

Лабораторная работа №11

Отчёт к лабораторной работе

Зайцева Анна Дмитриевна

Table of Contents

Цель работы

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

Задание

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами:
 - `-i`inputfile —прочитать данные из указанного файла;
 - `-o`outputfile —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. 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 о 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 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).
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`).

Выполнение лабораторной работы

1. Я открыла `emacs` (команда: `emacs`) (Рис. [-@fig:001]):

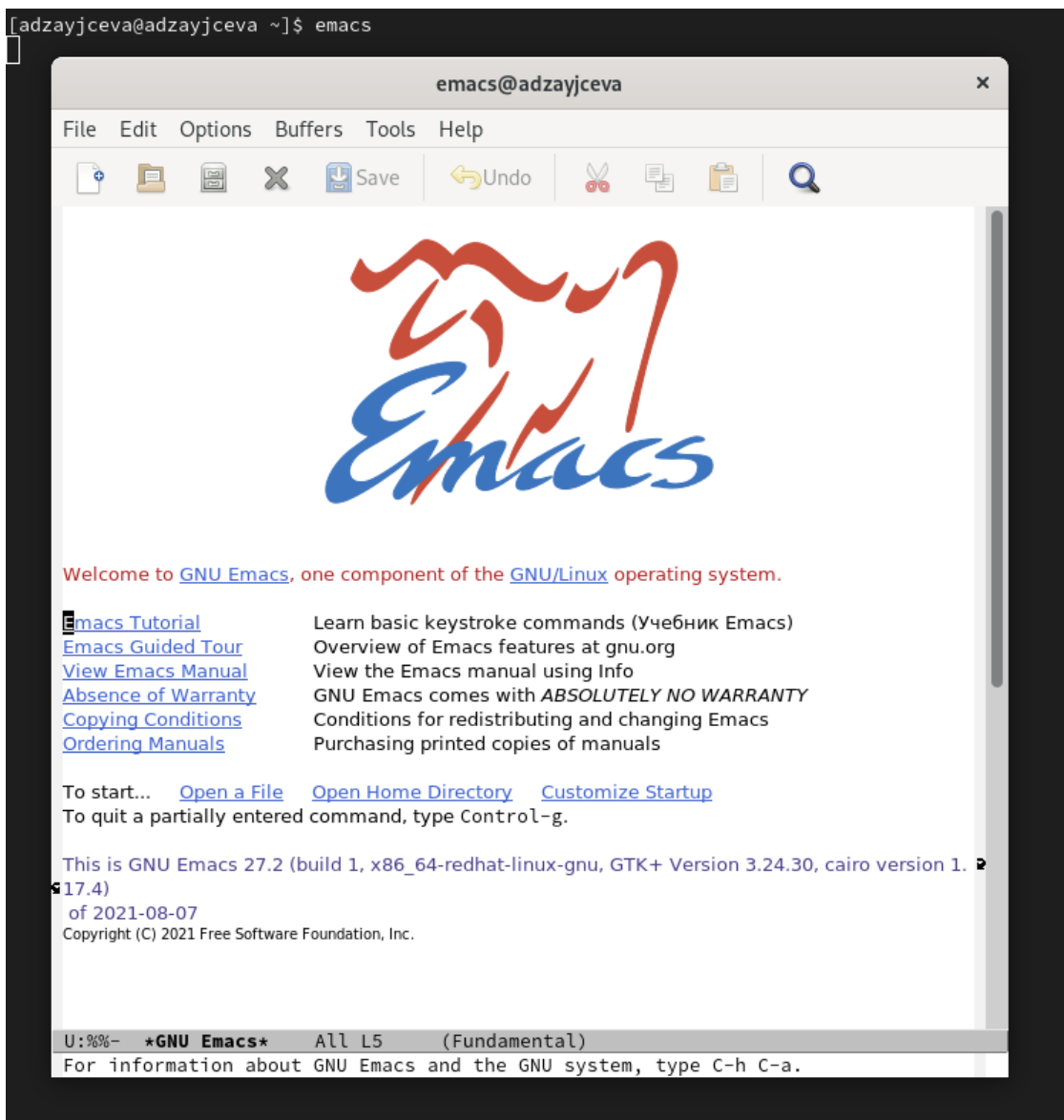


Рис. 1

2. Создала файл `pr1.sh` с помощью комбинации `Ctrl-x Ctrl-f` (`C-x C-f`). Написала скрипт, 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` 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 `-р`. (Рис. [-@fig:002]):

```
#!/bin/bash
iflag=0; oflag=0; Cflag=0; nflag=0; #инициализация пе
while getopts i:o:p:Cn optletter #анализируем командн
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 $optletter
esac
done
if (($plag==0)) #проверяем, указан ли шаблон для поиси
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
    fi
fi
```

U:--- **pr1.sh** Top L27 (Shell-script[sh])

Beginning of buffer

Рис. 2

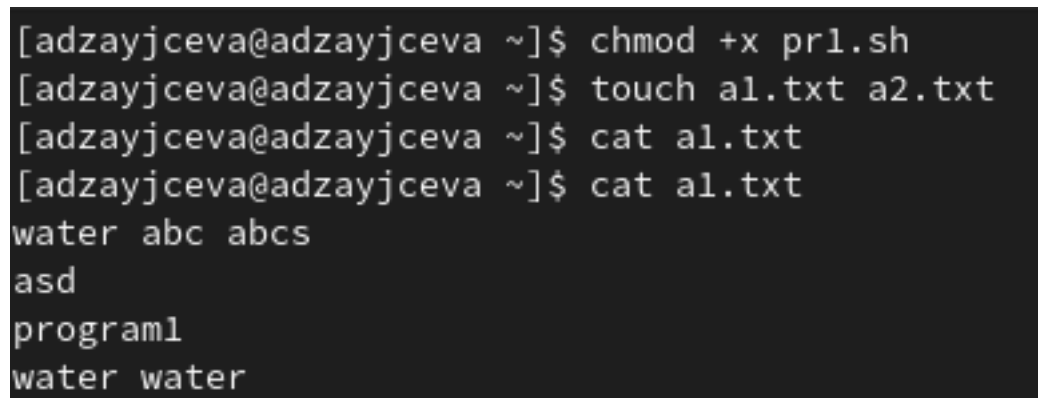
```
#!/bin/bash
iflag=0; oflag=0; pflag=0; Cflag=0; nflag=0; #инициализация переменных-флагов
while getopts i:o:p:Cn optletter #анализируем командную строку на наличие опций
do case $optletter in
    i)iflag=1; ival=$OPTARG;;
    o)oflag=1; oval=$OPTARG;;
    p)pflag=1; pval=$OPTARG;;
    C)Cflag=1;;
    n)nflag=1;;
    *)echo illegal option $optletter
esac
```

```

done
if (($pflag==0)) #проверяем, указан ли шаблон для поиска
then echo "Шаблон не найден"
else
    if (($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

```

Добавила право на исполнение файла (команда: *chmod +x pr1.sh*) и создала 2 файла, которые необходимы для выполнения программы (команда: *touch a1.txt a2.txt*) (Рис. [-@fig:003]):



```

[adzayjceva@adzayjceva ~]$ chmod +x pr1.sh
[adzayjceva@adzayjceva ~]$ touch a1.txt a2.txt
[adzayjceva@adzayjceva ~]$ cat a1.txt
[adzayjceva@adzayjceva ~]$ cat a1.txt
water abc abcs
asd
program1
water water

```

Рис. 3

Скрипт работает корректно (Рис. [-@fig:004]):

```
[adzayjceva@adzayjceva ~]$ ./pr1.sh -i a1.txt -o a2.txt -p water -n
[adzayjceva@adzayjceva ~]$ cat a2.txt
1:water abc abcs
4:water water
[adzayjceva@adzayjceva ~]$ ./pr1.sh -i a1.txt -o a2.txt -p water -C -n
[adzayjceva@adzayjceva ~]$ cat a2.txt
1:water abc abcs
4:water water
[adzayjceva@adzayjceva ~]$ ./pr1.sh -i a1.txt -C -n
Шаблон не найден
[adzayjceva@adzayjceva ~]$ ./pr1.sh -o a2.txt -p water -C -n
Файл не найден
[adzayjceva@adzayjceva ~]$
```

Рис. 4

3. Создала файлы `chislo.sh` и `chislo.c` с помощью комбинации Ctrl-x Ctrl-f (C-x C-f). 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 о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.

chislo.c:

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

int main()
{
    printf ("Enter your number\n");
    int a;
    scanf("%d", &a);
    if (a<0) exit(0);
    if (a>0) exit(1);
    if (a==0) exit(2);
    return 0;
}
```

chislo.sh:

```
#!/bin/bash
gcc chislo.c -o chislo
./chislo
code=$?
case $code in
    0) echo "Number is less than 0";;
    1) echo "Number is more than 0";;
    2) echo "Number is equal to 0"
esac
```

Добавила право на исполнение файла (команда: `chmod +x chislo.sh`) и запустила скрипт несколько раз (команда: `./chislo.sh`). Скрипт работает корректно (Рис. [-@fig:005]):

```

[adzayjceva@adzayjceva ~]$ chmod +x chislo.sh
[adzayjceva@adzayjceva ~]$ ./chislo.sh
Enter your number
4
Number is more than 0
[adzayjceva@adzayjceva ~]$ ./chislo.sh
Enter your number
0
Number is equal to 0
[adzayjceva@adzayjceva ~]$ ./chislo.sh
Enter your number
-34234
Number is less than 0
[adzayiceva@adzayiceva ~]$

```

Рис. 5

4. Создала файл `pr3.sh` с помощью комбинации `Ctrl-x Ctrl-f` (C-x C-f). 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).

```

#!/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

```

Добавила право на исполнение файла (команда: `chmod +x pr3.sh`) и создала 3 файла, удовлетворяющие условию задачи (команда: `./pr3.sh -c abc#.txt 3`), а затем удалила их (команда: `./pr3.sh -r abc#.txt 3`). Скрипт работает корректно (Рис. [-@fig:006]):

```
[adzayceva@adzayceva ~]$ chmod +x pr3.sh
[adzayceva@adzayceva ~]$ ls
123.cpp  a1.txt  adzayceva.github.io  blog  chislo.c  doll.doc  feathers  my_os  pandoc-2.18-linux-arm64.tar.gz  play  pr3.sh  text.txt  Зарплата  Общедоступные
1.sh    a2.txt  australia            catalog_lab07  chislo.sh  elephant.doc  file.txt  pandoc-crossref-linux.tar.gz  pr1.sh  pr3.sh-  ski_places  документы  Изображения  'Рабочий стол'
3.pdf   abc1    backup              chislo  conf.txt  fear.sh      laboratory  pandoc-crossref-linux.tar.gz  pr1.sh-  ski_places  документы  Музыка  Вайфоны
[adzayceva@adzayceva ~]$ ./pr3.sh -c abc#.txt 3
[adzayceva@adzayceva ~]$ ls
123.cpp  a1.txt  abc1.txt  adzayceva.github.io  blog  catalog_lab07  chislo.c  doll.doc  feathers  file.txt  my_os  pandoc-2.18-linux-arm64.tar.gz  play  pr3.sh  pr3.sh-  text.txt  Зарплата  Общедоступные
1.sh    a2.txt  abc2.txt  australia            chislo.sh  elephant.doc  fear.sh    laboratory  pandoc-crossref-linux.tar.gz  pr1.sh  pr3.sh-  ski_places  документы  Изображения  'Рабочий стол'
3.pdf   abc1    abc3.txt  backup              chislo  conf.txt  fear.sh    laboratory  pandoc-crossref-linux.tar.gz  pr1.sh-  ski_places  документы  Музыка  Вайфоны
[adzayceva@adzayceva ~]$ ./pr3.sh -r abc#.txt 3
[adzayceva@adzayceva ~]$ ls
123.cpp  a1.txt  abc1.txt  adzayceva.github.io  blog  catalog_lab07  chislo.c  doll.doc  feathers  file.txt  my_os  pandoc-2.18-linux-arm64.tar.gz  play  pr3.sh  pr3.sh-  text.txt  Зарплата  Общедоступные
1.sh    a2.txt  abc2.txt  australia            chislo.sh  elephant.doc  fear.sh    laboratory  pandoc-crossref-linux.tar.gz  pr1.sh  pr3.sh-  ski_places  документы  Изображения  'Рабочий стол'
3.pdf   abc1    backup              chislo  conf.txt  fear.sh    laboratory  pandoc-crossref-linux.tar.gz  pr1.sh-  ski_places  документы  Музыка  Вайфоны
[adzayceva@adzayceva ~]$
```

Рис. 6

5. Создала файл `pr4.sh` с помощью комбинации `Ctrl-x Ctrl-f` (C-x C-f). Написала 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`).

```
#!/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
```

Добавила право на исполнение файла (команда: `chmod +x pr4.sh`) и создала каталог `Catalog1` с файлами и перешла в него, а затем запустила программу и убедилась в том, что файлы, изменённые более недели назад заархивированы не были (команды: `./pr4.sh` и `tar -tf Catalog1.tar`). Скрипт работает корректно (Рис. [-@fig:007]):

```
[adzayjceva@adzayjceva ~]$ cd Catalog1
[adzayjceva@adzayjceva Catalog1]$ ./pr4.sh
1.sh
3.pdf
a1.txt
a2.txt
chislo
chislo.c
chislo.sh
doll.doc
elephant.doc
fear.sh
pr1.sh
pr3.sh
pr4.sh
[adzayjceva@adzayjceva Catalog1]$ tar -tf Catalog1.tar
1.sh
3.pdf
a1.txt
a2.txt
chislo
chislo.c
chislo.sh
doll.doc
elephant.doc
fear.sh
pr1.sh
pr3.sh
pr4.sh
```

Рис. 7

Ответы на 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 описок 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:
 1. `*` – соответствует произвольной, в том числе и пустой строке;
 2. `?` – соответствует любому одинарному символу;
 3. `[c1-c2]` – соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. Например,
 1. `echo*` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`;
 2. `ls*.c` – выведет все файлы с последними двумя символами, совпадающими с `c`.
 3. `echo prog.?` – выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.`
 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` и `until false do echo hello mike done`.
6. Строка `if test -f mans/i.s` проверяет, существует ли файл `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.

Вывод

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