

Лабораторная работа №12

Пермякова Елизавета Евгеньевна¹

26 мая, 2021, Москва, Россия

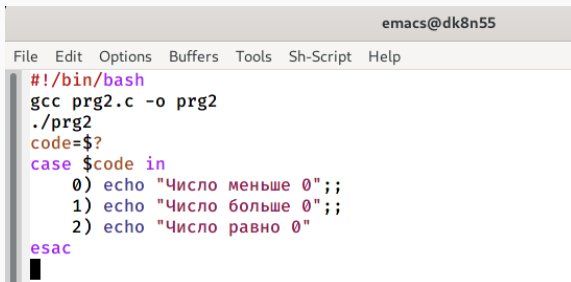
¹RUDN University, Moscow, Russian Federation

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

1. Познакомиться с логическими управляющими конструкцией и циклов.
2. В ходе работы написать 4 командных файла.
3. Выполнить отчет.

Выполнение лабораторной работы

Предварительно для командных файлов создаем файл и открываем emacs. Первый скрипт анализирует командную строку с ключами, используя команду `getopts` `grep`. Второй командный файл должен вызывать предварительно написанную на языке Си программу, которая выводит число и определяет его равенство или неравенство с нулем. (рис. 1)



```
emacs@dk8n55
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
gcc prg2.c -o prg2
./prg2
code=$?
case $code in
  0) echo "Число меньше 0";;
  1) echo "Число больше 0";;
  2) echo "Число равно 0"
esac
```

Figure 1: Файл `prg2.sh`

Выполнение лабораторной работы

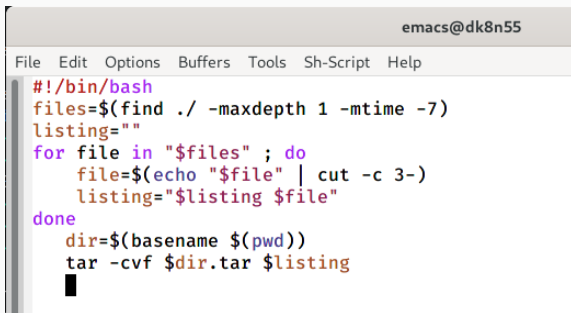
Третий командный файл создает указанное число файлов, пронумерованных от 1 до n. Также он умеет удалять все созданные им файлы. (рис. 2)

The image shows a screenshot of an Emacs editor window. The title bar at the top reads 'emacs@dk8n55'. Below the title bar is a menu bar with the following items: 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Sh-Script', and 'Help'. The main editing area contains a shell script. The script starts with a shebang line '#!/bin/bash'. It then assigns values to variables: 'opt=\$1;', 'format=\$2;', and 'number=\$3;'. A function named 'Files()' is defined. Inside this function, there is a 'for' loop that iterates from 'i=1' to 'i=\$number'. Inside the loop, a file path is constructed using 'file=\$(echo \$format | tr '#' "\$i")'. There is a conditional check 'if [\$opt == "-r"]' which, if true, runs 'rm -f \$file'. Otherwise, if the condition is false, it runs 'touch \$file'. The loop ends with 'fi', the function ends with 'done' and 'fi'. The script ends with a closing brace '}' and the word 'Files' followed by a cursor. The script is color-coded: shebang is purple, variable assignments are orange, function definition is purple, loop and conditional keywords are purple, file path construction is orange, and file operations are orange.

```
#!/bin/bash
opt=$1;
format=$2;
number=$3;
function Files() {
    for (( i=1; i<=$number; i++ )) do
        file=$(echo $format | tr '#' "$i")
        if [ $opt == "-r" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
Files
```

Figure 2: Третий скрипт

Четвертый командный файл с помощью команды tar запаковывает в архив все файлы в указанной директории. Он также модифицирован таким образом, чтобы запаковывались только те файлы, которые были изменены менее неделю тому назад. (рис. 3)

The image shows a terminal window with a title bar that reads 'emacs@dk8n55'. Below the title bar is a menu bar with the following items: 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Sh-Script', and 'Help'. The main area of the terminal displays a shell script. The script starts with a shebang line '#!/bin/bash'. It then assigns the output of 'find ./ -maxdepth 1 -mtime -7' to a variable named 'files'. A variable 'listing' is initialized as an empty string. A 'for' loop iterates over each file in '\$files'. Inside the loop, the file's name is extracted using 'echo "\$file" | cut -c 3-' and assigned to 'file'. The 'file' variable is then appended to the 'listing' variable. After the loop, the script uses 'basename \$(pwd)' to get the current directory name and assigns it to 'dir'. Finally, it runs the command 'tar -cvf \$dir.tar \$listing' to create the archive. The script ends with a blank line.

```
#!/bin/bash
files=$(find ./ -maxdepth 1 -mtime -7)
listing=""
for file in "$files" ; do
    file=$(echo "$file" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

Figure 3: Четвертый скрипт

В ходе выполнения данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX и научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.