

Отчёт по лабораторной работе №12

дисциплина: Операционные системы

Быстров Глеб Андреевич

Содержание

1	Цель работы	3
2	Теория	4
3	Задание	6
4	Выполнение лабораторной работы	8
5	Контрольные вопросы	16
6	Выводы	19
7	Библиографический список	20

1 Цель работы

В данной лабораторной работе мне будет необходимо изучить основы программирования в оболочке ОС UNIX/Linux. Будет необходимо научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2 Теория

Bash — популярный командный интерпретатор, используемый в юниксоподобных системах, например, в GNU/Linux. Это программа, которую называют оболочка либо шелл (shell), а само название «bash» является сокращением от «Bourne Again Shell». Интерпретатор Bash принимает ваши команды, передавая их операционной системе. Чтобы осуществлялось взаимодействие с ОС, применяются терминалы (gnome-terminal, nxterm и прочие).

Команда `getopts` является встроенной командой командной оболочки `bash`, предназначенной для разбора параметров сценариев. Она обрабатывает исключительно однобуквенные параметры как с аргументами, так и без них и этого вполне достаточно для передачи сценариям любых входных данных. Базовый синтаксис команды выглядит следующим образом: `$ getopts строка-параметров переменная [набор-параметров]`

В качестве инструмента для архивации данных в Linux используются разные программы. Например архиватор Zip Linux, приобретший большую популярность из-за совместимости с ОС Windows. Но это не стандартная для системы программа. Поэтому хотелось бы осветить команду `tar` Linux — встроенный архиватор.

Find - это одна из наиболее важных и часто используемых утилит системы Linux. Это команда для поиска файлов и каталогов на основе специальных условий. Ее можно использовать в различных обстоятельствах, например, для поиска файлов по разрешениям, владельцам, группам, типу, размеру и другим подобным критериям. Утилита `find` предустановлена по умолчанию во всех Linux дистри-

бутивах, поэтому вам не нужно будет устанавливать никаких дополнительных пакетов. Это очень важная находка для тех, кто хочет использовать командную строку наиболее эффективно. Команда `find` имеет такой синтаксис: `find [папка] [параметры] критерий шаблон [действие]`

3 Задание

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;
 - `-ршаблон` — указать шаблон для поиска;
 - `-С` — различать большие и малые буквы;
 - `-п` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-р`.
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).

4 Выполнение лабораторной работы

- ЗАДАНИЕ 1

1. С помощью команды emacs создал файл task1.sh. (рис. 4.1)

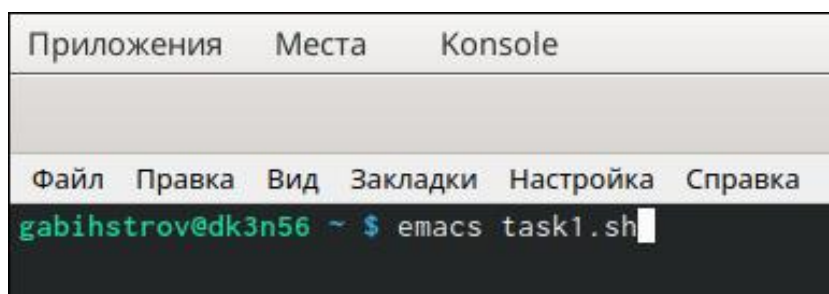


Figure 4.1: Создание файла

2. Используя команды getoptс grep, написал командный файл, который анализирует командную строку с ключами, а затем ищет в указанном файле нужные строки, определяемые ключом -p. (рис. 4.2)


```
#!/bin/bash
i=""
o=""
p=""
C=0
n=0
while getopts "i:o:p:Cn" opt
do
    case $opt in
    i)      i="$OPTARG";;
    o)      o="$OPTARG";;
    p)      p="$OPTARG";;
    C)      C=1;;
    n)      n=1;;
    esac
done
if (($C+$n==2))
then
    grep -i -n "$p" "$i">"$o"
elif (($C+$n==0))
then
    grep "$p" "$i">"$0"
elif (($C==1))
then
    grep -i "$p" "$i">"$o"
elif (($n==1))
then
    grep -n "$p" "$i">"$o"
fi
```

Figure 4.2: Скрипт командного файла

3. С помощью команды emacs создал файл result1.txt. (рис. 4.3)



```
gabihstrov@dk3n56 ~ $ emacs result1.txt
```

Figure 4.3: Создание файла

4. Записал в файл строки с разными буквами. (рис. 4.4)

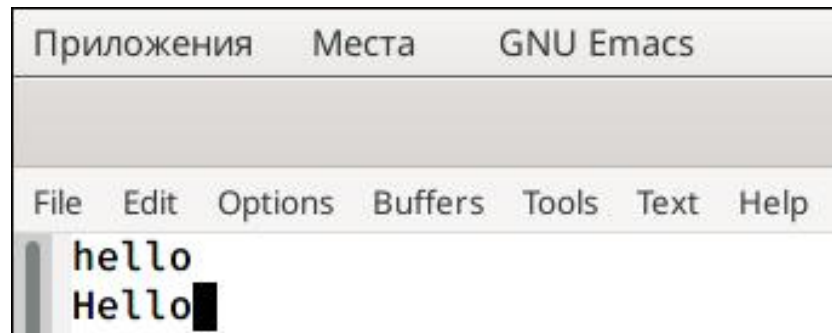


Figure 4.4: Запись строк в файле

5. Вызвал командный файл с помощью `./task1.sh -i result1.txt -o newresult1.txt -p program -C -n`. Код успешно справился с заданием. (рис. 4.5)

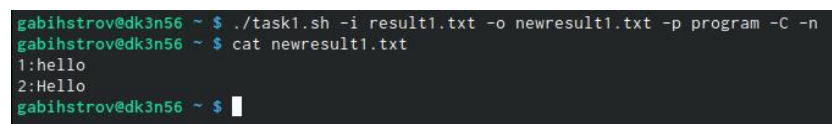


Figure 4.5: Демонстрация работы

- ЗАДАНИЕ 2

6. С помощью команды `emacs` создал файл `task2.cpp`. (рис. 4.6)

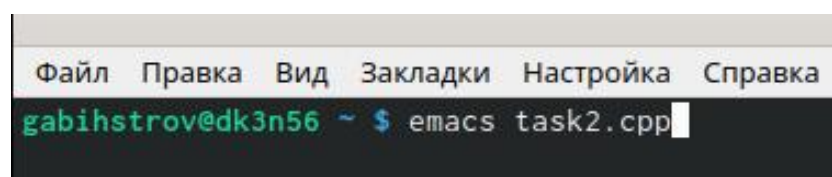


Figure 4.6: Создание файла

7. Написал на языке C++ программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(k)`, передавая информацию в о ко-де завершения в оболочку. Командный файл вызывать эту программу и,

проанализировав с помощью команды \$?, выдаёт сообщение о том, какое число было введено. (рис. 4.7)

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

using namespace std;

int main()
{
    int k;
    cout << "Введите число" << endl;
    cin >> k;
    if (k==0){cout<<"Число =0" << endl;}
    if (k<0){cout<<"Число <0" << endl;}
    if (k>0){cout<<"Число >0" << endl;}
    exit(k);
    return 0;
}
```

Figure 4.7: Скрипт программы

8. С помощью команды emacs создал файл result2.sh. (рис. 4.8)

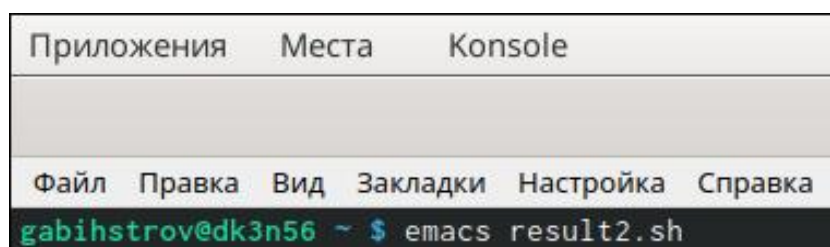
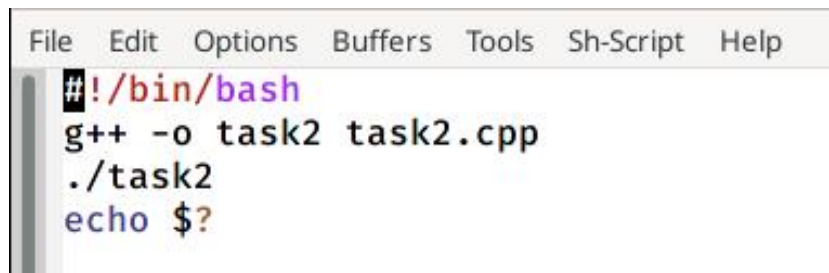


Figure 4.8: Создание файла

9. Написал командный файл. (рис. 4.9)



```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
g++ -o task2 task2.cpp
./task2
echo $?
```

Figure 4.9: Скрипт командного файла

10. Вызвал командный файл с помощью ./result2.sh. Код успешно справился с заданием. (рис. 4.10)



```
gabihstrov@dk3n56 ~ $ ./result2.sh
Введите число
0
Число =0
0
gabihstrov@dk3n56 ~ $ ./result2.sh
Введите число
5
Число >0
5
gabihstrov@dk3n56 ~ $ ./result2.sh
Введите число
-1
Число <0
255
gabihstrov@dk3n56 ~ $
```

Figure 4.10: Демонстрация работы

- ЗАДАНИЕ 3

11. С помощью команды emacs создал файл task3.sh. (рис. 4.11)

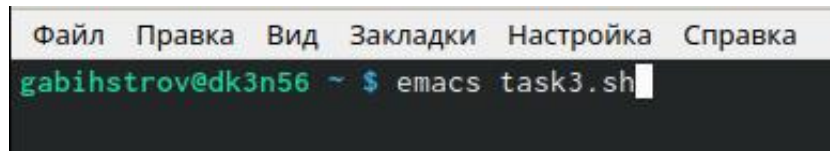


Figure 4.11: Создание файла

12. Написал 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 удаляет все созданные им файлы (если они существуют). (рис. 4.12)

```
#!/bin/bash
n=""
p=""
echo "Сколько создать файлов?"
read n
for ((i=1; i<=n; i=i+1))
do
    touch $i.tmp
done
echo "Каталог:"
ls
echo "Удалить файлы?"
read p
if (p=="y")
then
    for ((i=1; i<=n; i=i+1))
    do
        rm $i.tmp
    done
    echo "Каталог:"
    ls
fi
```

Figure 4.12: Скрипт командного файла

13. Вызвал командный файл с помощью ./task3.sh. Код успешно справился с заданием. (рис. 4.13)

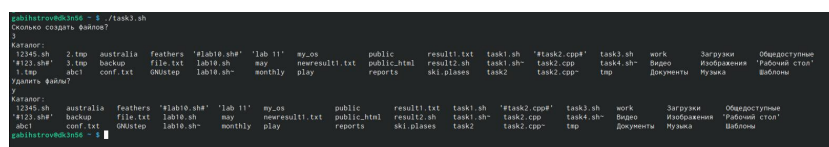


Figure 4.13: Демонстрация работы

• ЗАДАНИЕ 4

14. С помощью команды emacs создал файл task4.sh. (рис. 4.14)

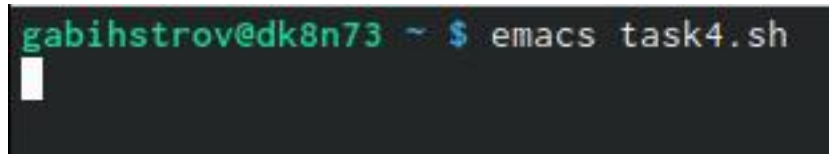


Figure 4.14: Создание файла

15. Написал командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировал его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовал команду find). (рис. 4.15)

```
#!/bin/bash
archive=""
echo "Какой архив создать?"
read archive
directory=""
echo "Введите директорию"
read directory
cd $dir
find . -mtime -7 -type f -print0 |xargs -0 tar -czf ${archive}.tar
ls
```

Figure 4.15: Скрипт командного файла

16. Вызвал командный файл с помощью ./task4.sh. Код успешно справился с заданием. (рис. 4.16)

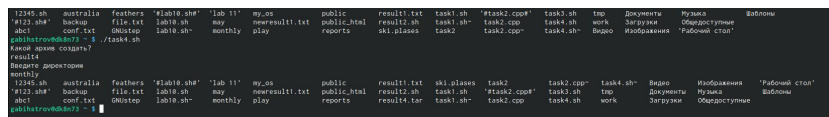


Figure 4.16: Демонстрация работы

5 Контрольные вопросы

1. Каково предназначение команды `getopts`?

Библиографический список ссылка №1

Команда `getopts` является встроенной командой командной оболочки `bash`, предназначенной для разбора параметров сценариев. Она обрабатывает исключительно однобуквенные параметры как с аргументами, так и без них и этого вполне достаточно для передачи сценариям любых входных данных. Базовый синтаксис команды выглядит следующим образом: `$ getopts строка-параметров переменная [набор-параметров]`

2. Какое отношение метасимволы имеют к генерации имён файлов?

Библиографический список ссылка №2

При перечислении имен файлов текущего каталога можно использовать следующие символы: `*` — соответствует произвольной, в том числе и пустой строке; `?` — соответствует любому одному символу; `[c1-c1]` — соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. Например, `echo *` — выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `ls .c` — выведет все файлы с последними двумя символами, равными `.c`. `echo prog.?` — выдаст все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.` `[a-z]` — соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

3. Какие операторы управления действиями вы знаете?

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 Вам 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 дела являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда .

4. Какие операторы используются для прерывания цикла?

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 перестает быть правильным. Пример бесконечного цикла `while`, с прерыванием в момент, когда файл перестает существовать: `while true do if [! -f $file] then break fi sleep 10 done`

5. Для чего нужны команды `false` и `true`?

Библиографический список ссылка №3

Команды интерпретатора `Bash` `true` и `false` Первая всегда возвращает ноль в качестве выходного статуса для индикации успеха, вторая, соответственно, не ноль в качестве выходного статуса для индикации неудачи.

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

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. Строка означает условие существования файла `mans/i.$s`.

7. Объясните различия между конструкциями `while` и `until`.

Библиографический список ссылка №4

Цикл `while` выполняет тело цикла пока условие истинно. Цикл `until` выполняет тело цикла пока условие ложно. Другими словами цикл `until` выполняется до тех пор пока условие не станет истинным.

6 Выводы

В данной лабораторной работе мне успешно удалось изучить основы программирования в оболочке ОС UNIX/Linux. Получилось научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

7 Библиографический список

1. Команда getopts (<https://linux-faq.ru/page/komanda-getopts>)
2. Основные понятия языка shell (<http://www.linuxlib.ru/shell/gl1.htm>)
3. Синтаксис Bash для начинающих. Интерпретатор Bash (<https://otus.ru/nest/post/914/>)
4. Разница между while и until (<https://devtype.blogspot.com/2014/10/Raznitsa-mezhdu-while-i-until.html>)