

# **Лабораторная работа №11**

**Программирование в командном процессоре ОС UNIX. Ветвления и циклы**

Заболотная Кристина Александровна

# Содержание

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Цель работы</b>                      | <b>5</b>  |
| <b>2</b> | <b>Задание</b>                          | <b>6</b>  |
| <b>3</b> | <b>Выполнение лабораторной работы</b>   | <b>7</b>  |
| <b>4</b> | <b>Контрольные вопросы</b>              | <b>14</b> |
| 4.1      | Ответы на контрольные вопросы . . . . . | 14        |
| <b>5</b> | <b>Выводы</b>                           | <b>16</b> |
|          | <b>Список литературы</b>                | <b>17</b> |

## Список иллюстраций

|     |                               |    |
|-----|-------------------------------|----|
| 3.1 | создаем lab11.txt . . . . .   | 7  |
| 3.2 | текст из интернета . . . . .  | 8  |
| 3.3 | первый скрипт . . . . .       | 9  |
| 3.4 | проверяем lab11.txt . . . . . | 10 |
| 3.5 | prog2.c и sh . . . . .        | 11 |
| 3.6 | prog3.sh . . . . .            | 12 |
| 3.7 | prog4.sh . . . . .            | 13 |

## Список таблиц

# 1 Цель работы

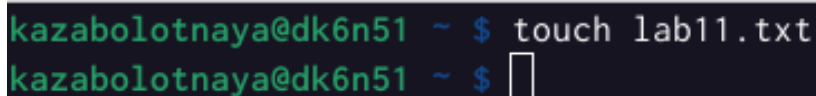
Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 2 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; `-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`.
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  $\infty$  (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 `tag` 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`).

### 3 Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, написан командный файл, который анализирует командную строку с ключами: `-iinputfile` — прочитать данные из указанного файла; `-ooutputfile` — вывести данные в указанный файл; `-р` — 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 `-р`.



```
kazabolotnaya@dk6n51 ~ $ touch lab11.txt
kazabolotnaya@dk6n51 ~ $
```

Рис. 3.1: создаем lab11.txt

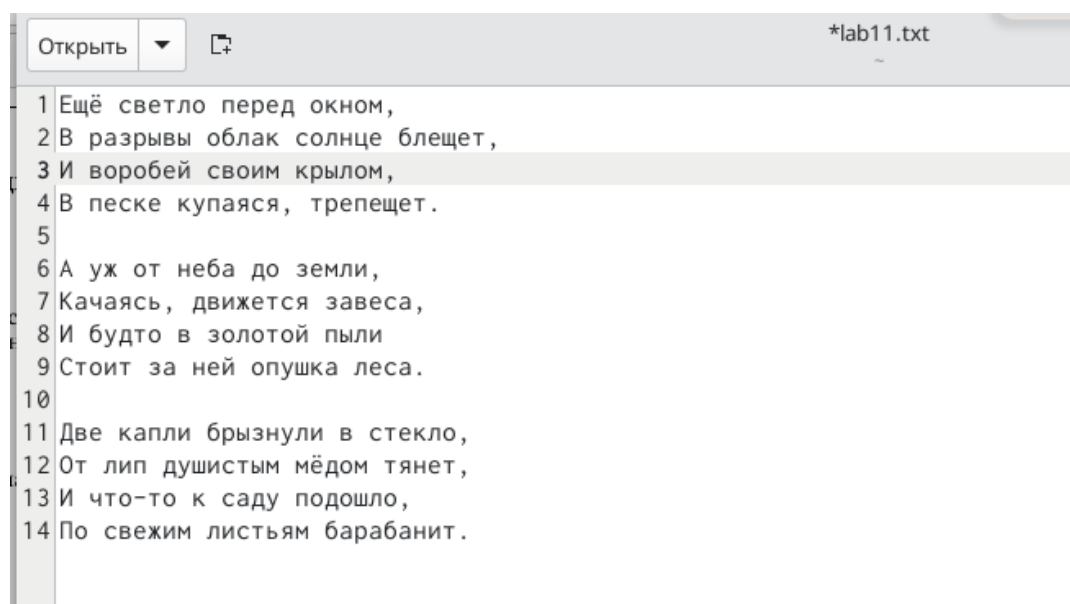


Рис. 3.2: текст из интернета



```
Открыть  lab11.sh
1 #!/bin/bash
2 iflag=0; oflag=0; pflag=0; Cflag=0; nflag=0;
3 while getopts i:o:p:C:n optletter
4 do case $optletter in
5     i) iflag=1; ival=$OPTARG;;
6     o) oflag=1; oval=$OPTARG;;
7     p) pflag=1; pval=$OPTARG;;
8     C) Cflag=1;;
9     n) nflag=1;;
10    *) echo illegal option $optletter
11    esac
12 done
13 if (($pflag==0))
14 then echo "Шаблон не найден"
15 else
16     if (($iflag==0))
17     then echo "Файл не найден"
18     else
19         if (($oflag==0))
20         then if (($Cflag==0))
21             then if (($nflag==0))
22                 then grep $pval $ival
23                 else grep -n $pval $ival
24                 fi
25             else if (($nflag==0))
26                 then grep -i $pval $ival
27                 else grep -i -n $pval $ival
28                 fi
29             fi
30         else if (($Cflag==0))
31             then if (($nflag==0))
32                 then grep $pval $ival > $oval
33                 else grep -n $pval $ival > $oval
34                 fi
35             else if (($nflag==0))
36                 then grep -i $pval $ival > $oval
37                 else grep -i -n $pval $ival > $oval
38                 fi
39             fi
40         fi
41     fi
42 fi
```

Рис. 3.3: первый скрипт

```
kazabolotnaya@dk6n51 ~ $ cat ~/lab11.txt
Ещё светло перед окном,
В разрывы облак солнце блещет,
И воробей своим крылом,
В песке купаясь, трепещет.

А уж от неба до земли,
Качаясь, движется завеса,
И будто в золотой пыли
Стоит за ней опушка леса.

Две капли брызнули в стекло,
От лип душистым мёдом тянет,
И что-то к саду подошло,
По свежим листьям барабанит.
kazabolotnaya@dk6n51 ~ $
```

Рис. 3.4: проверяем lab11.txt

2. Написана на языке Си программа, 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.

The screenshot shows a GNU Emacs editor window titled "prog2.sh - GNU Emacs at dk8n59". The menu bar includes File, Edit, Options, Buffers, Tools, Sh-Script, Outline, Hide/Show, and Help. The main text area contains a shell script for prog2.sh:

```
#!/bin/bash
gcc prog2.c -o prog2
./prog2
code=$?
case $code in
    0) echo "Число меньше 0";;
    1) echo "Число больше 0";;
    2) echo "Число равно 0";;
esac
```

Below the script, there is a toolbar with buttons for "Открыть" (Open), a file icon, "prog2.c", "Сохранить" (Save), and window management icons. The bottom pane shows the source code for prog2.c:

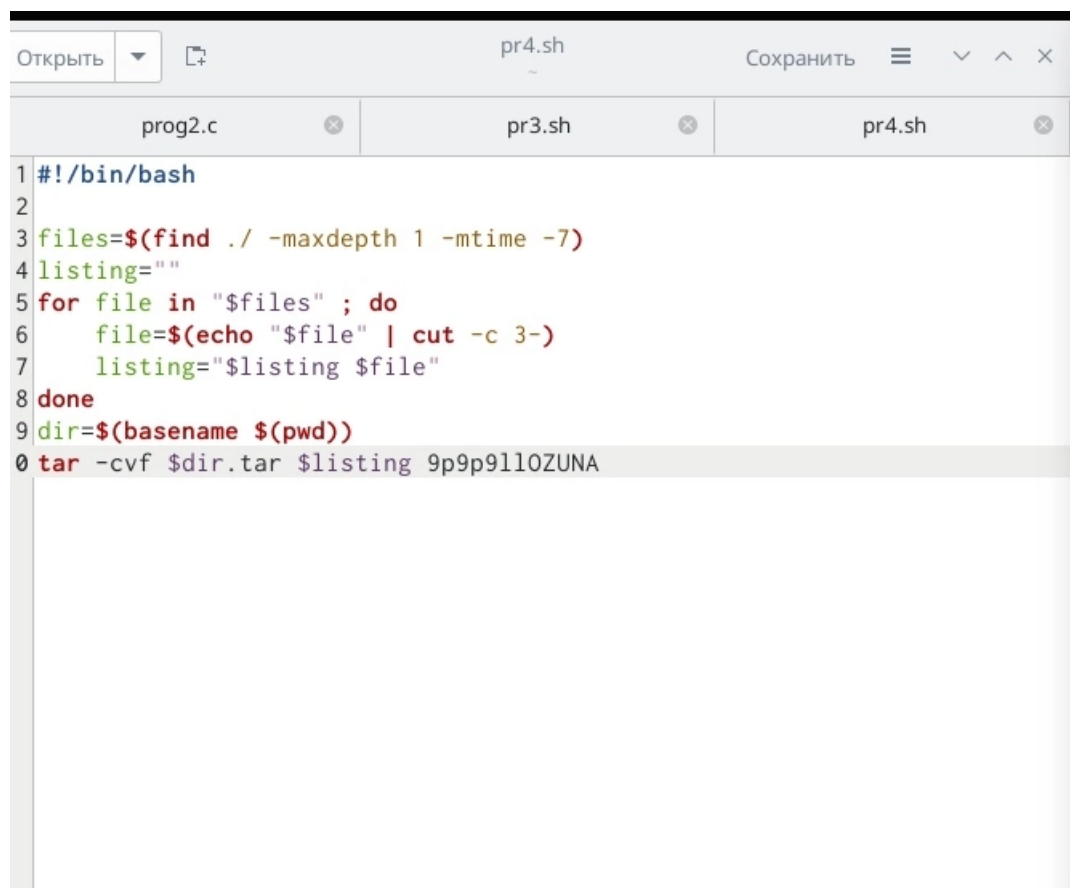
```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     printf("Введите число: ");
6     int a;
7     scanf("%d",&a);
8     if (a<0) exit(0);
9     if (a>0) exit(1);
10    if (a==0) exit(2);
11    return 0;
12 }
```

Рис. 3.5: prog2.c и sh

3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до  $\infty$  (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).

Рис. 3.6: prog3.sh

4. Написан 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).



```
1 #!/bin/bash
2
3 files=$(find ./ -maxdepth 1 -mtime -7)
4 listing=""
5 for file in "$files" ; do
6     file=$(echo "$file" | cut -c 3-)
7     listing="$listing $file"
8 done
9 dir=$(basename $(pwd))
0 tar -cvf $dir.tar $listing 9p9p9l1l0ZUNA
```

Рис. 3.7: prog4.sh

## 4 Контрольные вопросы

1. Каково предназначение команды `getopts`?
2. Какое отношение метасимволы имеют к генерации имён файлов?
3. Какие операторы управления действиями вы знаете?
4. Какие операторы используются для прерывания цикла?
5. Для чего нужны команды `false` и `true`?
6. Что означает строка `if test -f mans/i.$s`, встреченная в командном файле?
7. Объясните различия между конструкциями `while` и `until`.

### 4.1 Ответы на контрольные вопросы

1. Команда `getopts` является встроенной командой командной оболочки `bash`, предназначенной для разбора параметров сценариев. Она обрабатывает исключительно однобуквенные параметры как с аргументами, так и без них и этого вполне достаточно для передачи сценариям любых входных данных.
2. При генерации имен используют метасимволы: • произвольная (возможно пустая) последовательность символов; ? один произвольный символ; [...] любой из символов, указанных в скобках перечислением и/или с указанием диапазона; `cat f*` выдаст все файлы каталога, начинающиеся с “f”; `cat f` выдаст все файлы, содержащие “f”; `cat program.?` выдаст файлы данного каталога с однобуквенными расширениями, скажем “program.c” и “program.o”, но не выдаст “program.com”; `cat [a-d]*` выдаст файлы, которые начинаются

с “a”, “b”, “c”, “d”. Аналогичный эффект дадут и команды “cat [abcd]” и “cat [bdac]”.

3. Операторы && и || являются управляющими операторами. Если в командной строке стоит command1 && command2, то command2 выполняется в том, и только в том случае, если статус выхода из команды command1 равен нулю, что говорит об успешном ее завершении. Аналогично, если командная строка имеет вид command1 || command2, то команда command2 выполняется тогда, и только тогда, когда статус выхода из команды command1 отличен от нуля.
4. Оператор break завершает выполнение ближайшего включающего цикла или условного оператора, в котором он отображается.
5. Команда true всегда возвращает ноль в качестве выходного статуса для индикации успеха. Команда false всегда возвращает не-ноль в качестве выходного статуса для индикации неудачи. Во всех управляющих конструкциях в качестве логического значения используется код возврата из программы, указанной в качестве условия. Код возврата 0 – истина, любое другое значение – ложь. Программа true – всегда завершается с кодом 0, false – всегда завершается с кодом 1.
6. Введенная строка означает условие существования файла man❏/i.\$s
7. Цикл While выполняется до тех пор, пока указанное в нем условие истинно. Когда указанное условие становится ложным - цикл завершается. Цикл Until выполняется до тех пор, пока указанное в нем условие ложно.

## 5 Выводы

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



## **Список литературы**