

Отчёт по лабораторной работе №11

Операционные системы

Обрезкова Анастасия Владимировна

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Выводы	12
4	Ответы на контрольные вопросы	13

Список иллюстраций

2.1	Создание рабочего пространства	5
2.2	Создание файла	6
2.3	Текст скрипта	6
2.4	Создание файлов	7
2.5	Проверка скрипта	7
2.6	Создание файлов	7
2.7	Скрипт	8
2.8	Скрипт	8
2.9	Проверка	9
2.10	Создание файла	9
2.11	Скрипт	10
2.12	Проверка	10
2.13	Скрипт	11
2.14	Проверка	11
2.15	Проверка	11

1 Цель работы

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

2 Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, написала командный файл, который анализирует командную строку с ключами:

- `-iinputfile` — прочитать данные из указанного файла;
- `-ooutputfile` — вывести данные в указанный файл;
- `-р` — 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`.

Для начала создала папку, где будут находиться все программы до момента переноса в папку `lab11` и перешла в нее. (рис. 2.1).

```
[avobrezkova@avobrezkova:~] cd ~/work/os
[avobrezkova@avobrezkova:~/work/os] mkdir lab11
[avobrezkova@avobrezkova:~/work/os] cd lab 11
bash: cd: слишком много аргументов
[avobrezkova@avobrezkova:~/work/os] cd lfb11
bash: cd: lfb11: Нет такого файла или каталога
[avobrezkova@avobrezkova:~/work/os] cd lab11
```

Рис. 2.1: Создание рабочего пространства

Создала файл для написания скрипта и открыла его. (рис. 2.2)

```

[avobrezkova@avobrezkova:~/work/os/lab11] touch prog111.sh
[avobrezkova@avobrezkova:~/work/os/lab11] emacs $
[avobrezkova@avobrezkova:~/work/os/lab11] emacs prog111.sh

```

Рис. 2.2: Создание файла

Написала скрипт для выполнения первого задания. (рис. 2.3)

```

#!/bin/bash
iflag=0; oflag=0; pflag=0; cflag=0; nflag=0;
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 $optletter
    esac
done
if (($pflag=0))
then echo "Шаблон отсутствует"
else
    if ((Iflag=0))
    then echo "Файл отсутствует"
    exit
    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

```

U:--- prog111.sh Top L22 (Shell-script[bash])

Рис. 2.3: Текст скрипта

Для проверки работы скрипта создала два текстовых файла. В один из них записала текст для проверки. (рис. 2.4)

```

[avobrezkova@avobrezkova:~/work/os/lab11] touch 1.txt 2.txt
[avobrezkova@avobrezkova:~/work/os/lab11] ls
'$' 1.txt 2.txt prog111.sh prog111.sh~
[avobrezkova@avobrezkova:~/work/os/lab11] rm $
[avobrezkova@avobrezkova:~/work/os/lab11] ls
1.txt 2.txt prog111.sh prog111.sh~
[avobrezkova@avobrezkova:~/work/os/lab11] chmod +x prog111.sh

```

Рис. 2.4: Создание файлов

Дала право на выполнение с помощью команды “chmod” и использовала команды для проверки. Скрипт работает корректно.(рис. 2.5)

```

[avobrezkova@avobrezkova:~/work/os/lab11] cat 2.txt
[avobrezkova@avobrezkova:~/work/os/lab11] chmod +x prog111.sh
[avobrezkova@avobrezkova:~/work/os/lab11] cat 1.txt
Hello, my name is Nastya.
Hello, my name is Dasha.
Hello, my name is Kseniya.
Hello, my name is Anna.
[avobrezkova@avobrezkova:~/work/os/lab11] ./prog111.sh -i 1.txt -o 2.txt -p capital -C -n
[avobrezkova@avobrezkova:~/work/os/lab11] cat 2.txt
1:Hello, my name is Nastya.
2:Hello, my name is Dasha.
3:Hello, my name is Kseniya.
4:Hello, my name is Anna.
[avobrezkova@avobrezkova:~/work/os/lab11] ./prog111.sh -i 1.txt -o 2.txt -p capital -n
[avobrezkova@avobrezkova:~/work/os/lab11] cat 2.txt
1:Hello, my name is Nastya.
2:Hello, my name is Dasha.
3:Hello, my name is Kseniya.
[avobrezkova@avobrezkova:~/work/os/lab11] ./prog111.sh -i 1.txt -C -n
Шаблон отсутствует
[avobrezkova@avobrezkova:~/work/os/lab11] ./prog111.sh -o 2.txt -p capital -C -n
Файл отсутствует
[avobrezkova@avobrezkova:~/work/os/lab11]

```

Рис. 2.5: Проверка скрипта

2. Написала 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.

Написала скрипт для выполнения второго задания. Для этого также создала файл. (рис. 2.6)

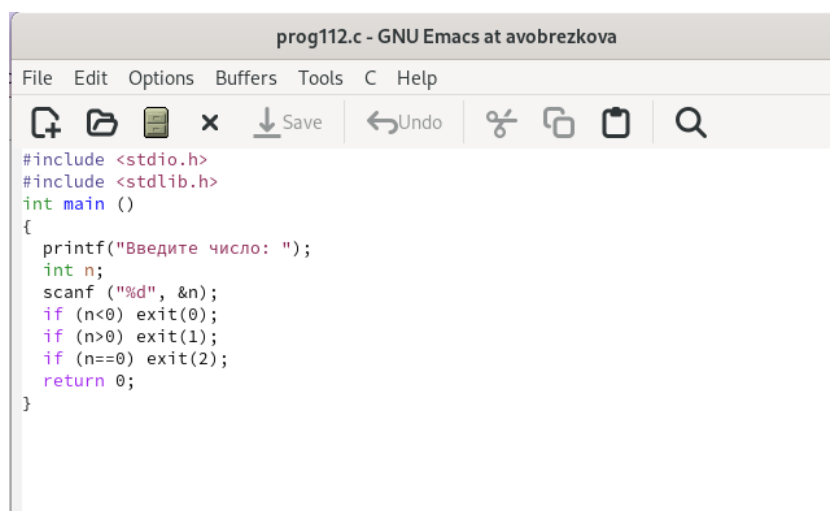
```

[avobrezkova@avobrezkova:~/work/os/lab11] touch prog112.sh prog112.c
[avobrezkova@avobrezkova:~/work/os/lab11] ls
prog111.sh~ prog112.c prog112.sh

```

Рис. 2.6: Создание файлов

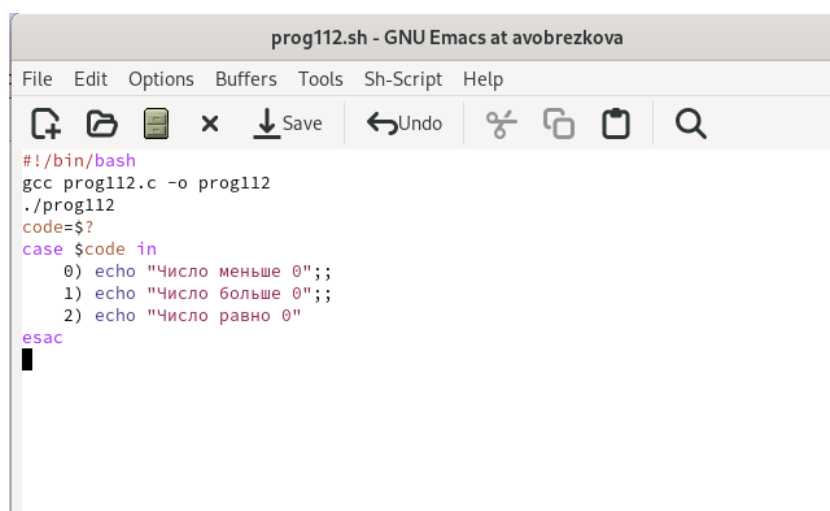
Написала скрипн на языке “C”. (рис. 2.7)

A screenshot of a GNU Emacs editor window titled "prog112.c - GNU Emacs at avobrezkova". The menu bar includes File, Edit, Options, Buffers, Tools, C, and Help. The toolbar shows icons for opening, saving, undo, redo, and search. The code is a C program that prompts the user to enter a number and then checks if it is less than, greater than, or equal to zero, exiting with status codes 0, 1, or 2 respectively.

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{
    printf("Введите число: ");
    int n;
    scanf ("%d", &n);
    if (n<0) exit(0);
    if (n>0) exit(1);
    if (n==0) exit(2);
    return 0;
}
```

Рис. 2.7: Скрипт

Написала скрипн на языке “bush”. (рис. 2.8)

A screenshot of a GNU Emacs editor window titled "prog112.sh - GNU Emacs at avobrezkova". The menu bar includes File, Edit, Options, Buffers, Tools, Sh-Script, and Help. The toolbar is the same as in the previous image. The code is a shell script that compiles the C program, runs it, and then uses a case statement to print messages in Russian based on the exit code.

```
#!/bin/bash
gcc prog112.c -o prog112
./prog112
code=$?
case $code in
    0) echo "Число меньше 0";;
    1) echo "Число больше 0";;
    2) echo "Число равно 0";;
esac
```

Рис. 2.8: Скрипт

Затем дала право на выполнение и проверила. Всё работает корректно. (рис. 2.9)


```

[avobrezkova@avobrezkova:~/work/os/lab11] emacs prog112.c
[avobrezkova@avobrezkova:~/work/os/lab11] emacs prog112.sh
[avobrezkova@avobrezkova:~/work/os/lab11] chmod +x prog112.sh
[avobrezkova@avobrezkova:~/work/os/lab11] ./prog112.sh
Введите число: 0
Число равно 0
[avobrezkova@avobrezkova:~/work/os/lab11] ./prog112.sh
Введите число: 19
Число больше 0
[avobrezkova@avobrezkova:~/work/os/lab11] ./prog112.sh
Введите число: -32
Число меньше 0
[avobrezkova@avobrezkova:~/work/os/lab11]

```

Рис. 2.9: Проверка

3. Написала 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).

Создала новый файл для скрипта и открыла его.(рис. 2.10)

```

[avobrezkova@avobrezkova:~/work/os/lab11] touch prog113.sh
[avobrezkova@avobrezkova:~/work/os/lab11] emacs prog113.sh

```

Рис. 2.10: Создание файла

Написала программу. (рис. 2.11),

Рис. 2.11: Скрипт

Првоерила работу скрипта, дала право на выполнение файла. Создала три файла, а затем удалим их с помощью программы.(рис. 2.12)

Рис. 2.12: Проверка

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).

Создала файл. (рис. 2.13)

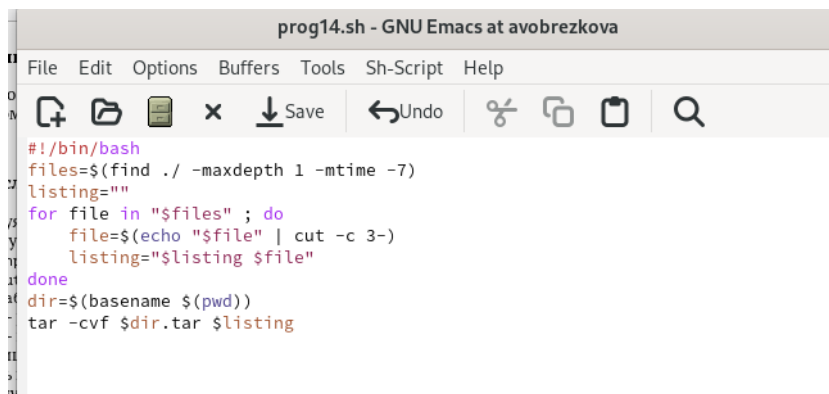
```

avobrezkova@avobrezkova:~/work/os/lab11 touch prog114.sh
avobrezkova@avobrezkova:~/work/os/lab11 emacs prog14.sh

```

Рис. 2.13: Скрипт

Написала программу. (рис. 2.14)



```

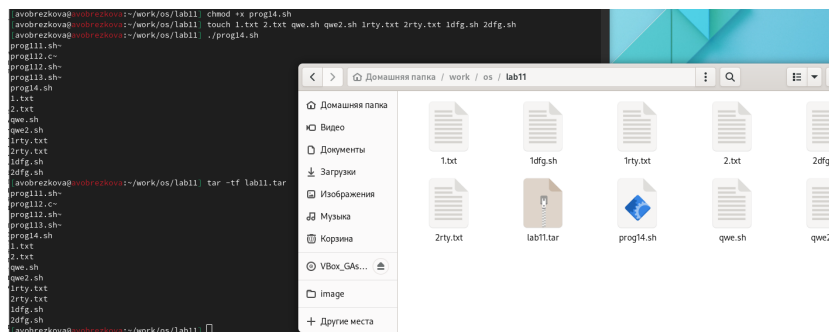
prog14.sh - GNU Emacs at avobrezkova
File Edit Options Buffers Tools Sh-Script Help
[Icons: Open, Save, Undo, Cut, Copy, Paste, Find]

#!/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. 2.14: Проверка

Дала право на выполнение и проверила. Всё работвет верно.(рис. 2.15)



```

avobrezkova@avobrezkova:~/work/os/lab11 chmod +x prog14.sh
avobrezkova@avobrezkova:~/work/os/lab11 touch 1.txt 2.txt qwe.sh qwe2.sh 1rty.txt 2rty.txt 1dfg.sh 2dfg.sh
avobrezkova@avobrezkova:~/work/os/lab11 ./prog14.sh
prog11.sh-
prog12.c-
prog12.sh-
prog13.sh-
prog14.sh
1.txt
2.txt
qwe.sh
qwe2.sh
1rty.txt
2rty.txt
1dfg.sh
2dfg.sh
avobrezkova@avobrezkova:~/work/os/lab11 tar -tf lab11.tar
prog11.sh-
prog12.c-
prog12.sh-
prog13.sh-
prog14.sh
1.txt
2.txt
qwe.sh
qwe2.sh
1rty.txt
2rty.txt
1dfg.sh
2dfg.sh
avobrezkova@avobrezkova:~/work/os/lab11

```

File Manager View: / work / os / lab11

- 1.txt
- 1dfg.sh
- 1rty.txt
- 2.txt
- 2dfg.sh
- 2rty.txt
- lab11.tar
- prog14.sh
- qwe.sh
- qwe2.sh

Рис. 2.15: Проверка

3 Выводы

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.

4 Ответы на контрольные вопросы

1. Команда `getopts` осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]` Флаги – это опции командной строки, обычно помеченные знаком минус; Например, для команды `ls` флагом может являться `-F`. Строка опций `option-string` – это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, то она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.
2. При перечислении имён файлов текущего каталога можно использовать следующие символы:
 - `-` – соответствует произвольной, в том числе и пустой строке;
 - `?` – соответствует любому одинарному символу;

- `[c1-c2]` – соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. Например,
 - `echo *` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`;
 - `ls *.c` – выведет все файлы с последними двумя символами, совпадающими с `.c`.
 - `echo prog.?` – выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.`.
 - `[a-z]*` – соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.
3. Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути, являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.
4. Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает

данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

5. Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования `bash`: это команда `true`, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда `false`, которая всегда возвращает код завершения, не равный нулю (т.е. ложь). Примеры бесконечных циклов: `while true do echo hello andy done` `until false do echo hello mike done`
6. Строка `if test -f mans/i.s, mans/i.s` является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).
7. Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.