

# **Отчет по лабораторной работе №11**

*дисциплина: Операционные системы*

Егорова Юлия Владимировна

# Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Контрольные вопросы	12
4	Выводы	15

## Список иллюстраций

2.1	Создание файла . . . . .	5
2.2	Скрипт №1 . . . . .	6
2.3	Содержимое a1.txt . . . . .	6
2.4	Содержимое a2.txt . . . . .	6
2.5	Проверка работы программы. . . . .	6
2.6	Проверка работы программы. . . . .	7
2.7	Создание файлов. . . . .	7
2.8	Скрипт файла chslo.c. . . . .	8
2.9	Скрипт файла chslo.sh. . . . .	8
2.10	Проверка работы программы. . . . .	8
2.11	Создание файла. . . . .	9
2.12	Скрипт №3 . . . . .	9
2.13	Проверка работы скрипта №3 . . . . .	9
2.14	Проверка работы скрипта №3 . . . . .	10
2.15	Создание файла. . . . .	10
2.16	Скрипт №4 . . . . .	10
2.17	Проверка работы скрипта №4 . . . . .	11
2.18	Проверка работы скрипта №4 . . . . .	11

# 1 Цель работы

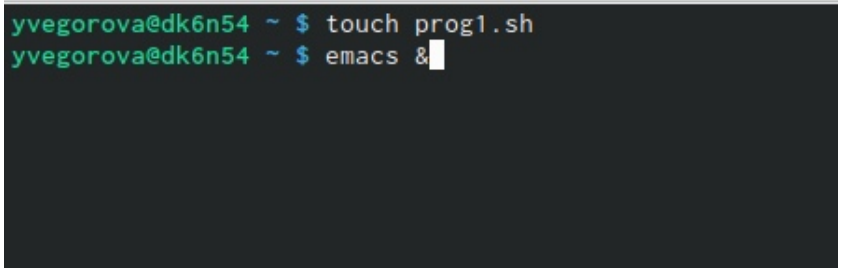
Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 2 Выполнение лабораторной работы

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

- iinputfile — прочитать данные из указанного файла;
- ooutputfile - вывести данные в указанный файл;
- ршаблон — указать шаблон для поиска;
- C — различать большие и малые буквы;–
- n — выдавать номера строк.

Затем нашла в указанном файле нужные строки, определяемые ключом `-р`, и создала файл `prog1.sh`, написав соответствующий скрипт:



```
yvegorova@dk6n54 ~ $ touch prog1.sh
yvegorova@dk6n54 ~ $ emacs &
```

Рис. 2.1: Создание файла

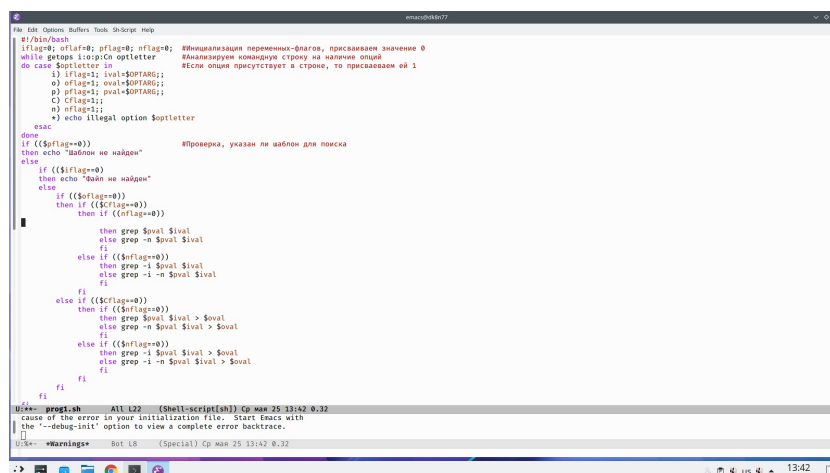


Рис. 2.2: Скрипт №1

2)Проверила работу написанного скрипта, предварительно создав файлы a1.txt и a2.txt. В a1.txt записала любой набор слов, а также дала доступ на исполнение файла:

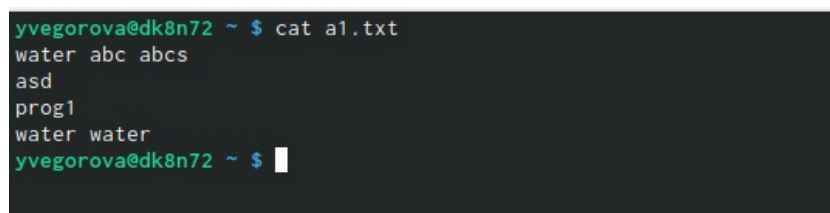


Рис. 2.3: Содержимое a1.txt

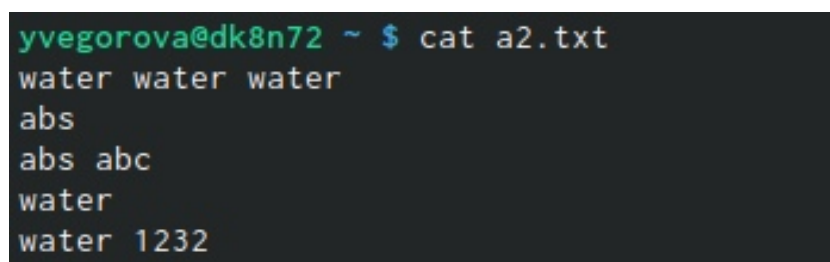


Рис. 2.4: Содержимое a2.txt

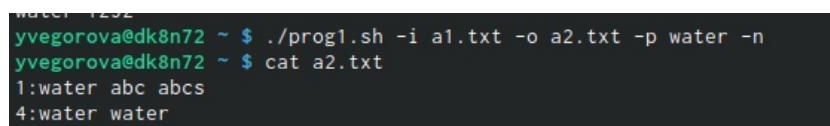


Рис. 2.5: Проверка работы программы.

```
yvegorova@dk8n72 ~ $ ./prog1.sh -i a1.txt -o a2.txt -p water -C -n
yvegorova@dk8n72 ~ $ cat a2.txt
1:water abc abcs
4:water water
```

Рис. 2.6: Проверка работы программы.

3) Написала на языке C 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 о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.

Для этого создала два файла: `chslo.c` и `chslo.sh`

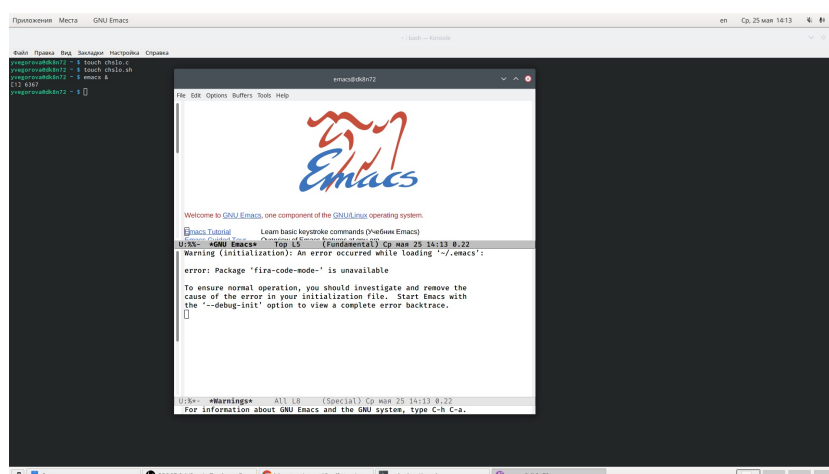


Рис. 2.7: Создание файлов.

Записала соответствующие скрипты:

```
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. 2.8: Скрипт файла chslo.c.

```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
gcc chislo.c -o chislo
./chislo
code=$?
case $code in
    0) echo "Число меньше 0";;
    1) echo "Число больше 0";;
    2) echo "Число равно 0"
esac
```

Рис. 2.9: Скрипт файла chslo.sh.

И проверила работу программы, предварительно открыв доступ на исполнение файла:

```
yvegorova@dk8n72 ~ $ chmod +x chslo.sh
yvegorova@dk8n72 ~ $ ./chslo.sh
Введите число
1
Число больше 0
yvegorova@dk8n72 ~ $
```

Рис. 2.10: Проверка работы программы.



Для этого я создала файл `files.sh` и записала соответствующий скрипт:

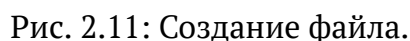
[illegible]

Рис. 2.13: Проверка работы скрипта №3



А затем проверила работу написанного скрипта, предварительно добавив право на исполнение файла и создав отдельный каталог Catalog1 с несколькими файлами.

```
yvegorova@dk8n72 ~/Catalog1 $ ls -l
итого 26
-rw-r--r-- 1 yvegorova studsci   37 мая 25 13:55 a1.txt
-rw-r--r-- 1 yvegorova studsci   33 мая 25 14:05 a2.txt
drwxr-xr-x 2 yvegorova studsci 2048 мая 25 15:38 Catalog1.tar
-rwxr-xr-x 1 yvegorova studsci 16296 мая 25 14:36 chslo
-rw-r--r-- 1 yvegorova studsci   199 мая 25 14:35 chslo.c
-rwxr-xr-x 1 yvegorova studsci   190 мая 25 14:31 chslo.sh
-rwxr-xr-x 1 yvegorova studsci   254 мая 25 14:45 files.sh
-rw-r--r-- 1 yvegorova studsci  1343 мая 25 13:45 '#prog1.sh#'
-rwxr-xr-x 1 yvegorova studsci   209 мая 25 15:35 prog4.sh
yvegorova@dk8n72 ~/Catalog1 $
```

Рис. 2.17: Проверка работы скрипта №4

```
yvegorova@dk8n72 ~ $ cd Catalog1
yvegorova@dk8n72 ~/Catalog1 $ ~/prog4.sh
chslo
chslo.c
chslo.sh
#prog1.sh#
a2.txt
a1.txt
yvegorova@dk8n72 ~/Catalog1 $ tar -tf Catalog1.tar
chslo
chslo.c
chslo.sh
#prog1.sh#
a2.txt
a1.txt
yvegorova@dk8n72 ~/Catalog1 $
```

Рис. 2.18: Проверка работы скрипта №4

### 3 Контрольные вопросы

1). Команда `getopts` осуществляет синтаксический анализ командной строки, выделяя флаги, 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 введённые 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 введённых 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`;

`z]*` – соответствует произвольному имени файла в текущем каталоге, начинающемуся с `z`.

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, неравный нулю (т.е. ложь). Примеры бесконечных циклов: `while true do echo hello andy done` и `until false do echo hello mike done`.

6). Строка `if test -f mans/i.s, mans/i.s` и 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.

## 4 Выводы

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