

Презентация к лабораторной работе №11

Старков Никита Алексеевич

¹RUDN University, Moscow, Russian Federation

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

1)Используя команды `getopts` `grep`, напомним 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.

а затем ищет в указанном файле нужные строки, определяемые ключом -р. Создаем файл `prog11.sh` и пишем соответствующий скрипт

```
nastarkov@dk8n67 ~ $ touch prog11.sh
nastarkov@dk8n67 ~ $ emacs &
```

Figure 1: Создание файла

```
#!/bin/bash
iflag=0; pflag=0; cflag=0; nflag=0;
while getopts i:0:p:c:n 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 $optletter
esac
done
if (($flag==0))
then echo "Шаблон не найден"
else
    if (($iflag==0))
    then echo "0a0n не найден"
    else
        if (($oflag==0))
        then if (($cflag==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
```

Figure 2: Скрипт №1

Проверяем работу написанного скрипта, предварительно создав 2 файла a1.txt и a2.txt. В a1.txt записываем любой набор слов. Также даем доступ на исполнение файла

```
nastarkov@dk8n67 ~ $ touch a1.txt a2.txt
```

Figure 3: Создание файлов

```
nastarkov@dk8n67 ~ $ cat a1.txt  
water abc abcs  
asd  
prog1  
water water
```

Figure 4: Содержимое a1.txt

```
nastarkov@dk8n67 ~ $ chmod +x prog11.sh
[1]+  Завершён      emacs
nastarkov@dk8n67 ~ $ chmod +x prog11.sh
nastarkov@dk8n67 ~ $ ./prog11.sh -i a1.txt -o a2.txt -p water -n
./prog11.sh: строка 2: 0flag=0: команда не найдена
nastarkov@dk8n67 ~ $ emacs &
[1] 12279
nastarkov@dk8n67 ~ $ ./prog11.sh -i a1.txt -o a2.txt -p water -n
[1]+  Завершён      emacs
nastarkov@dk8n67 ~ $ ./prog11.sh -i a1.txt -o a2.txt -p water -n
nastarkov@dk8n67 ~ $ cat a2.txt
1:water abc abcs
4:water water
```

Figure 5: Проверка работы программы

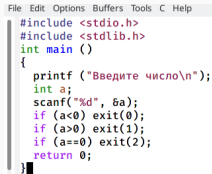
2) Напишем на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.

Создаем два файла: `chslo.c` и `chslo.sh`

```
nastarkov@dk8n67 ~ $ touch chslo.c
nastarkov@dk8n67 ~ $ touch chslo.sh
nastarkov@dk8n67 ~ $ emacs &
```

Figure 6: Создание файлов

Пишем соответствующие скрипты

A screenshot of a code editor window showing a C program. The menu bar at the top includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'C', and 'Help'. The code is as follows:

```
#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;
}
```

Figure 7: Скрипт файла chslo.c

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

Figure 8: Скрипт файла chslo.sh

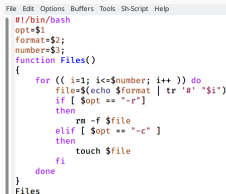
Проверяем работу программы, предварительно открыв доступ на исполнение файла

```
nastarkov@dk8n67 ~ $ chmod +x chslo.sh
nastarkov@dk8n67 ~ $ ./chslo.sh
Введите число
0
Число равно нулю
nastarkov@dk8n67 ~ $ ./chslo.sh
Введите число
5
Число больше 0
nastarkov@dk8n67 ~ $ ./chslo.sh
Введите число
-1
Число меньше 0
```

Figure 9: Проверка работы программы

3) Напишем 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)

Создаем файл files.sh и пишем соответствующий скрипт

A screenshot of a terminal window with a menu bar at the top containing 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Sh-Script', and 'Help'. The terminal displays the following script content:

```
#!/bin/bash
opt=$1
format=$2;
number=$3;
function Files()
{
    for (( i=1; i<=number; i++ )) do
        file=$(echo $format | tr '0' "$i")
        if [ $opt == "-r" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
Files
```

Figure 10: Скрипт №3

Далее проверяем работу написанного скрипта, добавив право на исполнение.

```
nastarkov@bk.ru: ~$ chmod +x files.sh
nastarkov@bk.ru: ~$ ls
a1.txt          f1.txt          format.sh~      public
a2.txt          f2.txt~        GRUB.cfg~       public_html
australia       f2.txt~        lab07.sh~       text.txt
backup          f2.txt~        lab07.sh~       tmp
backup.sh~      f3.txt~        may~            work
backup.sh~      f3.txt~        my_os~          Zeggo
bin             f4.txt~        'NOVEL KATALOG'
blog            f4.txt~        play~           Zaryan
cholo           feathers~       prog11.sh~      Издательства
cholo.d         file2.doc~     prog11.sh~      Музыка
cholo.c~        file2.doc~     prog11.sh~      Оборудование
cholo.sh        file.pdf~      prog11.sh~      'Рабочий стол'
cholo.sh~       files.sh~      prog2.sh~       Unknown
conf.txt        files.sh~      prog2.sh~
course~directory~student~template
equipment        format.sh~     prog11.sh~
```

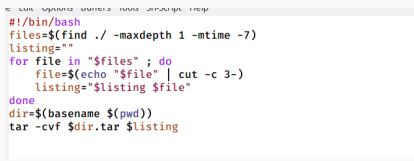
Figure 11: Проверка работы скрипта №3

[illegible]

Figure 12: Проверка работы скрипта №3

4) Напишем 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).

Создаем файл prog4.sh и пишем в нем соответствующий скрипт



```
#!/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
```

Figure 13: Скрипт №4

Далее проверяем работу написанного скрипта, предварительно добавив право на исполнение файла и создав отдельный каталог Catalog1 с несколькими файлами.

```
root@kali:~/katalog1# ls -l katalog1
-rwxr-xr-x 1 root root 4096 Aug 10 10:10 katalog1
-rwxr-xr-x 1 root root 4096 Aug 10 10:10 katalog1
-rwxr-xr-x 1 root root 4096 Aug 10 10:10 katalog1
-rwxr-xr-x 1 root root 4096 Aug 10 10:10 katalog1
-rwxr-xr-x 1 root root 4096 Aug 10 10:10 katalog1
-rwxr-xr-x 1 root root 4096 Aug 10 10:10 katalog1
-rwxr-xr-x 1 root root 4096 Aug 10 10:10 katalog1
-rwxr-xr-x 1 root root 4096 Aug 10 10:10 katalog1
-rwxr-xr-x 1 root root 4096 Aug 10 10:10 katalog1
-rwxr-xr-x 1 root root 4096 Aug 10 10:10 katalog1
```

Figure 14: Проверка работы скрипта №4

Вывод: в ходе выполнения лабораторной работы я изучил основы программирования в оболочке ОС UNIX. Научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.