

Отчёт по лабораторной работе №11

Программирование в командном процессоре ОС UNIX. Ветвления и
циклы

Смирнов-Мальцев Егор Дмитриевич

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;
- `-r`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 `-r`.

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 оболочке 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.

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;
- `-rшаблон` — 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`.

```
[edds@rnomal]ceeds@rnomal$ ./script1.sh -script1.sh -p. -n
1:#!/bin/bash
2:while getopts f:op:gn optletter
3:do case $optletter in
4:f) flag+=1; value+=OPTARG;;
5:*) echo 'incorrect option'
6:; if ((flag==0))
7:; then echo 'enter input file'
8:; else if ((flag==1))
9:; then grep -e $(value) -n $(value) > $(value)
10:; else if ((flag==2))
11:; then grep -e $(value) $(value) > $(value)
12:; else if ((flag==3))
13:; then grep -e $(value) -n $(value) > $(value)
14:; else if ((flag==4))
15:; then grep -e $(value) $(value) > $(value)
16:; else if ((flag==5))
17:; then cat $(value) | grep -e $(value) -n
18:; else grep -e $(value) $(value)
```

рис. 1: выполнение первого скрипта.

2. Написал на языке C++ 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.

```
[edsmirnovmaljce@edsmirnovmaljce programs]$ ./script2.sh
Enter x: 7
Entered number bigger than 0
```

рис. 2: выполнение второго скрипта.

3. Написал 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 умеет удалять все созданные им файлы (если они существуют).

```
[edsmirnovmaljce@edsmirnovmaljce programs]$ ./script3.sh 3 1
made tmp file number 1
made tmp file number 2
made tmp file number 3
[edsmirnovmaljce@edsmirnovmaljce programs]$ ./script3.sh 3 2
deleted tmp file number 1
deleted tmp file number 2
deleted tmp file number 3
```

рис. 3: выполнение третьего скрипта.

4. 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.

```
[edsmirnovmaljce@edsmirnovmaljce lab11]$ ./programs/script4.sh programs
made programs.tar.gz
[edsmirnovmaljce@edsmirnovmaljce lab11]$ ls
presentation  programs  programs.tar.gz  report
```

рис. 4: выполнение четвертого скрипта.

Ответы на контрольные вопросы

1. Команда `getopts` предназначена для разбора параметров сценариев. Она обрабатывает исключительно однобуквенные параметры как с аргументами, так и без них.

2. Метасимволы - символы, имеющие специальное значение для интерпретатора :

? * ; & () | ^ < > <пробел> <табуляция> <возврат_каретки>

Однако каждый из этих символов может представлять самого себя, если перед ним стоит `.`. Все символы, заключенные между кавычками `'` и `'`, представляют самих себя. Между двойными кавычками (`"`) выполняются подстановки команд и параметров.

После всех подстановок в каждом слове команды ищутся символы `*`, `?`, и `[`. Если находится хотя бы один из них, то это слово рассматривается как шаблон имен файлов и заменяется именами файлов, удовлетворяющих данному шаблону (в алфавитном порядке).

3. Набор управляющих операторов Bash включает в себя: `if`, `while`, `until`, `for` и `case`. Каждый из этих операторов является парным, то есть начинается он одним тегом и заканчивается другим.

4. Циклы прерываются оператором `done`. Также выход из цикла может произойти с помощью оператора `break`.

5. Команды `false` и `true` нужны для оценки успешности выполнения скрипта.

6. строка `if test -f man$s/$i.$s` означает что следующие действия надо выполнить если существует файл подходящий под указанный шаблон.

7. Суть цикла While в том, что он будет повторяться до тех пор, пока будет выполняться условие, указанное в объявлении цикла. Цикл until будет выполняться пока условие неверно.

Выводы

- Я научился писать программы с циклами и ветвлениями на языке `bash`.