Лабораторная работа №11

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Старовойтов Егор Сергеевич

Содержание

Цель работы

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

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 *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).

Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

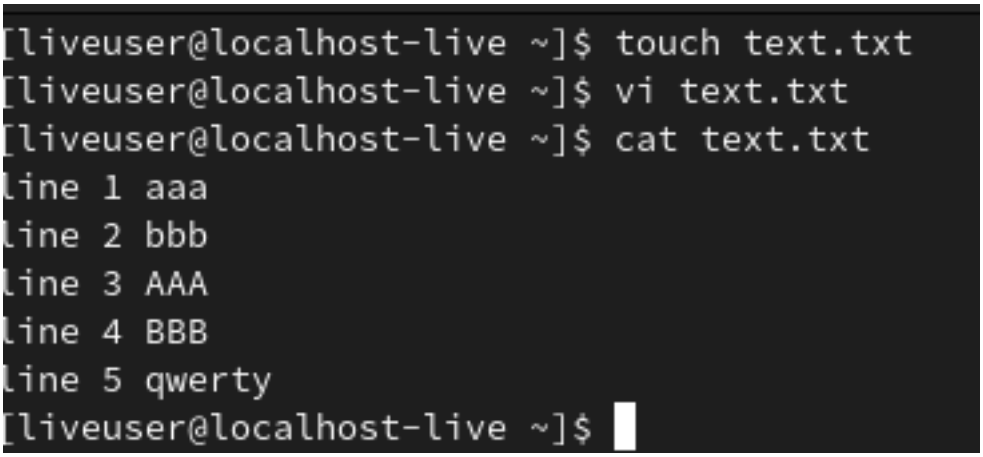
- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
- C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости

различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке bash. В других оболочках большинство команд будет совпадать с описанными ниже.

Ход работы

Задание 1

Я написал скрипт, выполняющий первое задание. Использовал текстовый редактор vi. Написанный скрипт содержится в командном файле “prog1.sh”. Для тестирования программы был создан файл text.txt со следующим содержимым:



text.txt

Код командного файла:

```
#!/bin/bash
while getopts i:o:p:cn optletter
do case $optletter in
    i)input_file=$OPTARG;;
    o)output_file=$OPTARG;;
    p)target=$OPTARG;;
    c)case_sens=true;;
    n)line_nums=true;;
esac
done

if [ $case_sens ]
then
    if [ $line_nums ]
    then
        grep -n $target $input_file > $output_file
        exit 0
    else
        grep $target $input_file > $output_file
        exit 0
    fi
else
    if [ $line_nums ]
    then
        grep -i -n $target $input_file > $output_file
        exit 0
    else
        grep -i $target $input_file > $output_file
        exit 0
    fi
fi
exit 0
[liveuser@localhost-live ~]$
```

prog1.sh

Работа программы:

```
[liveuser@localhost-live ~]$ ./prog1.sh -p "aaa" -i text.txt -o out.txt -c -n
[liveuser@localhost-live ~]$ cat out.txt
1:line 1 aaa
[liveuser@localhost-live ~]$ ./prog1.sh -p "aaa" -i text.txt -o out.txt -n
[liveuser@localhost-live ~]$ cat out.txt
1:line 1 aaa
3:line 3 AAA
[liveuser@localhost-live ~]$ ./prog1.sh -p "aaa" -i text.txt -o out.txt
[liveuser@localhost-live ~]$ cat out.txt
line 1 aaa
line 3 AAA
[liveuser@localhost-live ~]$ ./prog1.sh -p "999" -i text.txt -o out.txt
[liveuser@localhost-live ~]$ cat out.txt
[liveuser@localhost-live ~]$ ./prog1.sh -p "file" -i text.txt -o out.txt
[liveuser@localhost-live ~]$ cat out.txt
[liveuser@localhost-live ~]$ ./prog1.sh -p "line" -i text.txt -o out.txt
[liveuser@localhost-live ~]$ cat out.txt
line 1 aaa
line 2 bbb
line 3 AAA
line 4 BBB
line 5 qwerty
```

Работа prog1.sh

Задание 2

Я написал скрипт, выполняющий второе задание, а также требуемую программу на языке Си. Использовал текстовый редактор vi. Компилировал программу на языке Си с помощью gcc, имя испольняемого файла - a.out.

Коды: - 0 - число равно 0 - 1 - число больше 0 - 2 - число меньше 0

```
[liveuser@localhost-live ~]$ vi comp.c
[liveuser@localhost-live ~]$ gcc comp.c
[liveuser@localhost-live ~]$ cat comp.c
#include <stdio.h>
#include <stdlib.h>

int main() {
    printf("Input your number: ");
    int num;
    scanf("%d", &num);

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

[liveuser@localhost-live ~]$ ls
a.out  comp.c  Desktop  Documents  Downloads
[liveuser@localhost-live ~]$
```

comp.c

Листинг командного файла:

```
#!/bin/bash

./a.out
code=$?

case $code in
0) echo "= 0";;
1) echo "> 0";;
2) echo "< 0";;
esac

exit 0
```

prog2.sh

Работа командного файла

```
[liveuser@localhost-live ~]$ ./prog2.sh
Input your number: -5
[liveuser@localhost-live ~]$ vi prog2.sh
[liveuser@localhost-live ~]$ ./prog2.sh
Input your number: 5
> 0
[liveuser@localhost-live ~]$ ./prog2.sh
Input your number: 0
= 0
[liveuser@localhost-live ~]$ ./prog2.sh
Input your number: -5
< 0
[liveuser@localhost-live ~]$
```

Работа prog2.sh

Задание 3

Я написал скрипт, выполняющий третье задание. Использовал текстовый редактор vi. Программа анализирует строку аргументов с двумя ключами и параметром n.

Ключи: - m - создать n .tmp файлов - d - удалить все .tmp файлы

Код программы:

```
[liveuser@localhost-live ~]$ vi prog3.sh
[liveuser@localhost-live ~]$ cat prog3.sh
#!/bin/bash
while getopts m:d optletter
do
    case $optletter in
        m) n=$OPTARG;
           for i in $(seq 1 $n)
                do touch "$i.tmp";
           done;;
        d) for fname in $(find -name "*.tmp")
               do rm $fname;
           done;;
    esac;
done

exit 0
```

prog3.sh

Работа программы:

```
[liveuser@localhost-live ~]$ chmod +x prog3.sh
[liveuser@localhost-live ~]$ ls
a.out  comp.c  Desktop  Documents  Downloads  Music  out.txt  Pictures  prog1.sh  prog2.sh  prog3.sh  Public  Templates  text.txt  Videos
[liveuser@localhost-live ~]$ ./prog3.sh -m 5
[liveuser@localhost-live ~]$ ls
1.tmp  2.tmp  3.tmp  4.tmp  5.tmp  a.out  comp.c  Desktop  Documents  Downloads  Music  out.txt  Pictures  prog1.sh  prog2.sh  prog3.sh  Public  Templates  text.txt  Videos
[liveuser@localhost-live ~]$ ./prog3.sh -d
[liveuser@localhost-live ~]$ ls
a.out  comp.c  Desktop  Documents  Downloads  Music  out.txt  Pictures  prog1.sh  prog2.sh  prog3.sh  Public  Templates  text.txt  Videos
[liveuser@localhost-live ~]$
```

Работа prog3.sh

Задание 4

Я написал скрипт, выполняющий четвертое задание. Использовал текстовый редактор vi.

Код программы:

```
[liveuser@localhost-live ~]$ cat prog4.sh
#!/bin/bash
find $1 -mtime 0 -mtime -7 | xargs -o tar -cf files.tar
exit 0
```

prog4.sh

Работа программы:

Работа prog4.sh

Вывод

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

Контрольные вопросы

1. Каково предназначение команды getopts?

Команда getopts осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных.

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-c1] — 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 от результатов проверки некоторого условия. Для решения подобных задач язык программирования bash предоставляет возможность использовать такие управляющие конструкции, как for, case, if, while и until.

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 mans/i.s, встреченная в командном файле? Эта строка проверяет, является ли аргумент mans/i.s обычным файлом (не каталогом) и возвращает логическое значение.

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. В обобщённой форме оператор цикла until выглядит следующим образом: