

Отчёт по лабораторной работе 13

Операционные системы

Гомес Лопес Теофания

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	командный файл, который анализирует командную строку	7
3.2	программа, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю.	8
3.3	командный файл, создающий указанное число файлов	10
3.4	командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории.	11
4	Выводы	13
5	Ответы на контрольные вопросы	14
	Список литературы	18

Список иллюстраций

3.1 код для анализирование командной строки	7
3.2 право на исполнение	8
3.3 Запуск file1	8
3.4 программа на языке с	9
3.5 Командный файл программы на Си	9
3.6 Результаты программы	10
3.7 Командный файл для создания файлов	10
3.8 Создание файлов с помощью командного файла	11
3.9 Создание архива	11
3.10 Результаты кода	12

Список таблиц

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
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю.
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N
4. Написать командный файл, который с помощью команды `tag` запаковывает в архив все файлы в указанной директории.

3 Выполнение лабораторной работы

3.1 командный файл, который анализирует командную строку

Создаю файл file1 и в нем написала код, который анализирует командную строку с ключами -i (прочитать данные из указанного файла), -o (вывести данные в указанный файл), -p (указать шаблон для поиска), -C (различать большие и малые буквы), -n (выдавать номера строк) используя команды getoptс grep:



```
1 while getopt "i:o:p:c:n" opt
2 do
3 case $opt in
4 i)inputfile="$OPTARG";;
5 o)outputfile="$OPTARG";;
6 p)template="$OPTARG";;
7 c)register="$OPTARG";;
8 n)number="";;
9 esac
10 done
11
12 grep -n "$template" "$inputfile.txt" > "$outputfile.txt"
```

Рис. 3.1: код для анализирования командной строки

Далее я установила права на исполнение и запустила файл:

```

teofaniagomeslopes@teofanialopes:~$ touch file1
teofaniagomeslopes@teofanialopes:~$ gedit file1
teofaniagomeslopes@teofanialopes:~$ cp file1 file1.txt
teofaniagomeslopes@teofanialopes:~$ chmod
chmod: falta operando
Tente "chmod --help" para mais informações.
teofaniagomeslopes@teofanialopes:~$ chmod +x file1.txt
teofaniagomeslopes@teofanialopes:~$

```

Рис. 3.2: право на исполнение

```

teofaniagomeslopes@teofanialopes:~$ ./file1.txt -i file1 -o output -p n etconf -
C -n
teofaniagomeslopes@teofanialopes:~$ cat output.txt
1:while getopts "i:o:p:c:n" opt
3:case $opt in
4:i)inputfile="$OPTARG";
8:n)number="";
10:done
12:grep -n "$template" "$inputfile.txt" > "$outputfile.txt"
teofaniagomeslopes@teofanialopes:~$

```

Рис. 3.3: Запуск file1

3.2 программа, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю.

Написала 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:


```

1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int main()
5 {
6     int n;
7     printf("Enter a number: ");
8     scanf("%d", &n);
9     if(n>0)
10    {
11        exit(1);
12    }
13    else if (n==0) {
14        exit(0);}
15    else
16    {
17        exit(2);
18    }
19 }

```

Рис. 3.4: программа на языке с

Далее создала командный файл который вызывает эту программу и, проанализировав с помощью команды \$?, выдает сообщение о том, какое число было введено:

```

1 gcc -o cprog file2.c
2 ./cprog
3
4 case $? in
5 0) echo "равно нулю";;
6 1) echo "больше нуля";;
7 2) echo "меньше нуля";;
8
9 esac

```

Рис. 3.5: Командный файл программы на Си

Создала исполняемый файл и запустила:

```

teofaniagomeslopes@teofanialopes:~$ gedit file2
teofaniagomeslopes@teofanialopes:~$ gedit file2.c
teofaniagomeslopes@teofanialopes:~$ chmod +x command_file.sh
teofaniagomeslopes@teofanialopes:~$ ./command_file.sh
Enter a number: 6
больше нуля
teofaniagomeslopes@teofanialopes:~$

```

Рис. 3.6: Результаты программы

3.3 командный файл, создающий указанное число файлов

Я написала командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до 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):

```

1 for((i=1; i<=$*; i++))
2 do
3 if test -f "$i".tmp
4 then rm "$i".tmp
5 else touch "$i.tmp"
6 fi
7 done

```

Рис. 3.7: Командный файл для создания файлов

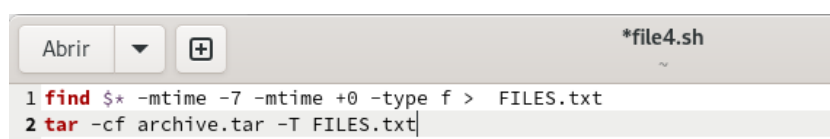
Создала исполняемый файл и запустила:

```
teofaniagomeslopes@teofanialopes:~$ chmod +x Files.sh
teofaniagomeslopes@teofanialopes:~$ ./file3.sh 3
teofaniagomeslopes@teofanialopes:~$ ls
1.tmp          Downloads      lab7.sh~      parentdir2
2.tmp          feathers      LICENSE       parentdir3
3.tmp          file1         may           Pictures
abc1           file1.txt     Milly         play
'Área de trabalho' file2        Modelos       project1.sh
'Arquitetura de computadores' file2.c      monthly      project2.sh
australia      file3.sh     Músicas       project3.sh
backup         file.txt     my_os         project.sh
bin            fun          'New Directory' Público
command_file.sh git-extended  ny_os         reports
conf.txt       helloworld.cpp output.txt     ski.places
cprog         Imagens      '~p'          text.txt
Documentos    '#lab7.sh#' parentdir     Videos
Documents     lab7.sh     parentdir1    work
teofaniagomeslopes@teofanialopes:~$
```

Рис. 3.8: Создание файлов с помощью командного файла

3.4 командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории.

создала командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировала его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).



```
Abrir  ▾  +  *file4.sh
~
1 find $* -mtime -7 -mtime +0 -type f > FILES.txt
2 tar -cf archive.tar -T FILES.txt
```

Рис. 3.9: Создание архива

```

teofaniagomeslopes@teofanialopes:~$ chmod +x file4.sh
teofaniagomeslopes@teofanialopes:~$ ./file4.sh /home/teofaniagomeslopes/work
tar: Removendo "/" inicial dos nomes dos membros
tar: Removendo "/" inicial dos alvos de links fisicos
teofaniagomeslopes@teofanialopes:~$ ls ~/work
arch-pc  github.io  os  study
teofaniagomeslopes@teofanialopes:~$ ls
1.tmp          Downloads    lab7.sh      parentdir2
2.tmp          feathers     lab7.sh~     parentdir3
3.tmp          file1        LICENSE      Pictures
abc1           file1.txt    may          play
archive.tar    file2        Milly        project1.sh
'Área de trabalho' file2.c      Modelos      project2.sh

```

Рис. 3.10: Результаты кода

4 Выводы

При выполнении проделанной работы я научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

5 Ответы на контрольные вопросы

1. Осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable`. Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` – это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Предположим, необходимо распознать командную строку следующего формата: `testprog -infile_in.txt -ofile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае:

```
while getopts o:i:Ltr optletter do
  case $optletter in
    o) iflag=1; oval=$OPTARG;;
    i) iflag=1; ival=$OPTARG;;
    L) Lflag=1;;
    t) tflag=1;;
    r) rflag=1;;
    *) echo "Illegal option $optletter"
  esac
done
```

 Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента (будет равна `file_in.txt` для опции `i` и `file_out.doc` для опции `o`). `OPTIND` является числовым индексом на упомянутый аргумент. Функция `getopts` также понимает переменные типа

массив, следовательно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

2. При перечислении имён файлов текущего каталога можно использовать следующие символы: – соответствует произвольной, в том числе и пустой строке; ? – соответствует любому одинарному символу; [c1-c2] – 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, совпадающими с `.c`. `echo prog.?` – выведет 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. 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 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, являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда. 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. Единствен-

ная функция этой команды заключается в выработке кода завершения.

4. 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 перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.
5. Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования `bash`: это команда `true`, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда `false`, которая всегда возвращает код завершения, не равный нулю (т. е. ложь).
6. Строка `if test -f mans/i.s, mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).
7. Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла

while служебного слова while на until условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла while и оператор цикла until идентичны.

Список литературы