

Презентация к лабораторной работе №11

Сячинова Ксения Ивановна

Российский университет дружбы народов

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

Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, напомним командный файл, который анализирует командную строку с ключами: (рис. 1)

- `-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.

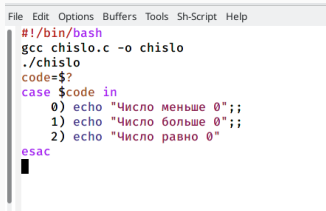
```
1#!/bin/bash
2iflag=0; oflag=0; rflag=0; cflag=0; nflag=0;
3while getopts i:oi:r:C:n optletter
4do case $optletter in
5  i) iflag=1; val=OPTARG;;
6  o) oflag=1; val=OPTARG;;
7  r) rflag=1; val=OPTARG;;
8  C) cflag=1;;
9  n) nflag=1;;
10 *) echo illegal option $optletter
11     exit
12 done
13 if ((iflag==0))
14 then echo "Notice no inputfile"
15 else
16   if ((cflag==0))
17   then echo "Notice no valgrn"
18   else
19     if ((iflag==0))
20     then if ((cflag==0))
21     then if ((iflag==0))
22     then grep $val $val
23     else grep -n $val $val
24     fi
25     else if ((iflag==0))
26     then grep -i $val $val
27     else grep -i -n $val $val
28     fi
29     fi
30     else if ((iflag==0))
31     then if ((iflag==0))
32     then grep $val $val > $val
33     else grep -n $val $val > $val
34     fi
35     else if ((iflag==0))
36     then grep -i $val $val > $val
37     else grep -i -n $val $val > $val
38     fi
39     fi
40     fi
41     fi
42     fi
43 fi
```

После этого, проверяем работу написанного скрипта, используя различные опции. Но перед этим добавим право на исполнение файла “chmod +x os11.sh”. Перед этим создадим два файла, в один из которых запишем текст. С ним и будем проверять выполнение программы. Используем разные команды “./os11.sh -i 1.txt -o 2.txt -p capital -C -n”, “./os11.sh -i 1.txt -o 2.txt -p capital -n” и т.д. (рис. 2)

```
diapachinov@ubuntu:~$ chmod +x os11.sh
diapachinov@ubuntu:~$ cat 1.txt
Moscow is the capital of Russia
London is the capital of Great Britain.
Berlin is the capital of Germany
Paris is the capital of France
diapachinov@ubuntu:~$ ./os11.sh -i 1.txt -o 2.txt -p capital -C -n
diapachinov@ubuntu:~$ cat 2.txt
1: Moscow is the capital of Russia
2: London is the capital of Great Britain.
3: Berlin is the capital of Germany
4: Paris is the CAPITAL of France
diapachinov@ubuntu:~$ ./os11.sh -i 1.txt -o 2.txt -p capital -n
diapachinov@ubuntu:~$ cat 2.txt
1: Moscow is the capital of Russia
2: London is the capital of Great Britain.
3: Berlin is the capital of Germany
diapachinov@ubuntu:~$ ./os11.sh -i 1.txt -o 2.txt -C -n
cat: 2.txt: No such file or directory
diapachinov@ubuntu:~$ ./os11.sh -o 2.txt -p capital -C -n
cat: 2.txt: No such file or directory
diapachinov@ubuntu:~$
```

Figure 2: Проверка работы

2. Напишем на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Эта программа будет завершаться с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку. Командный файл будет вызывать эту программу и, проанализировав с помощью команды “\$?”, выдать сообщение о том, какое число было введено. Создадим два файла: `chislo.c` и `chislo.sh`. (рис. 3). (рис. 4)



The image shows a terminal window with a menu bar at the top containing 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Sh-Script', and 'Help'. The terminal content is a shell script with the following lines: `#!/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`. A cursor is visible on the line following `esac`.

```
#!/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
```

Figure 3: Скрипт 2


```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main()
4 {
5     printf("Введите число\n");
6     int a;
7     scanf ("%d", &a);
8     if (a<0) exit(0);
9     if (a>0) exit(1);
10    if (a==0) exit(2);
11    return 0;
12 }
```

Figure 4: Скрипт 2.1

Проверим выполнение работы, даём разрешение. Всё работает верно.
(рис. 5)

```
kisyachinova@dk6n58 ~ $ chmod +x chislo.sh
kisyachinova@dk6n58 ~ $ ./chislo.sh
Введите число
0
Число равно 0
kisyachinova@dk6n58 ~ $ ./chislo.sh
Введите число
7
Число больше 0
kisyachinova@dk6n58 ~ $ ./chislo.sh
Введите число
-7
Число меньше 0
kisyachinova@dk6n58 ~ $ █
```

Figure 5: Проверка работы

3. Напишем 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). Создаём новый файл: fiels.sh. (рис. 6)

The image shows a terminal window with a title bar that says "report.m". Inside the terminal, a shell script is displayed with line numbers from 1 to 18. The script is a Bash script that takes three arguments: 'opt', 'format', and 'number'. It defines a function 'Files()' that loops from 1 to 'number'. In each iteration, it constructs a filename 'file' using 'echo' and 'tr' to insert a '#' character. It then checks the value of 'opt'. If 'opt' is '-r', it removes the file with 'rm -f \$file'. If 'opt' is '-c', it creates the file with 'touch \$file'. The script ends with a call to 'Files' and a closing brace for the function.

```
1#!/bin/bash
2opt=$1;
3format=$2;
4number=$3;
5function Files()
6{
7    for (( i=1; i<=$number; i++ )) do
8        file=$(echo $format | tr '#' "$i")
9        if [ $opt == "-r" ]
10        then
11            rm -f $file
12        elif [ $opt == "-c" ]
13        then
14            touch $file
15        fi
16    done
17}
18Files
```

Figure 6: Скрипт 3

Проверяем работу, Добавляем право на выполнение. Затем создаём три файла “./files.sh -c abc#.txt 3”, затем удаляем их с помощью команды “./files.sh -r abc#.txt 3”.(рис. 7)

```
kisayachinova@dken58 ~$ chmod *x files.sh
kisayachinova@dken58 ~$ ls
1.txt  australia  chislo  files.sh  may  osl.c~  public_html  work
2.txt  backup     chislo.c  Hello    monthly  osl.sh~  reports  Bape
2.txt  backup.sh  chislo.sh  KseniyaSyachinova.github.io  my_os  play  ski_places  Dony
2.txt  backup.sh~  feathers  lab07.sh  osll.sh  Project  tmp  Zarp
abc1   bin        files.sh  lab07.sh~  osll.sh~  public  tutorial  Hso6
kisayachinova@dken58 ~$ ./files.sh -c abc#.txt 3
kisayachinova@dken58 ~$ ls
1.txt  abc1.txt  backup.sh  chislo.sh  KseniyaSyachinova.github.io  my_os  play  sh
1.txt  abc2.txt  backup.sh~  feathers  lab07.sh  osll.sh  Project  te
2.txt  abc3.txt  bin        files.sh  lab07.sh~  osll.sh~  public  tu
2.txt  australia  chislo  files.sh~  may  osl.c~  public_html  we
abc1   backup    chislo.c  Hello    monthly  osl.sh~  reports  Bn
kisayachinova@dken58 ~$ ./files.sh -r abc#.txt 3
kisayachinova@dken58 ~$ ls
1.txt  australia  chislo  files.sh  may  osl.c~  public_html  work
1.txt  backup     chislo.c  Hello    monthly  osl.sh~  reports  Bape
1.txt  backup.sh  chislo.sh  KseniyaSyachinova.github.io  my_os  play  ski_places  Dony
2.txt  backup.sh~  feathers  lab07.sh  osll.sh  Project  tmp  Zarp
2.txt  bin        files.sh  lab07.sh~  osll.sh~  public  tutorial  Hso6
kisayachinova@dken58 ~$
```

Figure 7: Проверка работы

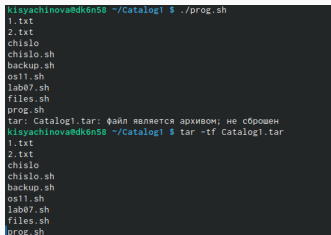
4. Напишем командный файл, который с помощью команды “tar” запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find). Создаю файл prog.sh и пишу скрипт.(рис. 8)



```
*report.md
1#!/bin/bash
2files=$(find ./ -maxdepth 1 -mtime -7)
3listing=""
4for file in "$files" ; do
5    file=$(echo "$file" | cut -c 3-)
6    listing="$listing $file"
7done
8dir=$(basename $(pwd))
9tar -cvf $dir.tar $listing
```

Figure 8: Скрипт 4

Далее для проверки разрешаем право на исполнение и с помощью команд “./prog.sh”, “tar -tf Catalog1.tar” проверяем работу скрипта. Файлы, которые были изменены более недели назад, не были заархивированны. (рис. 9)

A terminal window with a black background and white text. The prompt is 'kisyachinova@dken58 ~/Catalog1 \$'. The first command is './prog.sh', which lists several files: 1.txt, 2.txt, chislo, chislo.sh, backup.sh, osll.sh, lab07.sh, files.sh, and prog.sh. The second command is 'tar: Catalog1.tar: файл является архивом; не сброшен'. The third command is 'kisyachinova@dken58 ~/Catalog1 \$ tar -tf Catalog1.tar', which lists the same files as the first command.

```
kisyachinova@dken58 ~/Catalog1 $ ./prog.sh
1.txt
2.txt
chislo
chislo.sh
backup.sh
osll.sh
lab07.sh
files.sh
prog.sh
tar: Catalog1.tar: файл является архивом; не сброшен
kisyachinova@dken58 ~/Catalog1 $ tar -tf Catalog1.tar
1.txt
2.txt
chislo
chislo.sh
backup.sh
osll.sh
lab07.sh
files.sh
prog.sh
```

Figure 9: Проверка работы

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