tanya@tatyana Pavlova files_for_lab13$
```

Рис. 2: Компиляция файла

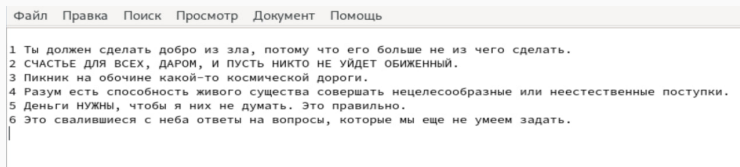


Рис. 3: input.txt

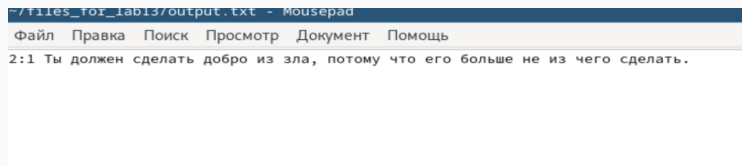
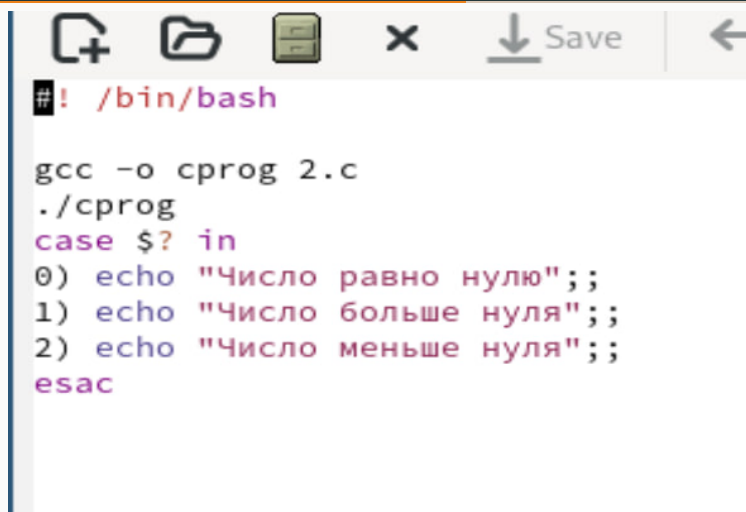


Рис. 4: output.txt

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), (рис. 6), (рис. 7).

```
#include <stdlib.h>
#include <stdio.h>

int main () {
    int n;
    printf ("Введите число: ");
    scanf ("%d", &n);
    if(n>0){
        exit(1);
    }
    else if (n==0) {
        exit(0);
    }
    else {
        exit(2);
    }
}
```



The image shows a terminal window with a standard Linux-style toolbar at the top. The toolbar includes icons for opening a file, saving, and closing the window, along with a 'Save' button and a back arrow. The terminal content shows a shell script starting with a shebang line, followed by compilation and execution commands, and a case statement for conditional output.

```
#!/bin/bash


gcc -o cprog 2.c
./cprog
case $? in
0) echo "Число равно нулю";;
1) echo "Число больше нуля";;
2) echo "Число меньше нуля";;
esac
```

Рис. 6: lab13-2.sh

```
[tanya@tatyana-pavlova files_for_lab13]$ bash lab13-2.sh  
Введите число: 6  
Число больше нуля  
[tanya@tatyana-pavlova files_for_lab13]$
```

Рис. 7: Компиляция файла

Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до ??? (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), (рис. 9).



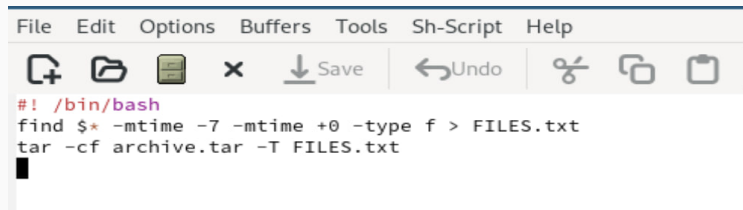
```
#!/bin/bash
for ((i=1; i<=$*; i++))
do
if test -f "$i".tmp
then rm "$i".tmp
else touch "$i.tmp"
fi
done
```



```
[tanya@tatyana-pavlova files_for_lab13]$ bash lab13-3.sh 4  
[tanya@tatyana-pavlova files_for_lab13]$ ls  
1.tmp 2.c~ 3.tmp cprog lab13-1.sh lab13-2.sh lab13-3.sh output.txt  
2.c 2.tmp 4.tmp input.txt lab13-1.sh~ lab13-2.sh~ lab13-3.sh~
```

Рис. 9: Компиляция файла

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), (рис. 11), (рис. 12).



The image shows a screenshot of a text editor window. The title bar is not visible. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Sh-Script', and 'Help'. The toolbar contains icons for opening a file, saving a file, closing a file, a close button (X), a save button (floppy disk), a 'Save' label, an 'Undo' button (curved arrow), a cut button (scissors), a copy button (two overlapping squares), and a paste button (clipboard). The text area contains the following shell script:

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

Рис. 10: lab13-4.sh

```
[tanya@tatyana-pavlova files_for_lab13]$ chmod +x lab13-4.sh  
[tanya@tatyana-pavlova files_for_lab13]$ bash lab13-4.sh /home/tanya/files_for_lab13  
[tanya@tatyana-pavlova files_for_lab13]$
```

Рис. 11: Компиляция файла



archive.tar

При выполнении данной лабораторной работы, я изучила основы программирования в оболочке ОС UNIX, а также научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.