

# **Лабораторная работа №11**

**Отчет по лабораторной работе**

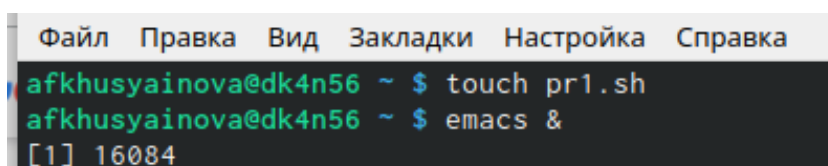
Хусяинова Адиля Фаитовна

# Цель работы

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

# Выполнение лабораторной работы

1. Создадим командный файл pr1.sh, далее откроем его в emacs (рис.1)



```
Файл  Правка  Вид  Закладки  Настройка  Справка
afkhusyainova@dk4n56 ~ $ touch pr1.sh
afkhusyainova@dk4n56 ~ $ emacs &
[1] 16084
```

Рис. 0.1.: Команда man

- Используя команды getopts grep, запишем в файл программу (рис.2), которая анализирует командную строку с ключами: – -inputfile — прочитать данные из указанного файла; – -outputfile — вывести данные в указанный файл; – -ршаблон — указать шаблон для поиска; – -C — различать большие и малые буквы; – -n — выдавать номера строк; а затем ищет в указанном файле нужные строки, определяемые ключом -p.

```
#!/bin/bash
iflag=0; oflag=0; pflag=0; cflag=0; nflag=0;
while getopts :lopnc optionletter
do case $optionletter in
  l) iflag=1; sval=$OPTARG;;
  o) oflag=1; oval=$OPTARG;;
  p) pflag=1; pval=$OPTARG;;
  c) cflag=1;;
  n) nflag=1;;
  *) echo "illegal option $optionletter"
  esac
done
if ((iflag==0))
then echo "badno no badno"
else
if ((iflag==0))
then echo "badno no badno"
else
if ((iflag==0))
then if ((iflag==0))
then if ((iflag==0))
then grep $oval $ival > $oval
else grep -n $oval $ival
fi
else if ((iflag==0))
then grep -i $oval $ival
else grep -i -n $oval $ival
fi
fi
else if ((iflag==0))
then if ((iflag==0))
then if ((iflag==0))
then grep $oval $ival > $oval
else grep -n $oval $ival
fi
else if ((iflag==0))
then grep -i $oval $ival > $oval
else grep -i -n $oval $ival > $oval
fi
fi
fi
fi
```

Рис. 0.2.: Командный файл

- Проверим работу файла, предварительно дав ему права на выполнение (chmod +x \*.sh), и создадим два файла для проверки работы (touch one.txt two.txt) (рис.3). Запускаем файл

```
afkhusyainova@dk4n56 ~ $ chmod +x *.sh
[1]+  Завершён          emacs
afkhusyainova@dk4n56 ~ $ touch one.txt
afkhusyainova@dk4n56 ~ $ touch two.txt
afkhusyainova@dk4n56 ~ $ mcedit one.txt
afkhusyainova@dk4n56 ~ $ mcedit two.txt
```

Рис. 0.3.: Права доступа

```

afkhusyainova@dk4n56 ~ $ ./pr1.sh -i one.txt -o two.txt -p heart -C -n
afkhusyainova@dk4n56 ~ $ cat one.txt
On a dark desert highway, cool wind in my hair
Warm smell of colitas, rising up through the air
Up ahead in the distance, I saw shimmering light
My head grew heavy and my sight grew dim
I had to stop for the night
There she stood in the doorway;
I heard the mission bell
And I was thinking to myself,
'This could be Heaven or this could be Hell'
Then she lit up a candle and she showed me the way
There were voices down the corridor,
I thought I heard them say...afkhusyainova@dk4n56 ~ $ cat two.txt
afkhusyainova@dk4n56 ~ $ cat two.txt
afkhusyainova@dk4n56 ~ $ ./pr1.sh -i one.txt -o two.txt -p heart -C -n
afkhusyainova@dk4n56 ~ $ cat two.txt
afkhusyainova@dk4n56 ~ $ mcedit two.txt

afkhusyainova@dk4n56 ~ $ ./pr1.sh -i one.txt -o two.txt -p heard -C -n
afkhusyainova@dk4n56 ~ $ cat two.txt
7:I heard the mission bell
12:I thought I heard them say...

```

Рис. 0.4.: Запуск файла

2. Создадим два файла для третьего задания (команда `touch pr2.c pr2.sh`) и откроем в `emacs`. Затем напишем 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

```

emacs@dk4n56
File Edit Options Buffers Tools Sh-Script 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;
}

U:~*- pr.c All L13 (C/*l Abbrev) Чт мая 19 15:14 0.64

#!/bin/bash
gcc pr2.c -o pr2
./pr2
code=$?
case $code in
0) echo "Число меньше 0";;
1) echo "Число равно 0";;
2) echo "Число больше 0";;
esac

U:~*- pr2.sh All L9 (Shell-script[sh]) Чт мая 19 15:14 0.64
Mark set

```

Рис. 0.5.: Программа 1

- Проверим работу командного файла, передав ему права на выполнения и запустив его

```

afkhusyainova@dk4n56 ~$ chmod +x pr2.sh
afkhusyainova@dk4n56 ~$ ./pr2.sh
Введите число
0
Число меньше 0
afkhusyainova@dk4n56 ~$

```

Рис. 0.6.: Выполнение

3. Создадим командный файл files.sh и откроем его в emacs. Напишем командный файл, создающий указанное число файлов, пронумерованных

последовательно от 1 до  $\infty$  (например 1.tmp, 2.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).

```

#!/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. 0.7.: Программа 2, командный файл

- Проверим его работу, передав ему права на выполнения и запусив его (команда ./files.sh)

```
#khuysaynov@kde-6 ~ % cd /etc/files.sh
#khuysaynov@kde-6 ~ % ./files.sh -C -t,t 4
7./files.sh: строка 7: ((: : синтаксическая ошибка: ожидается операция (неверный маркер «<>»)
#khuysaynov@kde-6 ~ % smss x
((: 2097
#khuysaynov@kde-6 ~ % ./files.sh -C -t,t 4
./files.sh: строка 7: ((: : синтаксическая ошибка: ожидается операция (неверный маркер «<>»)
#khuysaynov@kde-6 ~ % ./files.sh -c abc -t,t 4
./files.sh: строка 7: ((: : синтаксическая ошибка: ожидается операция (неверный маркер «<>»)
#khuysaynov@kde-6 ~ % ./files.sh -c -t,t 4
./files.sh: строка 7: ((: : синтаксическая ошибка: ожидается операция (неверный маркер «<>»)
#khuysaynov@kde-6 ~ % ls
```

Рис. 0.8.: Выполнение

4. Создадим командный файл `pr4.sh` и откроем его в `emacs`. Напишем командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории. Модифицируем его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду `find`).



```
File Edit Options Buffers Tools Sh-Script Help
#!/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

U:*- pr4.sh All L10 (Shell-script[sh]) Чт мая 19 15:33 2.31
Mark set
```

Рис. 0.9.: Программа 3, командный файл

- Создадим в домашнем каталоге каталог `catalog` и перенесем туда некоторые файлы, измененные в разное время. Дадим командному файлу право на выполнение (`chmod +x pr4.sh`) и запустим его в этом каталоге. Файл работает исправно.



```
afkhusyainova@dk4n56 ~ $ chmod +x pr4.sh
[2]+  Завершён      emacs
afkhusyainova@dk4n56 ~ $ cd catalog
bash: cd: catalog: Нет такого файла или каталога
afkhusyainova@dk4n56 ~ $ mkdir catalog
afkhusyainova@dk4n56 ~ $ cd catalog
afkhusyainova@dk4n56 ~/catalog $ ls
afkhusyainova@dk4n56 ~/catalog $ ls
afkhusyainova@dk4n56 ~/catalog $ ls
afkhusyainova@dk4n56 ~/catalog $ ls
afkhusyainova@dk4n56 ~/catalog $ ls
afkhusyainova@dk4n56 ~/catalog $ ls
file.docx  format.sh  friend.cpp  fri.sh
afkhusyainova@dk4n56 ~/catalog $ ~/pr4.sh
file.docx
format.sh
fri.sh
```

Рис. 0.10.: Выполнение

# Выводы

Я научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# 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`. Например, `1.1 echo` – выведет имена всех файлов

- текущего каталога, что представляет собой простейший аналог команды `ls`; 1.2. `ls.c`–выведет все файлы с последними двумя символами, совпадающими с.с. 1.3. `echoproг.?`–выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.`. 1.4.`[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, неравный 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 mani.s/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.