Отчет по лабораторной работе №10

Программирование в командном процессоре ОС UNIX. Ветвления и циклы.

Сомсиков Даниил Сергеевич

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:

   - -i - inputfile — прочитать данные из указанного файла;
   - -o - outputfile — вывести данные в указанный файл;
   - -p - 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 -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 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 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).

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 Теоретическое введение

3.1 Переменные в языке программирования bash

Командный процессор bash обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем. Пользователь имеет возможность присвоить переменной значение некоторой строки символов.

Например, команда

```
mark=/usr/andy/bin
```

переместит файл afile из текущего каталога в каталог с абсолютным полным именем /usr/andy/bin. Использование значения, присвоенного некоторой переменной, называется подстановкой. Для того чтобы имя переменной не сливалось с символами, которые могут следовать за ним в командной строке, при подстановке в общем случае используется следующая форма записи:

```
${имя переменной}
```

Оболочка bash позволяет работать с массивами. Для создания массива используется команда set с флагом -A. За флагом следует имя переменной, а затем список значений, разделённых пробелами. Например,

```
set -A states Delaware Michigan "New Jersey"
```

3.2 Использование арифметических вычислений. Операторы let и read

Команда let берет два операнда и присваивает их переменной. Положительным моментом команды let можно считать то, что для идентификации переменной ей не нужен знак доллара; вы можете писать команды типа let sum=x+7, и let будет искать переменную x и добавлять к ней 7.

Команда let также расширяет другие выражения let, если они заключены в двойные круглые скобки. Таким способом вы можете создавать довольно сложные выражения. Команда let не ограничена простыми арифметическими выражениями.

Команда read позволяет читать значения переменных со стандартного ввода:

```
echo "Please enter Month and Day of Birth ?"
read mon day trash
```

3.3 Командные файлы и функции

Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным. Далее этот файл можно выполнить по команде:

```
bash командный_файл [аргументы]
```

Чтобы не вводить каждый раз последовательности символов bash, необходимо изменить код защиты этого командного файла, обеспечив

доступ к этому файлу по выполнению. Это может быть сделано с помощью команды

```
chmod +x имя_файла
```

4 Выполнение лабораторной работы

1. Используя команды getopts grep, нужно написать командный файл, который анализирует командную строку с ключами (-i, -o, -p,-c, -n), а затем ищет в указанном файле нужные строки, определяемые ключом -p (рис. 4.1):

```
#!/bin/bash

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 ! test $cflag
        then cf=-i
fi

if test $nflag
        then nf=-n
fi

grep $cf $nf $pval $ival >> $oval
```

Рис. 4.1: Командный файл №1

Создаем один текстовый файл со стихотворением “input.txt” и файл, в который будет записываться результат “output.txt”. Делаем файл “prog10-1.sh” исполняемым и выводим результат (рис. 4.2), (рис. 4.3).

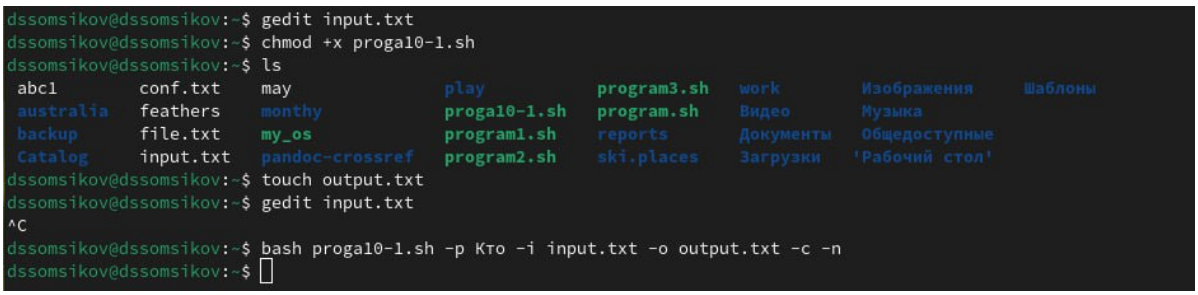

Рис. 4.2: Создание нужных файлов

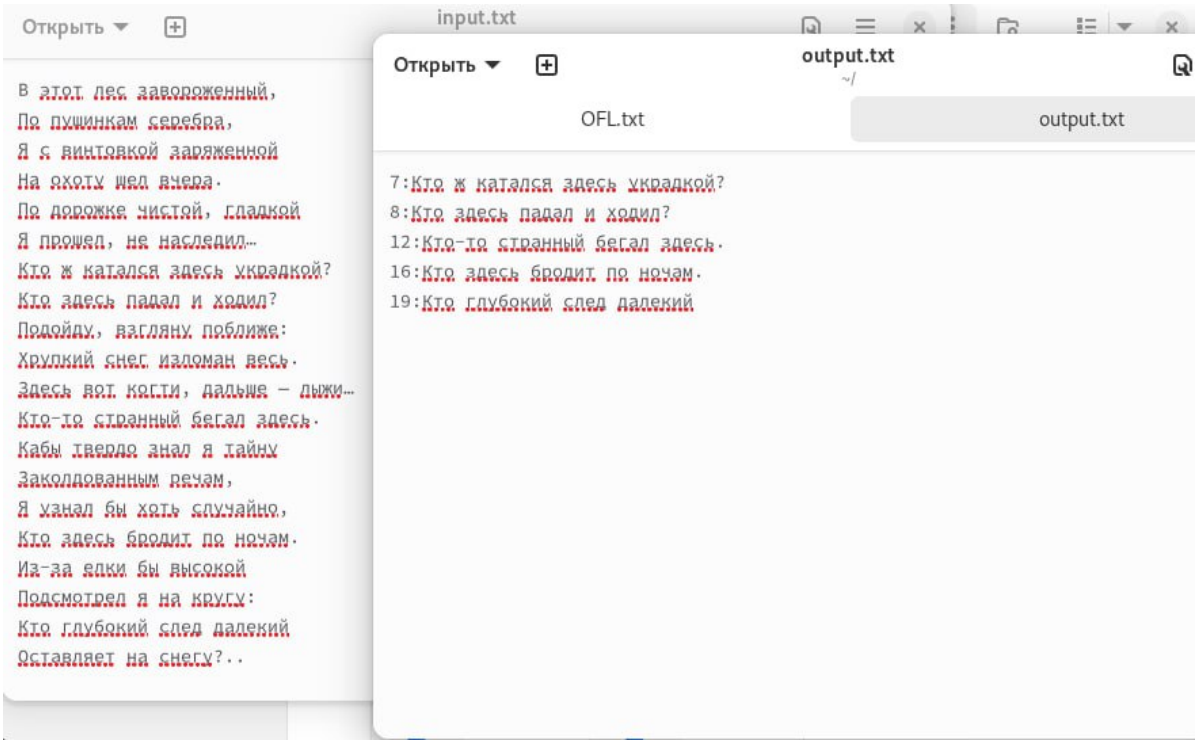

Рис. 4.3: Результат выполнения командного файла №1

2. Нужно 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 о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено (рис. 4.4), (рис. 4.5):

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

int main () {
        int n;
        printf("Велите число: ");
        scanf ("%d", &n);
        if (n > 0){
                exit(1);
        }
        else if (n == 0) {
                exit(0);
        }
        else {
                exit(2);
        }
}
```

Рис. 4.4: Код на C

```bash
#!/bin/bash

gcc -o cprog prog10-2.c
./cprog
case $? in
        0) echo "Число равно нулю";;
        1) echo "Число больше нуля";;
        2) echo "Число меньше нуля";;
esac
```

Рис. 4.5: Код bash

Делаем файлы исполняемыми и выводим результат (рис. 4.6).

```
dssomsikov@dssomsikov:~$ gedit proga10-2.c
dssomsikov@dssomsikov:~$ gedit proga10-2.sh
dssomsikov@dssomsikov:~$ chmod +x proga10-2.c
dssomsikov@dssomsikov:~$ chmod +x proga10-2.sh
dssomsikov@dssomsikov:~$ bash proga10-2.sh
cc1: фатальная ошибка: prog10-2.c: Нет такого файла или каталога
компиляция прервана.
proga10-2.sh: строка 4: ./cprog: Нет такого файла или каталога
dssomsikov@dssomsikov:~$ gedit proga10-2.sh
dssomsikov@dssomsikov:~$ gedit proga10-2.sh
dssomsikov@dssomsikov:~$ bash proga10-2.sh
Велите число: 10
Число больше нуля
dssomsikov@dssomsikov:~$ bash proga10-2.sh
Велите число: 0
Число равно нулю
dssomsikov@dssomsikov:~$ bash proga10-2.sh
Велите число: -222222
Число меньше нуля
dssomsikov@dssomsikov:~$
```

Рис. 4.6: Результат выполнения командного файла №2

3. Нужно написать 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 (рис. 4.7):

```
#!/bin/bash

for ((i=1; i<=$*; i++))
do
        if test -f "$i".tmp
                then rm "$i".tmp
        else touch "$i".tmp
        fi
done
```

Рис. 4.7: Командный файл №3

Делаем файлы исполняемыми и выводим результат (рис. 4.8).

```
dssomsikov@dssomsikov:~$ gedit proga10-3.sh
dssomsikov@dssomsikov:~$ chmod +x proga10-3.sh
dssomsikov@dssomsikov:~$ ls
 abc1        cprog       monthy           proga10-1.sh   program2.sh   work          Музыка
 australia   feathers    my_os            proga10-2.c    program3.sh   Видео         Общедоступные
 backup      file.txt    output.txt       proga10-2.sh   program.sh    Документы     'Рабочий стол'
 Catalog     input.txt   pandoc-crossref  proga10-3.sh   reports       Загрузки      Шаблоны
 conf.txt    may         play             program1.sh    ski.places    Изображения
dssomsikov@dssomsikov:~$ bash proga10-3.sh 4
dssomsikov@dssomsikov:~$ ls
 1.tmp   australia   feathers    my_os            proga10-2.c    program3.sh   Видео         Общедоступные
 2.tmp   backup      file.txt    output.txt       proga10-2.sh   program.sh    Документы     'Рабочий стол'
 3.tmp   Catalog     input.txt   pandoc-crossref  proga10-3.sh   reports       Загрузки      Шаблоны
 4.tmp   conf.txt    may         play             program1.sh    ski.places    Изображения
 abc1    cprog       monthy      proga10-1.sh     program2.sh    work          Музыка
dssomsikov@dssomsikov:~$ bash proga10-3.sh 4
dssomsikov@dssomsikov:~$ ls
 abc1        cprog       monthy           proga10-1.sh   program2.sh   work          Музыка
 australia   feathers    my_os            proga10-2.c    program3.sh   Видео         Общедоступные
 backup      file.txt    output.txt       proga10-2.sh   program.sh    Документы     'Рабочий стол'
 Catalog     input.txt   pandoc-crossref  proga10-3.sh   reports       Загрузки      Шаблоны
 conf.txt    may         play             program1.sh    ski.places    Изображения
dssomsikov@dssomsikov:~$
```

Рис. 4.8: Результат выполнения командного файла №3

4. Нужно 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, использовав команду find (рис. 4.9):

```
#!/bin/bash

find $* -mtime -7 -mtime +0 -type f > Files.txt
tar -cf archive.tar -T Files.txt
```

Рис. 4.9: Командный файл №4

Делаем файлы исполняемыми и выводим результат (рис. 4.10).

```
dssomsikov@dssomsikov:~$ gedit proga10-4.sh
dssomsikov@dssomsikov:~$ chmod +x proga10-4.sh
dssomsikov@dssomsikov:~$ ls
 abc1         cprog       monthy            proga10-1.sh   program1.sh   ski.places   Изображения
 australia    feathers    my_os             proga10-2.c    program2.sh   work         Музыка
 backup       file.txt    output.txt        proga10-2.sh   program3.sh   Видео        Общедоступные
 Catalog      input.txt   pandoc-crossref   proga10-3.sh   program.sh    Документы    'Рабочий стол'
 conf.txt     may         play              proga10-4.sh   reports       Загрузки     Шаблоны
dssomsikov@dssomsikov:~$ bash proga10-4.sh
dssomsikov@dssomsikov:~$ gedit proga10-4.sh
dssomsikov@dssomsikov:~$ ls
 abc1          conf.txt    input.txt    pandoc-crossref   proga10-3.sh   program.sh   Документы      'Рабочий стол'
 archive.tar   cprog       may          play              proga10-4.sh   reports      Загрузки       Шаблоны
 australia     feathers    monthy       proga10-1.sh      program1.sh    ski.places   Изображения
 backup        Files.txt   my_os        proga10-2.c       program2.sh    work         Музыка
 Catalog       file.txt    output.txt   proga10-2.sh      program3.sh    Видео        Общедоступные
dssomsikov@dssomsikov:~$
```

Рис. 4.10: Результат выполнения командного файла №4

5 Контрольные вопросы

1. Каково предназначение команды getopts?

Команда getopts используется для обработки аргументов командной строки. Она позволяет извлекать опции и их значения из списка аргументов.

2. Какое отношение метасимволы имеют к генерации имён файлов?

Метасимволы используются в генерации имён файлов для сопоставления шаблонов. Например, звездочка (*) сопоставляет любое количество символов, а знак вопроса (?) сопоставляет любой один символ.

3. Какие операторы управления действиями вы знаете?

Операторы управления действиями используются для изменения потока выполнения скрипта. Вот некоторые из наиболее распространенных операторов управления действиями:

- if...then...else: Выполняет блок кода, если условие истинно. Если условие ложно, выполняется блок кода else.
- for...do...done: Выполняет блок кода для каждого элемента в списке.
- while...do...done: Выполняет блок кода, пока условие истинно.
- until...do...done: Выполняет блок кода, пока условие ложно.

4. Какие операторы используются для прерывания цикла?

- continue: Переходит к следующей итерации цикла, пропуская оставшиеся операторы в текущей итерации.
- break: Немедленно выходит из цикла.

5. Для чего нужны команды false и true?

Команды false и true используются для возврата кода выхода, указывающего на успех (true) или неудачу (false).

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

Эта строка проверяет, существует ли файл с именем man$s$/i.$s. Если файл существует, выполняется оператор then.

7. Объясните различия между конструкциями while и until.

- while: Выполняет код, пока условие истинно.
- until: Выполняет код, пока условие ложно.

6 Выводы

В данной лабораторной работе я изучила основы программирования в оболочке ОС UNIX, а также научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.