

Отчёт по лабораторной работе №11

**Программирование в командном процессоре ОС UNIX. Ветвления и
циклы**

Грузинова Елизавета Константиновна. НКНбд-02-21

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
5	Контрольные вопросы.	12
6	Выводы	14
	Список литературы	15

Список иллюстраций

4.1	Код командного файла	8
4.2	Текст файла 1.txt	8
4.3	Результаты поиска файла 1.txt в 2.txt	9
4.4	Код командного файла	9
4.5	Создание файло 1.tmp, 2.tmp и т. д.	9
4.6	Удаление этих файлов этой же программой	10
4.7	Код командного файла	10
4.8	Работа в терминале	10
4.9	Результат в виде архива tar	11
4.10	Содержимое этого архива	11

1 Цель работы

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

2 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. Написать 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.
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 `tag` 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`).

3 Теоретическое введение

Циклы Bash - это циклические конструкции, используемые для итерационного выполнения (перебора) любого заданного количества задач до тех пор, пока не будут выполнены все пункты в указанном списке или же predetermined условия. Циклы в Bash имеют три основных типа. Цикл `for` используется для повторения любого заданного кода для любого количества элементов в заданном списке. Следующий вид циклов в нашем списке - цикл `while`. Конкретно этот цикл действует по заданному условию. То есть он будет выполнять код, заключенный в рамки `DO` и `DONE` пока заданное условие истинно. Как только заданное условие станет ложным, выполнение цикла прекратится. Последний цикл, который мы рассмотрим в этой статье по написанию скриптов - это цикл `until`. Цикл `until` действует прямо противоположно циклу `while`. Цикл `until` также действует по заданному условию. Однако код, заключенный между `DO` и `DONE`, будет выполняться только до тех пор, пока это условие не изменится с ложного на истинное.

[1]

4 Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-iinputfile` — прочитать данные из указанного файла; `-ooutputfile` — вывести данные в указанный файл; `-rшаблон` — 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`. (рис. 4.1, 4.2, 4.3)

```

1 #! /bin/bash
2 while getopts o:i:p:cn optletter
3 do
4 case $optletter in
5 i) iflag=1; ival=$OPTARG;;
6 o) oflag=1; oval=$OPTARG;;
7 p) pflag=1; pval=$OPTARG;;
8 c) cflag=1;;
9 n) nflag=1;;
10 *) echo Illegal option $optletter
11 esac
12 done
13
14 if ! test $cflag
15 then
16 cf=-i
17 fi
18 if test $nflag
19 then
20 nf=-n
21 fi
22
23
24 grep $cf $nf $pval $ival >> $oval

```

Рис. 4.1: Код командного файла

- 1 Изучить основы программирования в оболочке ОС UNIX.
- 2 Научится писать более
- 3 сложные командные файлы с
- 4 использованием логических
- 5 управляющих конструкций и циклов

Рис. 4.2: Текст файла 1.txt


```
1 1:Изучить основы программирования в оболочке ОС UNIX.  
2 2:Научиться писать более  
3 4:использованием логических
```

Рис. 4.3: Результаты поиска файла 1.txt в 2.txt

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). (рис. 4.4, 4.5, 4.6)

```
#!/bin/bash  
for((i=1; i<=$*; i++))  
do  
if test -f "$i".tmp  
then rm "$i".tmp  
else touch "$i".tmp  
fi done
```

Рис. 4.4: Код командного файла

```
[ekguzinova@fedora programms]$ bash prog3.sh 8  
[ekguzinova@fedora programms]$ ls  
1.tmp 2.tmp 4.tmp 7.tmp prog1.sh prog2.ch prog4.sh  
1.txt 2.txt 5.tmp 8.tmp prog2.1.c prog2.sh  
2 3.tmp 6.tmp cprog prog2.1.c~ prog3.sh
```

Рис. 4.5: Создание файло 1.tmp, 2.tmp и т. д.

```
[ekgruzinova@fedora programms]$ bash prog3.sh 8
[ekgruzinova@fedora programms]$ ls
1.txt 2.txt prog1.sh prog2.1.c~ prog2.sh prog4.sh
2      cprog prog2.1.c prog2.ch  prog3.sh
[ekgruzinova@fedora programms]$
```

Рис. 4.6: Удаление этих файлов этой же программой

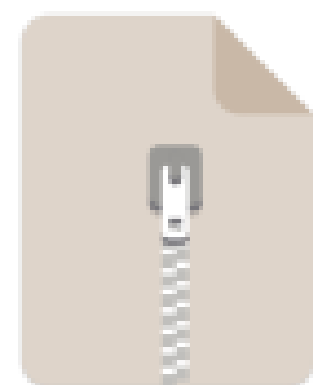
4. Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find). рис. 4.7, 4.8, 4.9, 4.10)

```
#!/bin/bash
:
find $* -mtime -7 -mtime +0 -type f > files.txt
tar -cf files.txt.tar -T files.txt
```

Рис. 4.7: Код командного файла

```
[ekgruzinova@fedora programms]$ bash prog4.sh /home/ekgruzinova
tar: Удаляется начальный '/' из имен объектов
tar: Удаляются начальные '/' из целей жестких ссылок
```

Рис. 4.8: Работа в терминале



`files.txt.tar`

Рис. 4.9: Результат в виде архива tar

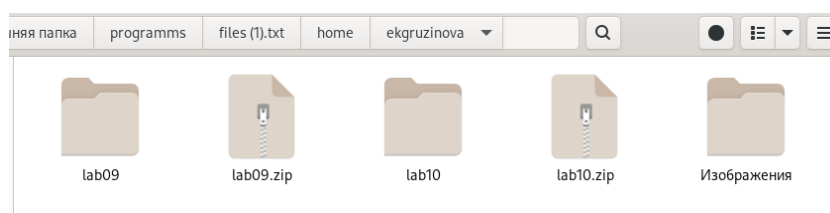


Рис. 4.10: Содержимое этого архива

5 Контрольные вопросы.

1. Каково предназначение команды `getopts`?

`Getopts` – это POSIX-совместимая встроенная функция `bash`, которая принимает короткие аргументы, такие как `-h`, `-v`, `-b` и т.д.

2. Какое отношение метасимволы имеют к генерации имён файлов?

После всех подстановок в каждом слове команды ищутся символы `*`, `?`, и `[`. Если находится хотя бы один из них, то это слово рассматривается как шаблон имен файлов и заменяется именами файлов, удовлетворяющих данному шаблону (в алфавитном порядке). Если ни одно имя файла не удовлетворяет шаблону, то он остается неизменным. Значения указанных символов:

- любая строка, включая и пустую `?` один любой символ `[...]` любой из указанных между ними символов. Пара символов, разделенных знаком `-`, означает любой символ, который находится между ними, включая и их самих. Если первым символом после `[` идет `!`, то указанные символы не должны входить в имя файла

3. Какие операторы управления действиями вы знаете?

Точка с запятой (`;`), амперсанд (`&`), символ доллара со знаком вопроса (`$?`), двойной амперсанд (`&&`) и Знак фунта (`#`).

4. Какие операторы используются для прерывания цикла?

Break, return.

5. Для чего нужны команды false и true?

Если мы выполняем команды, они действительно “ничего не делают”, за исключением того, что команда true возвращает 0, а команда false возвращает 1. Команда false необходима для возвращения кода сбоя, а true позволяет создавать бесконечные циклы.

6. Что означает строка `if test -f mans/i.$s`, встреченная в командном файле?

Если `mans/i.$s` существует и является обычным файлом, то...

7. Объясните различия между конструкциями while и until

While действует до тех пор, пока истинное в условии не станет ложным. Until же, наоборот, действует до тех пор, пока ложное в условии не станет истинным.

6 Выводы

В процессе выполнения лабораторной работы изучила основы программирования в оболочке ОС UNIX и научилась писать более сложные командные файлы с использованием логических управляющих конструкций.

Список литературы

1. Программирование Bash. Часть третья [Электронный ресурс]. URL: <https://blog.yodo.im/programmirovanie-bash-chast-tretya/>.