Лабораторная работа №11

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Крутова Екатерина Дмитриевна

Содержание

Список иллюстраций

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 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.

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 ⊠ (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 tar 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).

3 Выполнение лабораторной работы

3.1 Задание 1 (рис. 3.1-3.4)

```
[edkrutova@fedora ~]$ touch c0.sh
[edkrutova@fedora ~]$ emacs&
```

Рис. 3.1: Создание файла и вызов редактора

```
#!/bin/bash

while getopts :i:o:p:cn opt
do

    case "${opt}" in
        i)input=${OPTARG};;
        o)output=${OPTARG};;
        p)mask=${OPTARG};;
        c)c=true;;
        n)n=true;;
    esac
done

if [ $c ]
then
    if [ $n ]
    then
        grep -n -i $mask $input > $output
        exit 0
    else
        grep -i $mask $input > $output
        exit 0
    fi
else
    grep -n $mask $input > $output
    exit 0
fi
#!/bin/bash

while getopts "i:o:p:Cn" opt; do
    case $opt in
        i)
        input_file=$OPTARG
        ;;
        o)
        output_file=$OPTARG
        ;;
```

Рис. 3.2: Текст командного файла к заданию 1 (1)

```
        output_file=$OPTARG
        ;;
        p)
        pattern=$OPTARG
        ;;
        C)
        case_insensitive=1
        ;;
        n)
        print_line_numbers=1
        ;;
        \?)
        echo "Invalid option: -$OPTARG" >&2
        exit 1
        ;;
        :)
        echo "Option -$OPTARG requires an argument." >&2
        exit 1
        ;;
    esac
done

if [ -z "$input_file" ] || [ -z "$output_file" ] || [ -z "$pattern" ]; then
    echo "Usage: $0 -i <input_file> -o <output_file> -p <pattern> [-C] [-n]"
    exit 1
fi

grep $(if [ -n "$case_insensitive" ]; then echo "-i"; fi) $(if [ -n "$print_line_numbers" ]; then
printf %s "-n"; fi) "$pattern" "$input_file" > "$output_file"
```

Рис. 3.3: Текст командного файла к заданию 1 (2)

```
[edkrutova@fedora ~]$ ./c0.sh -i main.txt -o fi.txt -p cout -n
[edkrutova@fedora ~]$ cat fi.txt
24:    cout<<"введите маршрут";
29:            cout<<tr_st[i].nomer<<tr_st[i].fio<<"\n";
33:            cout<<"маршрут не обслуживается";
[edkrutova@fedora ~]$
```

Рис. 3.4: Проверка

3.2 Задание 2 (рис. 3.5-3.9)

```
[edkrutova@fedora ~]$ touch f1.c
[edkrutova@fedora ~]$ emacs &
```

Рис. 3.5: Создание файла и вызов редактора

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
  printf("vvedi chislo: ");
  int a;
  scanf("%d", &a);
  if (a<0) exit(0);
  if (a>0) exit(1);
  if (a==0) exit(2);
  return 0;
}
```

Рис. 3.6: Текст программы на C заданию 2

```
[edkrutova@fedora ~]$ touch c1.sh
[1]+  Завершён        emacs
[edkrutova@fedora ~]$ emacs &
```

Рис. 3.7: Создание файла и вызов редактора

```
#!/bin/bash

gcc f1.c -o f1
./f1
code=$?
case $code in
    0) echo "chislo <0";;
    1) echo "chislo >0";;
    2) echo "chislo =0";;
esac
```

Рис. 3.8: Текст командного файла к заданию 2

```
[edkrutova@fedora ~]$ chmod +x c1.sh
[edkrutova@fedora ~]$ ./c1.sh
vvedi chislo: 3
chislo >0
```

Рис. 3.9: Создание исполняемого файла и проверка

3.3 Задание 3 (рис. 3.10-3.12)

```
[edkrutova@fedora ~]$ touch c2.sh
[edkrutova@fedora ~]$ emacs &
[1] 6731
```

Рис. 3.10: Создание файла и вызов редактора

```
#!/bin/bash

opt=$1;
form=$2;
num=$3;
function files(){
    for ((i=1; i<=$num; i++)) do
        file=$(echo $form | tr '#' "$i")
        if [ $opt == "-r" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
files
```

Рис. 3.11: Текст командного файла к заданию 3

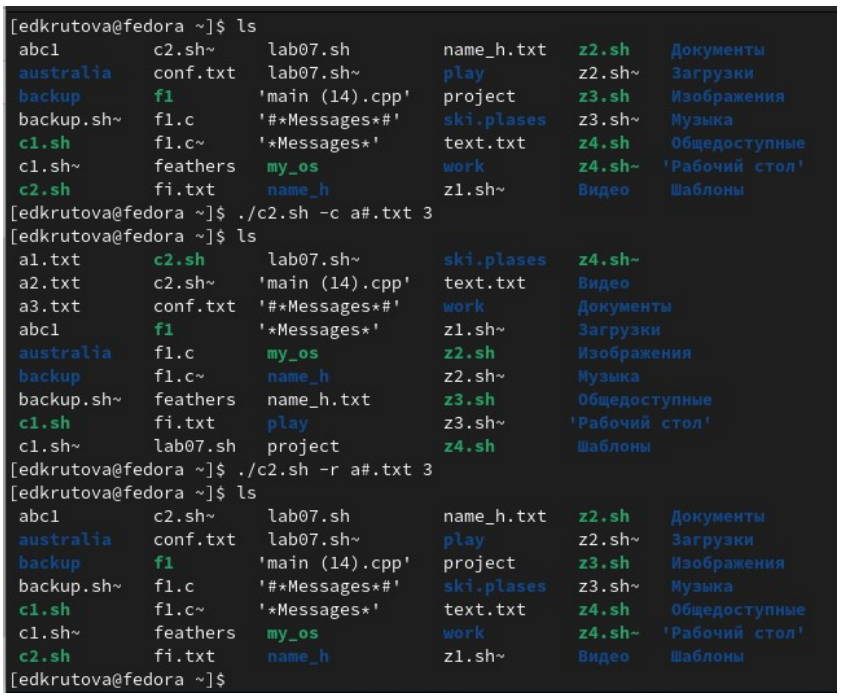

Рис. 3.12: Создание исполняемого файла и проверка

3.4 Задание 4 (рис. 3.13-3.15)

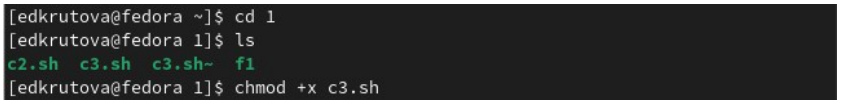

Рис. 3.13: Просмотр содержимого каталога

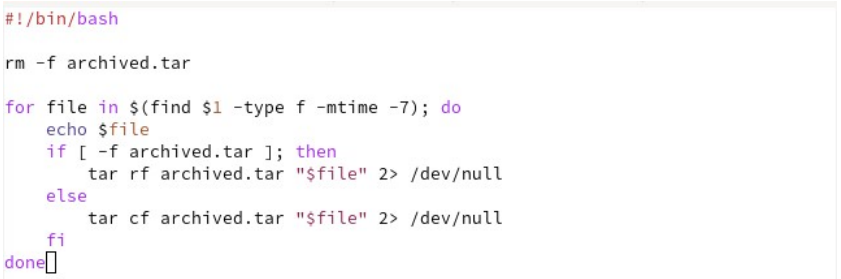

Рис. 3.14: Текст командного файла к заданию 4

Рис. 3.15: Ппроверка

4 Выводы

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

5 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 ...]

2. Какое отношение метасимволы имеют к генерации имён файлов?

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. Какие операторы управления действиями вы знаете?

Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости отрезультатов 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□/i.□проверяет,существуетлифайл□□□s/□.s и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет 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 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.