

Отчёт по лабораторной работе №11

Дисциплина: Операционные системы

Шмырин Михаил Сергеевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	12
4	Ответы на контрольные вопросы	13

Список иллюстраций

2.1	Создание файла	6
2.2	Скрипт	7
2.3	Скрипт	7
2.4	Проверка работы	8
2.5	Создание файла	8
2.6	Скрипт	9
2.7	Скрипт	9
2.8	Скрипт	10
2.9	Скрипт	11
2.10	Проверка работы	11

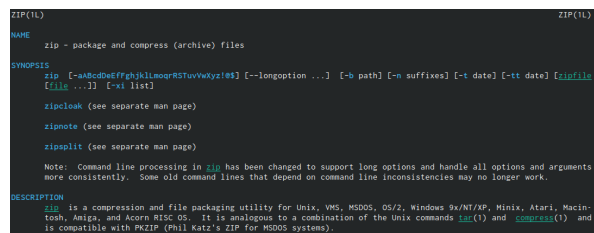
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

- 1) Используя команды `getopts` `grep`, написал командный файл, который анализирует командную строку с ключами:
 - `-iinputfile` — прочитать данные из указанного файла;
 - `-ooutputfile` — вывести данные в указанный файл;
 - `-р` — 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 `-р`. Для данной задачи я создал файл `progra1.sh` (рис. 2.1) и написал соответствующий скрипт (рис. 2.2) , (рис. 2.3)



```
ZIP(1L)                                ZIP(1L)
NAME
    zip - package and compress (archive) files
SYNOPSIS
    zip [-aAdCdDeffghjklmnoRSTuvWxyz@#] [--longoption ...] [-b path] [-n suffixes] [-t date] [-tt date] [zipfile
    [file ...]] [-x list]
    zipcloak (see separate man page)
    zipnote (see separate man page)
    zipsplit (see separate man page)
Note: Command line processing in zip has been changed to support long options and handle all options and arguments
more consistently. Some old command lines that depend on command line inconsistencies may no longer work.
DESCRIPTION
    zip is a compression and file packaging utility for Unix, VMS, MSDOS, OS/2, Windows 9x/NT/XP, Minix, Atari, Macin-
    tosh, Amiga, and Acorn RISC OS. It is analogous to a combination of the Unix commands tar(1) and compress(1) and
    is compatible with PKZIP (Phil Katz's ZIP for MSDOS systems).
```

Рис. 2.1: Создание файла

```
#!/bin/bash
iflag=0; oflag=0; pflag=0; Cflag=0; nflag=0;
while getopts i:o:p:Cn optletter
do case $optletter in
    i) iflag=1; ival=$OPTARG;;
    o) oflag=1; oval=$OPTARG;;
    p) pflag=1; pval=$OPTARG;;
    C) Cflag=1;;
    n) nflag=1;;
    *) echo illegal option $optletter
    esac
done
if ((pflag==0))
then
    echo "Шаблон не найден"
    exit
fi
if ((iflag==0))
then
    echo "Входящий файл не найден"
    exit
fi
if ((oflag==0))
then
    echo "Исходящий файл не найден"
    exit
fi
```

Рис. 2.2: Скрипт

```
if ((nflag==0))
then
    if ((iflag==0))
    then
        grep $pval $ival > $oval
    else
        grep -i $pval $ival > $oval
    fi
else
    if ((iflag==0))
    then
        grep -n $pval $ival > $oval
    else
        grep -i -n $pval $ival > $oval
    fi
fi
```

Рис. 2.3: Скрипт

- 2) Проверил работу написанного скрипта, используя различные опции (например команду `./progra1.sh -i a1.txt -o a2.txt -C -n`), предварительно добавив право на исполнение файла (`chmod +x progra1.sh`) и создав 2 файла, которые необходимы для выполнения программы (`a1.txt`, `a2.txt`). Скрипт работает корректно. (рис. 2.4)

```

bzip2(1)                                General Commands Manual                                bzip2(1)
NAME
  bzip2, bunzip2 - a block-sorting file compressor, v1.0.8
  bzipcat - decompresses files to stdout
  bzip2recover - recovers data from damaged bzip2 files
SYNOPSIS
  bzip2 [ -cdffkqvz123456789 ] [ filename ... ]
  bunzip2 [ -fsvz12 ] [ filename ... ]
  bzipcat [ -s ] [ filename ... ]
  bzip2recover filename
DESCRIPTION
  bzip2 compresses files using the Burrows-Wheeler block sorting text compression algorithm, and Huffman coding.
  Compression is generally considerably better than that achieved by more conventional L277/L278-based compressors,
  and approaches the performance of the PPM family of statistical compressors.

```

Рис. 2.4: Проверка работы

2. 1) Написал 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. Для данной задачи я создал 2 файла: `chslo.c` `chislo.sh` (рис. 2.5) и написал соответствующие скрипты (рис. 2.6) (рис. 2.7)

```

midhs@msshmihrin:~$ emacs &
[1] 4489
midhs@msshmihrin:~$ emacs &
[2] 4667
[1]   Завершён      emacs

```

Рис. 2.5: Создание файла


```
#!/bin/bash
gcc chslo.c -o chslo
./chslo
code=$?
case $code in
    0) echo "Число меньше 0";;
    1) echo "Число больше 0";;
    2) echo "Число равно 0"
esac
```

J:***- **chislo.sh** All L10 (

Рис. 2.6: Скрипт

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    printf("Введите число\n");
    int a;
    scanf("%d", &a);
    if (a<0) exit(0);
    if (a>0) exit(1);
    if (a==0) exit(2);
    return 0;
}
```


J:--- **chslo.c** All L11

Рис. 2.7: Скрипт

- 2) Проверил работу написанных скриптов (команда ./chislo.sh), предварительно добавив право на исполнение файла (chmod +x chislo.sh). Скрипты

работают корректно.

3. 1) Написал командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до (например 1.tmp, 2.tmp, 3.tmp, 4.tmp и т.д.). Число файлов, которые необходимо создать, передается в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют). Для данной задачи я создал файл files.sh (рис. ??) и написал соответствующий скрипт



```
#!/bin/bash
opt=$1;
format=$2;
number=$3;
function Files()
{
    for ((i=1; i<=$number; i++)) do
        file=$(echo $format | tr '#' "$i")
        if [ $opt == "-r" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
Files
```

Рис. 2.8: Скрипт

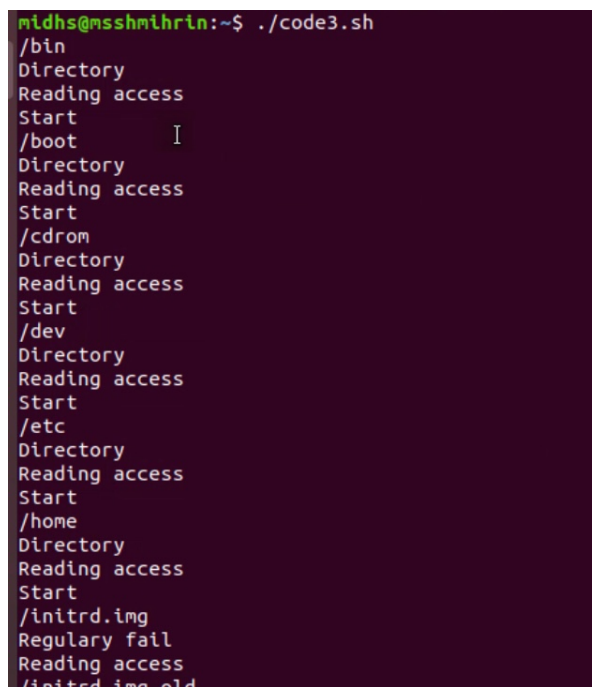
- 2) Далее я проверил работу написанного скрипта (./files.sh), предварительно добавив право на исполнение файла. Сначала я создал три файла, удовлетворяющих условию задач, а потом удалил их. Скрипт работает корректно
4. 1) Написал командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировал его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find). Для данной задачи я создал файл pr4.sh (рис. ??) и написал соответствующий скрипт (рис. 2.9)

Создание файла

```
#!/bin/bash
files=$(find ./ -maxdepth 1 -mtime -7)
listing=""
for file in "$files"; do
    files=$(echo "$file" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

Рис. 2.9: Скрипт

- 2) Далее я проверил работу написанного скрипта, предварительно добавив право на исполнение файла и создав отдельный каталог с несколькими файлами ((рис. ??))



```
midhs@msshmihrin:~$ ./code3.sh
/bin
Directory
Reading access
Start
/boot
Directory
Reading access
Start
/cdrom
Directory
Reading access
Start
/dev
Directory
Reading access
Start
/etc
Directory
Reading access
Start
/home
Directory
Reading access
Start
/initrd.img
Regular file
Reading access
/initrd.img.old
```

Рис. 2.10: Проверка работы

3 Выводы

В ходе выполнения лабораторной работы я изучил основы программирования в оболочке ОС UNIX и научился писать более сложные командные файлы с использованием логических управляющих конструкций циклов.

4 Ответы на контрольные вопросы

1. Команда `getopts` осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg...]`. Флаги – это опции командной строки, обычно помеченные знаком минус; Например, для команды `ls` флагом может являться `-F`. Строка опций `option-string` – это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, то она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Функция `getopts` включает две специальные переменные среды `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента. 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 имен файлов текущего каталога можно использовать следующие символы:
3. `-` соответствует произвольной, в том числе и пустой строке;
4. `?` – соответствует любому одинарному символу;

5. [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.
 - echoproг.?–выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются прог..
 - [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 иwhile. 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 терминомкоманда. Команды ОСUNIX 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.
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 перестает 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, неравный нулю (т.е. ложь). Примеры бесконечных циклов: `while true do echo hello andy done` `until false do echo hello mike done`.
6. Строка `if test -f mani.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 задает 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 задает список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлен тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задает 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.