

Отчёт по лабораторной работе № 11

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Кристина Алексеевна Антипина НБИбд-01-21

Содержание

Цель работы	1
Задания.....	1
Выполнение лабораторной работы	2
Вывод:	7
Ответы на контрольные вопросы:	7

–

Цель работы

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

Задания

1. Используя команды `grep`, написать 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`.
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

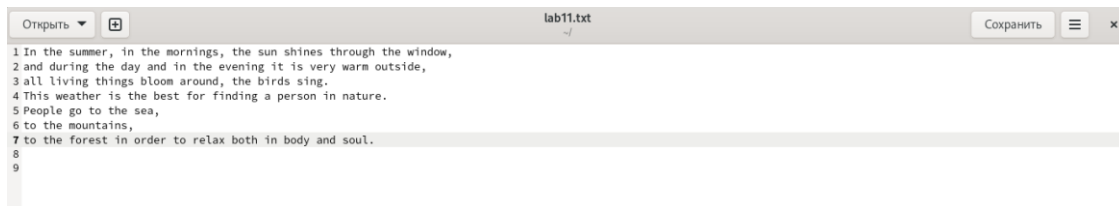
от 1 до 4 (например 1.tmp, 2.tmp, 3.tmp, 4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).

4. Написать командный файл, который с помощью команды `tar` запаковывает архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду `find`).

Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, напишу 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) (рис. - @fig:002) (рис. - @fig:003)



Вставляю в файл текст

Напишу первый скрипт

Проверяю в терминале

2. Напишу 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.(рис. -@fig:004)(рис. -@fig:005)(рис. -@fig:006)

```
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(2);
10    if (a==0) exit(4);
11    return 0;
12 }
```

Напишу новый скрипт на языке Си

```
#!/bin/bash

gcc program.c -o program
./program
code=$?
case $code in
0) echo "Число меньше 0";;
2) echo "Число больше 0";;
4) echo "Число равно 0";;
esac
```

U:--- **program.sh** All L1 (Shell-script[bash])
Welcome to [GNU Emacs](#), one component of the [GNU/Linux](#) operating system.

Emacs Tutorial	Learn basic keystroke commands (Учебник Emacs)
Emacs Guided Tour	Overview of Emacs features at gnu.org
View Emacs Manual	View the Emacs manual using Info
Absence of Warranty	GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions	Conditions for redistributing and changing Emacs
Ordering Manuals	Purchasing printed copies of manuals

To quit a partially entered command, type Control-g.

U:%%- ***GNU Emacs*** Top L3 (Fundamental)

Напишу второй скрипт

```
[kaantipina@fedora ~]$ touch program.c program.sh
[kaantipina@fedora ~]$ chmod +x *.sh
[kaantipina@fedora ~]$ ./program.sh
Введите число-3
Число меньше 0
[kaantipina@fedora ~]$ ./program.sh
Введите число8
Число больше 0
[kaantipina@fedora ~]$ ./program.sh
Введите число0
Число равно 0
```

Проверю все в терминале

3. Напишу 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).(рис. -@fig:007)(рис. -@fig:008)(рис. -@fig:009)

```
Обзор  Текстовый редактор  Сб, 21 мая 23:26  en  [иконки]
program2.sh
~/work/study/2021-2022/Операционные системы/laboratory/lab11
Открыть  Сохранить  [меню]  [крестик]

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 's' "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

sh  Ширина табуляции: 8  Стр 18, Стлб 6  ВСТ
```

Напишу новый скрипт

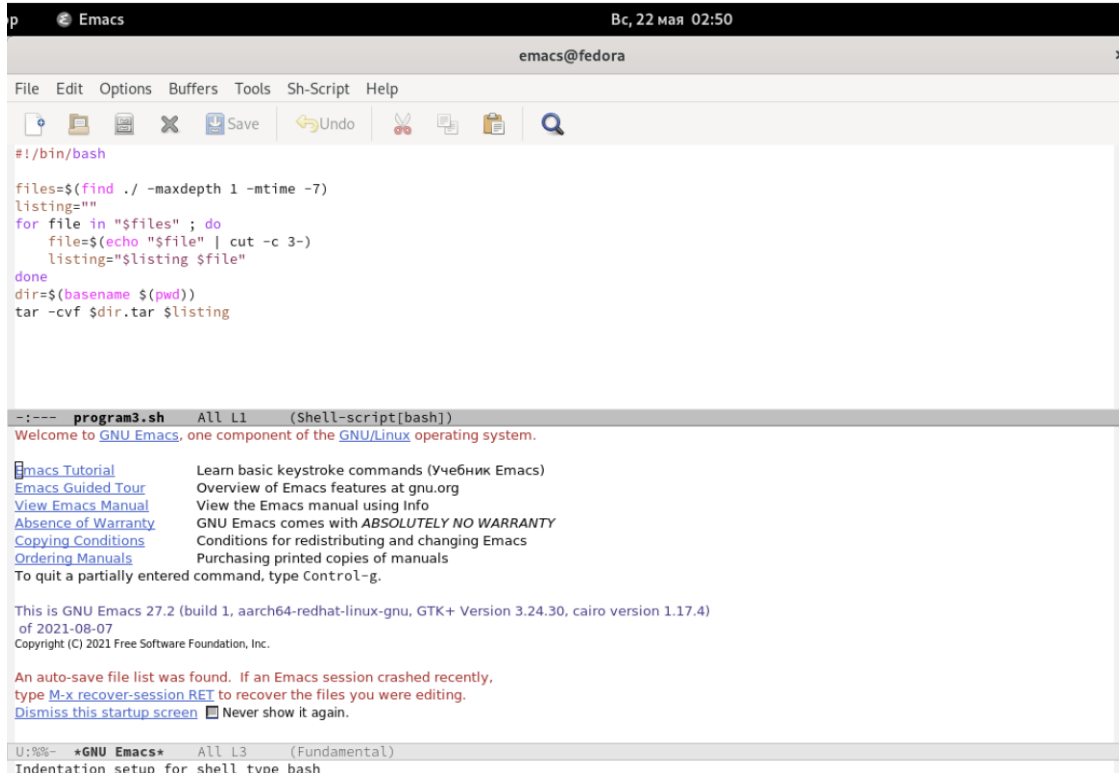
```
[kaantipina@fedora ~]$ touch program2.sh
[kaantipina@fedora ~]$ chmod +x *.sh
```

Проверю его в терминале

```
[kaantipina@fedora lab11]$ ls
program  program2.sh  program.c  program.sh
[kaantipina@fedora lab11]$ ls
program  program2.sh  program.c  program.sh
[kaantipina@fedora lab11]$ ./program2.sh -c a#.txt 3
[kaantipina@fedora lab11]$ ls
a1.txt  a2.txt  a3.txt  program  program2.sh  program.c  program.sh
[kaantipina@fedora lab11]$ ./program2.sh -r a#.txt 3
[kaantipina@fedora lab11]$ ls
program  program2.sh  program.c  program.sh
[kaantipina@fedora lab11]$
```

Проверю его в терминале

4. Напишу командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировав его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовав команду find).(рис. -@fig:010)(рис. -@fig:011)(рис. -@fig:012)



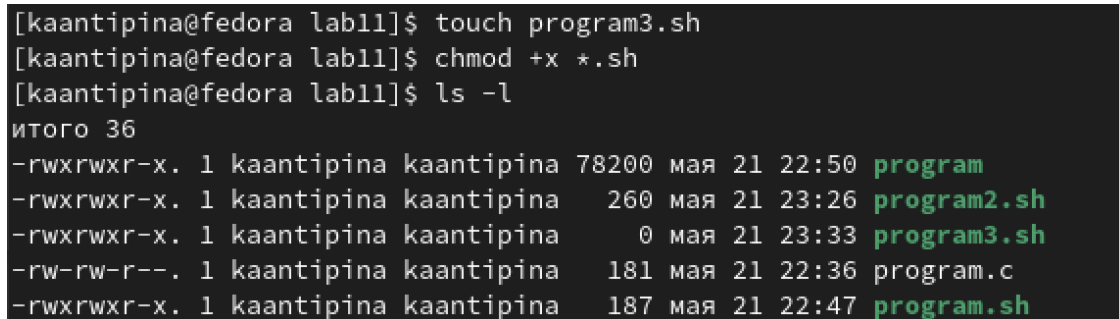
The screenshot shows the Emacs editor interface. The title bar indicates the window is titled 'emacs@fedora'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Sh-Script', and 'Help'. The toolbar contains icons for file operations and editing. The main text area displays a shell script named 'program3.sh' 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
```

Below the script, the Emacs startup screen is visible, showing the 'Welcome to GNU Emacs' message and various links for tutorials and manuals. The status bar at the bottom indicates the current buffer is 'program3.sh' and the shell type is 'bash'.

Напишу новый скрипт



The screenshot shows a terminal window with the following commands and output:

```
[kaantipina@fedora lab11]$ touch program3.sh
[kaantipina@fedora lab11]$ chmod +x *.sh
[kaantipina@fedora lab11]$ ls -l
итого 36
-rwxrwxr-x. 1 kaantipina kaantipina 78200 мая 21 22:50 program
-rwxrwxr-x. 1 kaantipina kaantipina 260 мая 21 23:26 program2.sh
-rwxrwxr-x. 1 kaantipina kaantipina 0 мая 21 23:33 program3.sh
-rw-rw-r--. 1 kaantipina kaantipina 181 мая 21 22:36 program.c
-rwxrwxr-x. 1 kaantipina kaantipina 187 мая 21 22:47 program.sh
```

Проверю его в терминале

```

[kaantipina@fedora f]$ sudo ~/f/program3.sh
[sudo] пароль для kaantipina:
program
program.c
program.sh
program2.sh
program3.sh
[kaantipina@fedora f]$ ls -l
итого 132
-rw-r--r--. 1 root      root      92160 мая 22 01:06 f.tar
-rwxrwxr-x. 1 kaantipina kaantipina 78200 мая 21 22:50 program
-rwxrwxr-x. 1 kaantipina kaantipina 260 мая 21 23:26 program2.sh
-rwxrwxrwx. 1 kaantipina kaantipina 211 мая 22 00:31 program3.sh
-rw-rw-r--. 1 kaantipina kaantipina 181 мая 21 22:36 program.c
-rwxrwxr-x. 1 kaantipina kaantipina 187 мая 21 22:47 program.sh
[kaantipina@fedora f]$ tar -tf f.tar
program
program.c
program.sh
program2.sh
program3.sh
[kaantipina@fedora f]$

```

Проверю его в терминале

Вывод:

В данной лабораторной работе № 11 я изучила основы программирования в оболочке ОС UNIX, научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

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 `mans/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.