

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Лабораторная работа No 11

Козлова Нонна

Российский университет дружбы народов, Москва, Россия

НБИбд-04-22

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

1. Используя команды `getopts` `grep`, написала командный файл, который анализирует командную строку с ключами: `-i`inputfile — прочитать данные из указанного файла; `-o`outputfile — 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`.(рис. (fig:001?)).

```
#!/bin/bash
iflag=0; oflag=0; cflag=0; nflag=0;
while getopts i:o:p:C:n 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 $optletter
esac
done
if (($pflag==0))
then echo "шаблон не найден"
else
    if (($iflag==0))
    then echo "шаблон не найден"
    else
```

2. Написала на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю.(рис. (fig:002?)), (рис. (fig:003?)).

A screenshot of a code editor window showing a C program. The window has a title bar with a button labeled 'Открыть' (Open), a dropdown arrow, and a file icon. The file name is '*prog2.c' and the path is '~/work/study/2022-2023/Операционные с'. The code is as follows:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     printf("Введите число: ");
6     int a;
7     scanf("%d", &a);
8     if (a<0) exit(0);
9     if (a>0) exit(1);
10    if(a==0) exit(2);
11    return 0;
12 }
13
```

Рис. 2: Код



```
1 #!/bin/bash
2
3 gcc prog2.c -o prog2
4 ./prog2
5 code=$?
6 case $code in
7     0) echo "число меньше 0";;
8     1) echo "число больше 0";;
9     2) echo "число равно 0";;
10 esac
```

Рис. 3: Код

3. Проверяю работу программы (рис. (fig:004?)).

```
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/1
ab11 $ ./prog2.sh
Введите число: 3
число больше 0
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $ ./prog2.sh
Введите число: -5
число меньше 0
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $ ./prog2.sh
Введите число: 0
число равно 0
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $
```

Рис. 4: Работает

4. Пишу командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до [?] (рис. (fig:005?)).

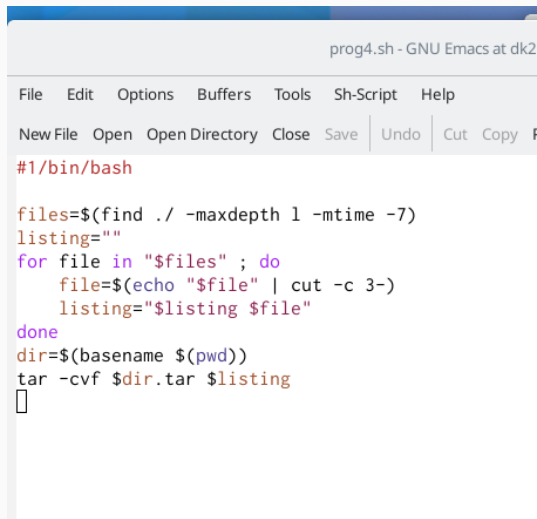
```
1 #!/bin/bash
2
3 opt=$1;
4 form=$2;
5 num=$3;
6 function Files() {
7     for ((i=1; i<$num; i++)) do
8         file=$(echo $form | tr '#' "$i")
9         if [ $opt == "-r" ]
10            then
11                rm -f &file
12            elif [ $opt == "c" ]
13                then
14                    touch $file
15            fi
16        done
17    }
18 Files
```

5. Проверяю работу программы (рис. (fig:006?)).

```
udy/2022-2023/Операционные системы/os-intro/labs/lab11 $ touch prog3.sh
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $ chmod +x *.sh
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $ ls
lab11.sh lab11.txt presentation prog2 prog2.c prog2.sh prog3.sh report
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $ ./prog3.sh -r a#.txt 3
Usage: file [-bCdEhikLlNnprsSvzZ0] [--apple] [--extension] [--mime-encoding]
           [--mime-type] [-e <testname>] [-F <separator>] [-f <namefile>]
           [-m <magicfiles>] [-P <parameter=value>] [--exclude-quiet]
           <file> ...
           file -C [-m <magicfiles>]
           file [--help]
Usage: file [-bCdEhikLlNnprsSvzZ0] [--apple] [--extension] [--mime-encoding]
           [--mime-type] [-e <testname>] [-F <separator>] [-f <namefile>]
           [-m <magicfiles>] [-P <parameter=value>] [--exclude-quiet]
           <file> ...
           file -C [-m <magicfiles>]
           file [--help]
Usage: file [-bCdEhikLlNnprsSvzZ0] [--apple] [--extension] [--mime-encoding]
           [--mime-type] [-e <testname>] [-F <separator>] [-f <namefile>]
           [-m <magicfiles>] [-P <parameter=value>] [--exclude-quiet]
           <file> ...
           file -C [-m <magicfiles>]
           file [--help]
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $ ls
lab11.sh lab11.txt presentation prog2 prog2.c prog2.sh prog3.sh report
```

Рис. 6: Работает

6. Пишу командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицирую его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (рис. (fig:007?)).

A screenshot of a GNU Emacs editor window. The title bar at the top reads "prog4.sh - GNU Emacs at dk2". Below the title bar is a menu bar with items: File, Edit, Options, Buffers, Tools, Sh-Script, and Help. Below the menu bar is a toolbar with icons for New File, Open, Open Directory, Close, Save, Undo, Cut, Copy, and Paste. The main text area contains a shell script with the following content:

```
#!/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
```

The script is a bash script that finds files in the current directory and its subdirectories (up to a depth of 1) that have been modified within the last 7 days. It then creates a listing of these files and uses the tar command to create an archive named \$dir.tar, where \$dir is the basename of the current directory.

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