

Лабораторная работа № 13. Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Отчёт

Сергеев Д. О.

7 мая 2025

Российский университет дружбы народов, Москва, Россия

Информация

- Сергеев Даниил Олегович
- Студент
- Направление: Прикладная информатика
- Российский университет дружбы народов
- 1132246837@pfur.ru

Цель работы и задание

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

Написать командные файлы для задач лабораторной работы.

Ход выполнения лабораторной работы

Создадим каталог lab13 с дополнительными директориями для каждого задания. Приступим к выполнению первой задачи.

Используя команды `getopts` и `grep`, напомним командный файл, который анализирует командную строку с ключами, а затем ищет в указанном файле нужные строки, определяемые ключом.

- `-i: inputfile` – прочитать данные из указанного файла;
- `-o: outputfile` – вывести данные в указанный файл;
- `-p: template` – указать шаблон для поиска;
- `-C` – различать большие и малые буквы;
- `-n` – выдавать номера строк;

Выполнение упражнений

```
[dosergeev@vbox 1]$ ls
a  code.sh  code.sh~
[dosergeev@vbox 1]$ ./code.sh -o ~/file.txt -p fi
inputfile=$0
outputfile=""
    cat $inputfile | grep $cflag $nflag "$template"
        inputfile=$OPTARG
    fi
        outputfile=$OPTARG
    fi
    if [ $outputfile ]
        read > $outputfile
    fi
fi
[dosergeev@vbox 1]$ cat ~/file.txt
inputfile=$0
outputfile=""
    cat $inputfile | grep $cflag $nflag "$template"
        inputfile=$OPTARG
    fi
        outputfile=$OPTARG
    fi
    if [ $outputfile ]
        read > $outputfile
    fi
fi
```

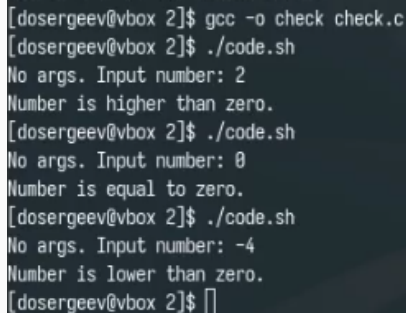
Рис. 1: Работа первого скрипта.

```
foot
[dosergeev@vbox 1]$ ./code.sh -o ~/file.txt -p optarg -n
12:     if [ -f $OPTARG ]
14:         inputfile=$OPTARG
18:     if [ -f $OPTARG ]
20:         outputfile=$OPTARG
24:         template=$OPTARG
[dosergeev@vbox 1]$ ./code.sh -o ~/file.txt -p optarg -n
C
[dosergeev@vbox 1]$ ./code.sh -o ~/file.txt -p OPTARG -n
C
12:     if [ -f $OPTARG ]
14:         inputfile=$OPTARG
18:     if [ -f $OPTARG ]
20:         outputfile=$OPTARG
24:         template=$OPTARG
[dosergeev@vbox 1]$
```

Рис. 2: Работа первого скрипта с дополнительными ключами.

Выполнение упражнений

Теперь напишем на языке Си программу, которая определяет, является ли введенное число меньше, больше нуля или равно нулю. Данная программа должна завершаться с помощью команды `exit(n)`, передавая код завершения `n` в оболочку. Также необходимо написать командный файл, который будет анализировать результат с помощью команды `$?`.



```
[dosergeev@vbox 2]$ gcc -o check check.c
[dosergeev@vbox 2]$ ./code.sh
No args. Input number: 2
Number is higher than zero.
[dosergeev@vbox 2]$ ./code.sh
No args. Input number: 0
Number is equal to zero.
[dosergeev@vbox 2]$ ./code.sh
No args. Input number: -4
Number is lower than zero.
[dosergeev@vbox 2]$
```

Рис. 3: Результат второго скрипта.

Следующий командный файл должен уметь создавать указанное число файлов, пронумерованных от 1 до некоторого N. Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы, если они существуют.

```
[dosergeev@vbox 3]$ ./code.sh 2
2
[dosergeev@vbox 3]$ ls
1.tmp 2.tmp code.sh code.sh~
[dosergeev@vbox 3]$ ./code.sh 2
2
[dosergeev@vbox 3]$ ls
code.sh code.sh~
[dosergeev@vbox 3]$ ./code.sh 7
7
[dosergeev@vbox 3]$ ls
1.tmp 3.tmp 5.tmp 7.tmp code.sh~
2.tmp 4.tmp 6.tmp code.sh
[dosergeev@vbox 3]$ ./code.sh 4
4
[dosergeev@vbox 3]$ ls
5.tmp 6.tmp 7.tmp code.sh code.sh~
[dosergeev@vbox 3]$ ./code.sh 7
7
[dosergeev@vbox 3]$ ./code.sh 4
4
[dosergeev@vbox 3]$ ls
code.sh code.sh~
[dosergeev@vbox 3]$
```

Рис. 4: Результат третьего скрипта.

Выполнение упражнений

Последний командный файл должен с помощью команды tar запаковывать в архив все файлы в указанной директории, которые были изменены менее недели тому назад.

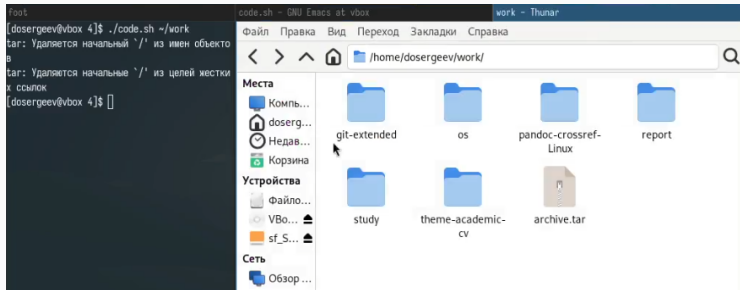


Рис. 5: Результат четвертого скрипта.

1. Команда `getopts` считывает аргументы командной строки в поиске ключей и записывает их в заданную переменную `optletter`
2. Перед выполнением команды каждый аргумент команды просматривается в поисках метасимволов, например `*`, `?`, и `[`, которые считаются как шаблон имён файлов и заменяется именами, соответствующими этому шаблону в алфавитном порядке.
3. Операторы управления действиями:
 - Операторы условия: `if`, `else`, `elif`;
 - Циклы: `for`, `while`;
 - Управление выполнением: `break`, `exit`, `continue`;
 - Логические операторы: `&&(И)`, `||(ИЛИ)`, `!(НЕ)`;
 - Группировки команд: `()` - Создание подпроцесса для выполнения команд;

4. Для прерывания цикла используются операторы
 - break – для выхода из оператора
 - exit – для выхода из программы
 - continue – для прерывания итерации цикла
5. Операторы false и true нужны для обозначения успешного и неуспешного завершения выполнения команды

6. `'if test -f man$s/$i.$s'` – данная строка проверяет, существует ли объект `'$i.$s'` и является ли он файлом в относительном каталоге `'man$s/'`, где `'$i'` и `'$s'` – подставленные значения переменных `i` и `s` соответственно.

7.

- `while` – цикл с предусловием, пока условие не станет `false`;
- `until` – цикл с постусловием, пока условие не станет `true`;

Вывод

В результате выполнения лабораторной работы я изучил основы программирования в оболочке ОС UNIX/Linux и научился писать более сложные командные файлы.