

Отчёт по лабораторной работе 13

Операционные системы

Ерфан Хосейнабади

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	9
3.4	командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории.	11
4	Выводы	12
5	Ответы на контрольные вопросы	13
	Список литературы	17

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	8
3.5 Командный файл программы на Си	9
3.6 Результаты программы	9
3.7 Командный файл для создания файлов	10
3.8 Создание файлов с помощью командного файла	10
3.9 Создание архива	11
3.10 Результаты кода	11

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.

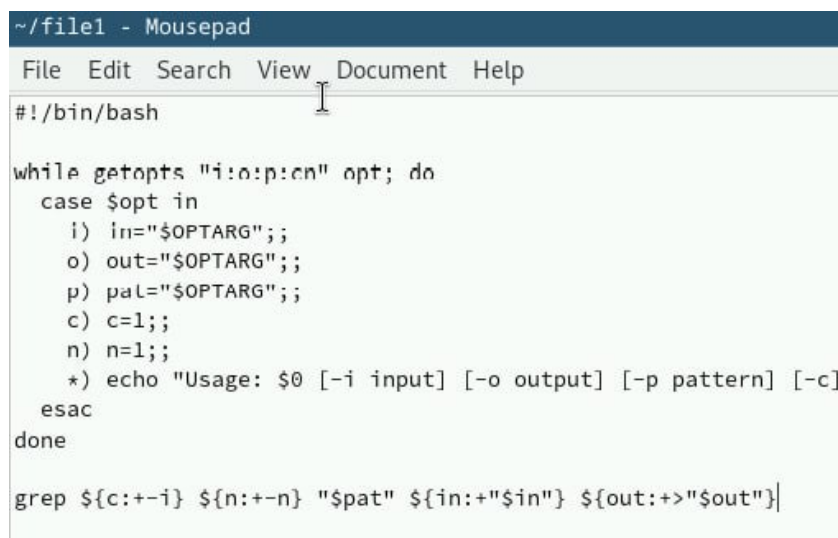
2 Задание

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:



```
~/file1 - Mousepad
File Edit Search View Document Help
#!/bin/bash

while getoptс "i:o:p:cn" opt; do
    case $opt in
        i) in="$OPTARG";;
        o) out="$OPTARG";;
        p) pat="$OPTARG";;
        c) c=1;;
        n) n=1;;
        *) echo "Usage: $0 [-i input] [-o output] [-p pattern] [-c]";;
    esac
done

grep ${c:+-i} ${n:+-n} "$pat" ${in:+"$in"} ${out:+>"$out"}|
```

Рис. 3.1: код для анализирование командной строки

Далее я установила права на исполнение и запустила файл:

```

[erfanhosseinabadi@vbox ~]$ mousepad file1
[erfanhosseinabadi@vbox ~]$ chmod +x file1.sh
chmod: cannot access 'file1.sh': No such file or directory
[erfanhosseinabadi@vbox ~]$ chmod +x file1
[erfanhosseinabadi@vbox ~]$

```

Рис. 3.2: право на исполнение

```

[erfanhosseinabadi@vbox ~]$ ./file1 -i fruit.txt -p "apple" -cn -o result.txt
fruit.txt:1:apple
fruit.txt:3:Apple

```

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:

```

~/file2.c - Mousepad
File Edit Search View Document Help
#include <stdio.h>
#include <stdlib.h>

int main() {
    float num;
    printf("enter a number : ");
    scanf("%f", &num);

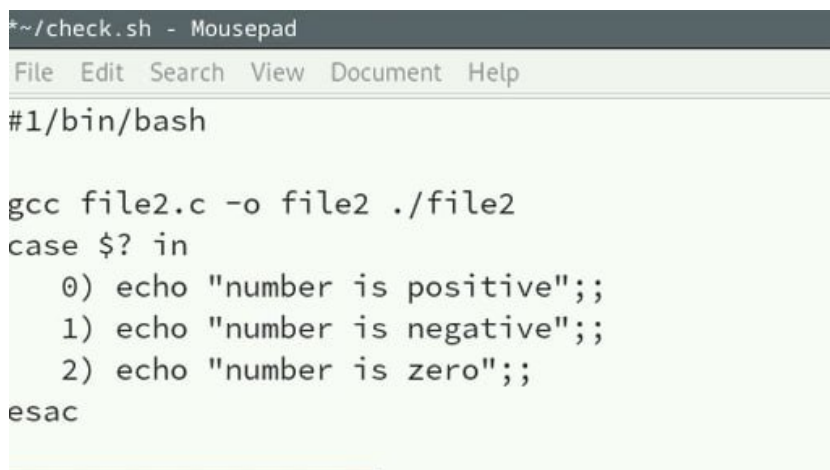
    if(num > 0) exit(0);
    else if(num < 0) exit(1);
    else exit(2);
}

```

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:



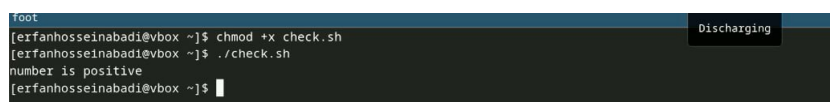
```
*~/check.sh - Mousepad
File Edit Search View Document Help

#!/bin/bash

gcc file2.c -o file2 ./file2
case $? in
    0) echo "number is positive";;
    1) echo "number is negative";;
    2) echo "number is zero";;
esac
```

Рис. 3.5: Командный файл программы на Си

Создала исполняемый файл и запустила:



```
foot
[erfanhosseinabadi@vbox ~]$ chmod +x check.sh
[erfanhosseinabadi@vbox ~]$ ./check.sh
number is positive
[erfanhosseinabadi@vbox ~]$
```

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):

```
~/file3.sh - Mousepad
File Edit Search View Document Help
#!/bin/bash

if [ "$1" = "--clean" ]; then
    rm -f *.tmp
    echo "removed all .tmp files"
fi

if ! [[ "$1" =~ ^[0-9]+$ ]]; then
    echo "Usage: $0 <N> or $0 --clean"
    exit 1
fi

for ((i=1;i<=$1;++i)); do
    touch "$i.tmp"
done
echo "created $1 .tmp files"
```

Рис. 3.7: Командный файл для создания файлов

Создала исполняемый файл и запустила:

```
[erfanhosseinabadi@vbox ~]$ mousepad file3.sh
[erfanhosseinabadi@vbox ~]$ chmod +x file3.sh
[erfanhosseinabadi@vbox ~]$ ./file3.sh 3
created 3 .tmp files
[erfanhosseinabadi@vbox ~]$ ls
1.tmp      Documents  file4.sh  lab11.sh~  project3.sh  test.txt
2.tmp      Downloads  file.cpp  Music      project.sh   text.txt
3.tmp      file1      file.txt  'new directory'  Public      Videos
backup     file1.txt  fruit.txt  pandoc-3.1.3-1-amd64.deb  README.md
backup.tar.gz file2      image.dd  Pictures    result.txt
check.sh   file2.c    '#lab11.sh#'  project1.sh  study
Desktop    file3.sh   lab11.sh  project2.sh  Templates
[erfanhosseinabadi@vbox ~]$ ./file3.sh --clean
removed all .tmp files
Usage: ./file3.sh <N> or ./file3.sh --clean
[erfanhosseinabadi@vbox ~]$ ls
backup     file1      file.cpp  lab11.sh~  project2.sh  study
backup.tar.gz file1.txt  file.txt  Music      project3.sh  Templates
check.sh   file2      fruit.txt  'new directory'  project.sh   text.txt
Desktop    file2.c    image.dd  pandoc-3.1.3-1-amd64.deb  Public      text.txt
Documents  file3.sh   '#lab11.sh#'  Pictures    README.md   Videos
Downloads  file4.sh   lab11.sh  project1.sh  result.txt
[erfanhosseinabadi@vbox ~]$
```

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).

```
find "$1" type f -mtime -7 | tar -czvf "$2" -T -  
echo "created archive $2 with files modified in last 7 days"
```

Рис. 3.9: Создание архива

```
terfanhosseinabadi@vbox ~]$ chmod +x file4.sh  
terfanhosseinabadi@vbox ~]$ ./file4.sh Downloads backup.tar.gz  
find: 'type': No such file or directory  
find: 'f': No such file or directory  
created archive backup.tar.gz with files modified in last 7 days  
terfanhosseinabadi@vbox ~]$
```

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