

Лабораторная работа № 13.
Программирование в командном
процессоре ОС UNIX. Ветвления и
ЦИКЛЫ

Отчёт

Сергеев Даниил Олегович

Содержание

1	Цель работы	5
2	Задание	6
3	Ход выполнения лабораторной работы	7
3.1	Выполнение упражнений	7
3.2	Ответы на контрольные вопросы	14
4	Вывод	16
	Список литературы	17

Список иллюстраций

3.1	Работа первого скрипта.	9
3.2	Работа первого скрипта с дополнительными ключами.	10
3.3	Результат второго скрипта.	12
3.4	Результат третьего скрипта.	13
3.5	Результат четвертого скрипта.	14

Список таблиц

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов. [1]

2 Задание

Написать командные файлы для задач лабораторной работы

3 Ход выполнения лабораторной работы

3.1 Выполнение упражнений

Создадим каталог lab13 с дополнительными директориями для каждого задания. Приступим к выполнению первой задачи.

Используя команды `getopts` и `grep`, напомним командный файл, который анализирует командную строку с ключами, а затем ищет в указанном файле нужные строки, определяемые ключом. (рис. 3.1-3.2)

- `-i: inputfile` – прочитать данные из указанного файла;
- `-o: outputfile` – вывести данные в указанный файл;
- `-p: template` – указать шаблон для поиска;
- `-C` – различать большие и малые буквы;
- `-n` – выдавать номера строк;

Листинг 3.1. – код программы командного файла первого задания

```
inputfile=$0
outputfile=""
template=""
cflag="-i"
nflag=""
function read {
```

```

        cat $inputfile | grep $cflag $nflag "$template"
    }
while getopts i:o:p:Cn optletter
do case $optletter in
    i)
        if [ -f $OPTARG ]
        then
            inputfile=$OPTARG
        fi
        ;;
    o)
        if [ -f $OPTARG ]
        then
            outputfile=$OPTARG
        fi
        ;;
    p)
        template=$OPTARG
        ;;
    C)
        cflag=""
        ;;
    n)
        nflag="-n"
        ;;
    *)
        echo Illegal option $optletter
    esac
done

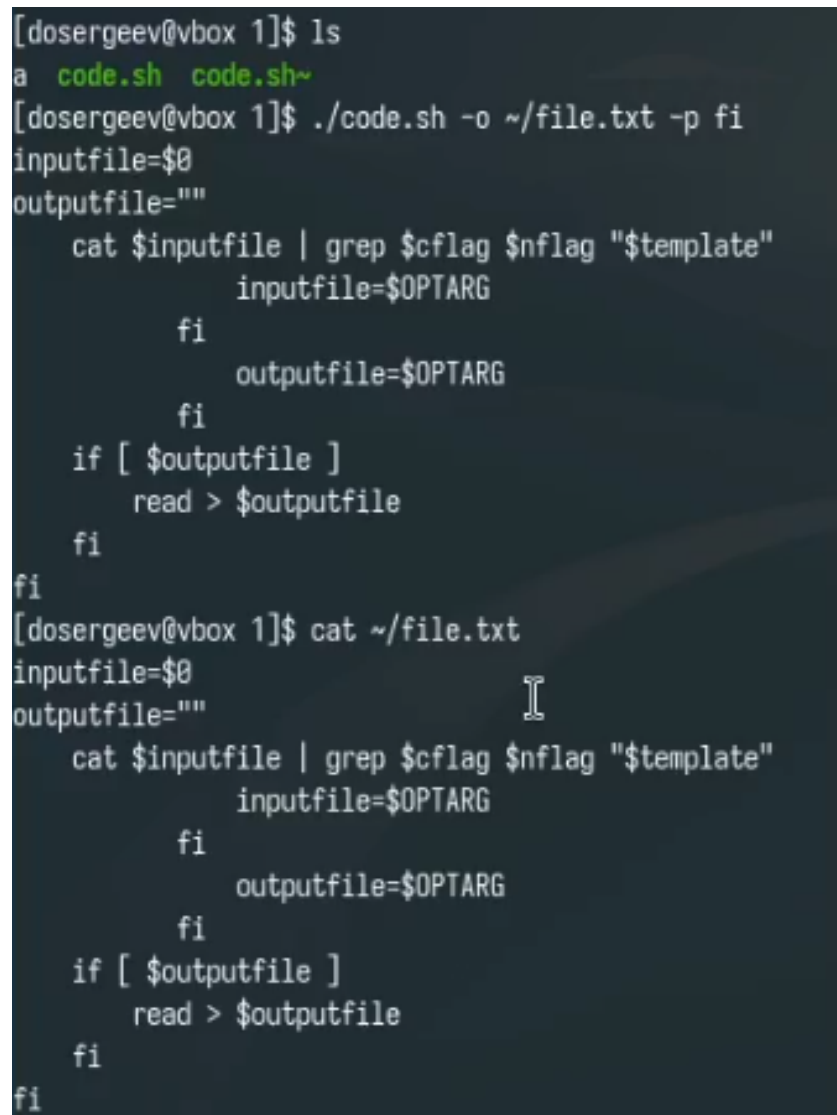
```



```

if [ $template ]
then
    read
    if [ $outputfile ]
    then
        read > $outputfile
    fi
fi

```



```

[dosergeev@vbox 1]$ ls
a code.sh code.sh~
[dosergeev@vbox 1]$ ./code.sh -o ~/file.txt -p fi
inputfile=$0
outputfile=""
    cat $inputfile | grep $cflag $nflag "$template"
        inputfile=$0TARG
        fi
        outputfile=$0TARG
        fi
    if [ $outputfile ]
        read > $outputfile
    fi
fi
[dosergeev@vbox 1]$ cat ~/file.txt
inputfile=$0
outputfile=""
    cat $inputfile | grep $cflag $nflag "$template"
        inputfile=$0TARG
        fi
        outputfile=$0TARG
        fi
    if [ $outputfile ]
        read > $outputfile
    fi
fi

```

Рис. 3.1: Работа первого скрипта.

```

foot
[dosergeev@vbox 1]$ ./code.sh -o ~/file.txt -p optarg -n
12:     if [ -f $OPTARG ]
14:         inputfile=$OPTARG
18:     if [ -f $OPTARG ]
20:         outputfile=$OPTARG
24:     template=$OPTARG
[dosergeev@vbox 1]$ ./code.sh -o ~/file.txt -p optarg -n
C
[dosergeev@vbox 1]$ ./code.sh -o ~/file.txt -p OPTARG -n
C
12:     if [ -f $OPTARG ]
14:         inputfile=$OPTARG
18:     if [ -f $OPTARG ]
20:         outputfile=$OPTARG
24:     template=$OPTARG
[dosergeev@vbox 1]$

```

Рис. 3.2: Работа первого скрипта с дополнительными ключами.

Теперь напомним на языке Си программу, которая определяет, является ли введенное число меньше, больше нуля или равно нулю. Данная программа должна завершаться с помощью команды `exit(n)`, передавая код завершения `n` в оболочку. Также необходимо написать командный файл, который будет анализировать результат с помощью команды `$?`. (рис. 3.3)

Листинг 3.2. – код программы на Си второго задания

```

#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char *argv[]){
    int input;
    if (argc > 1){
        input = atoi(argv[1]);
    } else {
        printf("No args. Input number: ");
    }
}

```

```

        scanf("%d", &input);
    }
    if (input > 0){
        exit(1);
    } else if (input < 0){
        exit(2);
    } else {
        exit(3);
    }
    exit(0);
}

```

Листинг 3.3. – код программы командного файла второго задания

```

./check
case $? in
    1)
        echo Number is higher than zero.
        ;;
    2)
        echo Number is lower than zero.
        ;;
    3)
        echo Number is equal to zero.
        ;;
    *)
        echo Illegal exit code.
esac

```

```
[dosergeev@vbox 2]$ gcc -o check check.c
[dosergeev@vbox 2]$ ./code.sh
No args. Input number: 2
Number is higher than zero.
[dosergeev@vbox 2]$ ./code.sh
No args. Input number: 0
Number is equal to zero.
[dosergeev@vbox 2]$ ./code.sh
No args. Input number: -4
Number is lower than zero.
[dosergeev@vbox 2]$
```

Рис. 3.3: Результат второго скрипта.

Следующий командный файл должен уметь создавать указанное число файлов, пронумерованных от 1 до некоторого N. Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы, если они существуют. (рис. 3.4)

Листинг 3.4. – код программы командного файла третьего задания

```
let counter=0
if (($#>0))
then
    counter=$1
fi
echo $counter
for ((i=1; i<=$counter; i++))
do
    if [ -f $i.tmp ]
    then
        rm $i.tmp
    else
        touch $i.tmp
    fi
done
```

done

```
[dosergeev@vbox 3]$ ./code.sh 2
2
[dosergeev@vbox 3]$ ls
1.tmp 2.tmp code.sh code.sh~
[dosergeev@vbox 3]$ ./code.sh 2
2
[dosergeev@vbox 3]$ ls
code.sh code.sh~
[dosergeev@vbox 3]$ ./code.sh 7
7
[dosergeev@vbox 3]$ ls
1.tmp 3.tmp 5.tmp 7.tmp code.sh~
2.tmp 4.tmp 6.tmp code.sh
[dosergeev@vbox 3]$ ./code.sh 4
4
[dosergeev@vbox 3]$ ls
5.tmp 6.tmp 7.tmp code.sh code.sh~
[dosergeev@vbox 3]$ ./code.sh 7
7
[dosergeev@vbox 3]$ ./code.sh 4
4
[dosergeev@vbox 3]$ ls
code.sh code.sh~
[dosergeev@vbox 3]$
```

Рис. 3.4: Результат третьего скрипта.

Последний командный файл должен с помощью команды `tar` запаковывать в архив все файлы в указанной директории, которые были изменены менее недели тому назад. (рис. 3.5)

Листинг 3.5. – код программы командного файла четвертого задания

```
if (($#!=0))
then
    while (($#>0))
    do
        if [ -d $1 ]
```

```

then
    tar -cf $1/archive.tar $(find $1/* -mtime -7)
    shift
else
    echo Dir $1 is not found
    shift
fi
done
else
    tar -cf $(pwd)/archive.tar $(find * -mtime -7)
fi

```

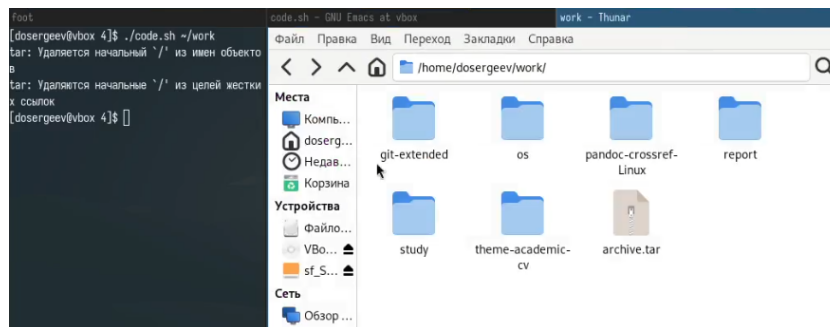


Рис. 3.5: Результат четвертого скрипта.

3.2 Ответы на контрольные вопросы

1. Команда `getopts` считывает аргументы командной строки в поиске ключей и записывает их в заданную переменную `optletter`
2. Перед выполнением команды каждый аргумент команды просматривается в поисках метасимволов, например `*`, `?`, и `[]`, которые считаются как шаблон имён файлов и заменяется именами, соответствующими этому шаблону в алфавитном порядке.
3. Операторы управления действиями:

- Операторы условия: if, else, elif;
- Циклы: for, while;
- Управление выполнением: break, exit, continue;
- Логические операторы: &&(И), ||(ИЛИ), !(НЕ);
- Группировки команд: () - Создание подпроцесса для выполнения команд;

4. Для прерывания цикла используются операторы

- break – для выхода из оператора
- exit – для выхода из программы
- continue – для прерывания итерации цикла

5. Операторы false и true нужны для обозначения успешного и неуспешного завершения выполнения команды

6. 'if test -f man\$s/\$i.\$s' – данная строка проверяет, существует ли объект '\$i.\$s' и является ли он файлом в относительном каталоге 'man\$s/', где '\$i' и '\$s' – подставленные значения переменных i и s соответственно.

7.

- while – цикл с предусловием, пока условие не станет false;
- until – цикл с постусловием, пока условие не станет true;

4 Вывод

В результате выполнения лабораторной работы я изучил основы программирования в оболочке ОС UNIX/Linux и научился писать более сложные командные файлы.

Список литературы

1. Kulyabov. Лабораторная работа № 13. Программирование в командном процессоре ОС UNIX. Ветвления и циклы. https://esystem.rudn.ru/pluginfile.php/2586591/mod_resource/content/5/011-lab_shell_prog_2.pdf; RUDN.