Отчёт по лабораторной работе №11

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Аскеров Александр Эдуардович

Содержание

Список иллюстраций

1 Цель работы

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

2 Выполнение лабораторной работы

1. Используя команды getopts grep, напишем командный файл, который анализирует командную строку с ключами:

- -iinputfile — прочитать данные из указанного файла,
- -ooutputfile — вывести данные в указанный файл,
- -pшаблон — 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.

Откроем командный файл, напишем программу, сохраним её, дадим файлу право на выполнение, запустим его и проверим наличие файла, в который выводятся данные.

```
[aeaskerov@fedora lab11]$ vi lab111
[aeaskerov@fedora lab11]$ chmod +x lab111
[aeaskerov@fedora lab11]$ ./lab111 -i example.txt -o output.txt -p h -c -n
[aeaskerov@fedora lab11]$ ls
compare.cpp  example.txt  lab111  output.txt  presentation  report
[aeaskerov@fedora lab11]$
```

Рис. 2.1: Открытие командного файла, разрешение на исполнение, запуск, проверка (сформировался ли файл output.txt, куда выводятся данные)

```
while getopts "i:o:p:c:n" opt
do
case $opt in
        i)inputfile="$OPTARG";;
        o)outputfile="$OPTARG";;
        p)shablon="$OPTARG";;
        c)registr="";;
        n)number="";;
esac
done

grep -n "$shablon" "$inputfile" > "$outputfile"
```

Рис. 2.2: Программа для задания один

2. Напишем 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 коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено.

Напишем программу на языке C++.

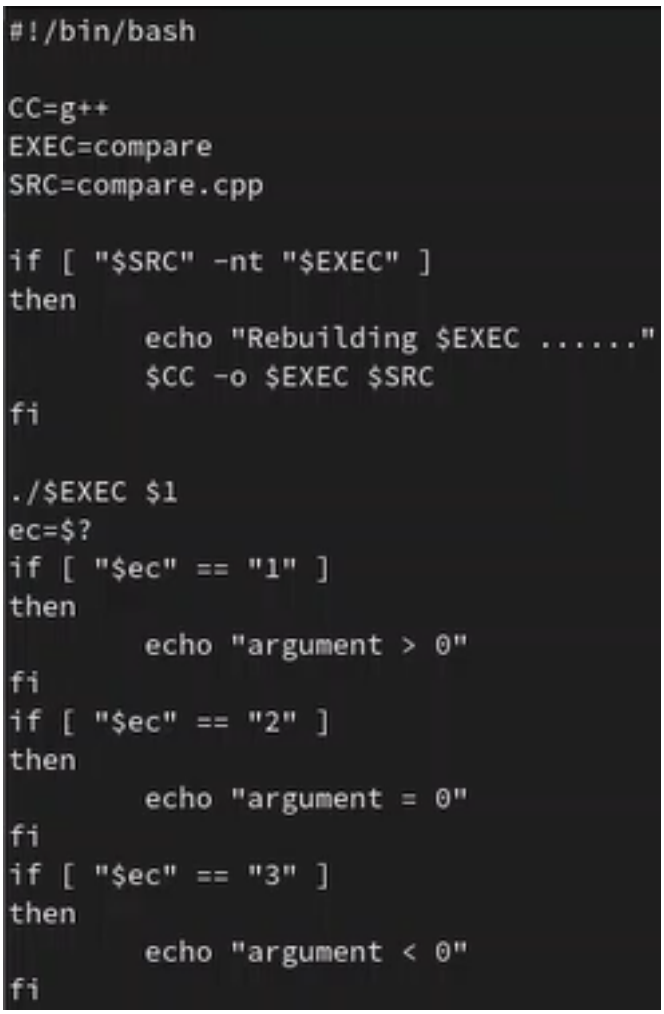
```cpp
#include <iostream>
using namespace std;

int main(int argument, char *arg[]) {
        if (atoi(arg[1]) > 0) {
                exit(1);
        }
        else if (atoi(arg[1]) == 0) {
                exit(2);
        }
        else {
                exit(3);
        }

        return 0;
}
```

Рис. 2.3: Программа на языке C++

Откроем командный файл, напишем программу, сохраним её, дадим файлу право на выполнение и запустим его.

```
[aeaskerov@fedora lab11]$ vi lab112
[aeaskerov@fedora lab11]$ chmod +x lab112
[aeaskerov@fedora lab11]$ ./lab112 10
argument > 0
[aeaskerov@fedora lab11]$ ./lab112 0
argument = 0
[aeaskerov@fedora lab11]$ ./lab112 -1
argument < 0
[aeaskerov@fedora lab11]$
```

Рис. 2.4: Открытие командного файла, разрешение на исполнение, запуск



```
#!/bin/bash

CC=g++
EXEC=compare
SRC=compare.cpp

if [ "$SRC" -nt "$EXEC" ]
then
        echo "Rebuilding $EXEC ......"
        $CC -o $EXEC $SRC
fi

./$EXEC $1
ec=$?
if [ "$ec" == "1" ]
then
        echo "argument > 0"
fi
if [ "$ec" == "2" ]
then
        echo "argument = 0"
fi
if [ "$ec" == "3" ]
then
        echo "argument < 0"
fi
```

Рис. 2.5: Программа для задания два

3. Напишем 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).

Откроем командный файл, напишем программу, сохраним её, дадим файлу право на выполнение, запустим его и проверим, были ли созданы и затем удалены пять файлов формата tmp.

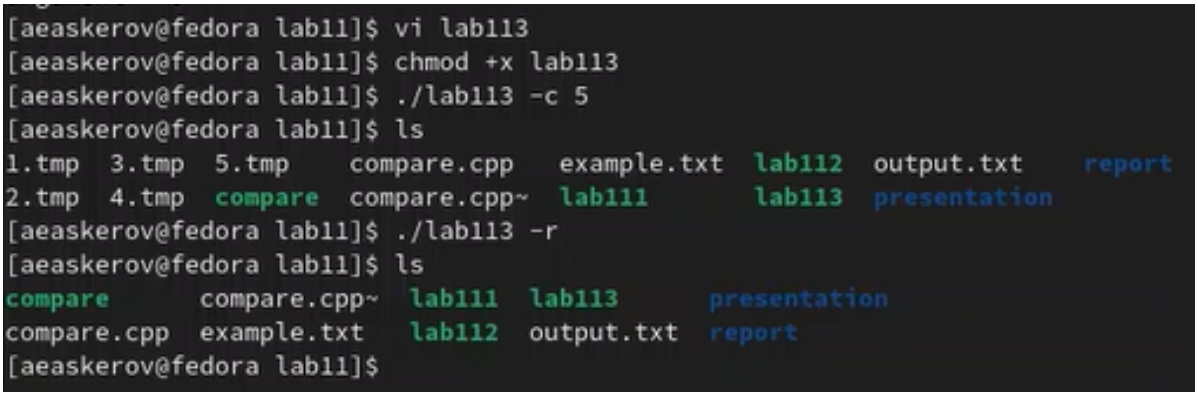


Рис. 2.6: Открытие командного файла, разрешение на исполнение, запуск, проверка

```
#!/bin/bash

while getopts c:r opt
do
case $opt in
        c)n="$OPTARG"; for i in $(seq 1 $n); do touch "$i.tmp"; done;;
        r)for i in $(find -name "*.tmp"); do rm $i; done;;
esac
done
```

Рис. 2.7: Программа для задания три

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

Откроем командный файл, напишем программу, сохраним её, дадим файлу право на выполнение, создадим каталог catalog2, перейдём в него, создадим в нём несколько файлов, запустим командный файл.

```
[aeaskerov@fedora ~]$ touch lab114.sh
[aeaskerov@fedora ~]$ vi lab114.sh
[aeaskerov@fedora ~]$ chmod +x lab114.sh
[aeaskerov@fedora ~]$ mkdir catalog2
[aeaskerov@fedora ~]$ cd catalog2
[aeaskerov@fedora catalog2]$ touch file1.txt file2.txt file3.txt
[aeaskerov@fedora catalog2]$ ls
file1.txt  file2.txt  file3.txt
[aeaskerov@fedora catalog2]$ sudo ~/lab114.sh
file1.txt
file2.txt
file3.txt
[aeaskerov@fedora catalog2]$
```

Рис. 2.8: Открытие командного файла, разрешение на исполнение, создание каталога и файлов в нём, запуск командного файла

Проверим, были ли запакованы файлы в каталоге.

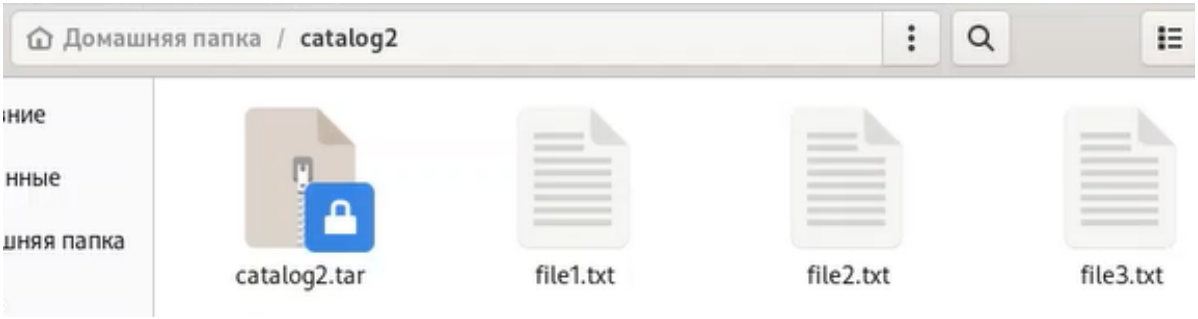


Рис. 2.9: Проверка наличия архива

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

Рис. 2.10: Программа для задания четыре

3 Контрольные вопросы

1. Каково предназначение команды getopts?

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 введённые 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 её в функции не только для синтаксического анализа аргументов функций, но и для анализа введённых пользователем данных.

2. Какое отношение метасимволы имеют к генерации имён файлов?

При перечислении имён файлов текущего каталога можно использовать следующие символы:

1. • соответствует произвольной, в том числе и пустой строке;

2. ? соответствует любому одинарному символу;

3. [c1-c2] соответствует любому символу, лексикографически находящемуся между символами c1 и c2. Например,

   1.1 echo * выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды ls;

   1.2. ls.c выведет все файлы с последними двумя символами, совпадающими с.с.

   1.3. echoprog.? выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются prog.

   1.4.[a-z]* соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

4. 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 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. Единственная функция этой команды заключается в выработке кода завершения.

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 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. Для чего нужны команды false и true?

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

Строка if test -f man$s/$i.$s проверяет, существует ли файл man$s/$i.$s и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернёт нулевое значение (ложь).

7. Объясните различия между конструкциями while и until.

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 не нулевой 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.

4 Выводы

Изучены основы программирования в оболочке ОС UNIX. Приобретён навык написания более сложных командных файлов с использованием логических управляющих конструкций и циклов.