

# **Львовторная работа №13**

**Операционные системы**

Краснова Камилла Геннадьевна

# 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>Выводы</b>	<b>13</b>
<b>5</b>	<b>Ответы на контрольные вопросы</b>	<b>14</b>

# Список иллюстраций

3.1	Создание файла . . . . .	7
3.2	код . . . . .	8
3.3	Результат . . . . .	8
3.4	Создание файла . . . . .	8
3.5	код . . . . .	9
3.6	код . . . . .	9
3.7	результат . . . . .	10
3.8	код . . . . .	10
3.9	результат . . . . .	11
3.10	код . . . . .	11
3.11	результат . . . . .	12

## **Список таблиц**

# 1 Цель работы

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; `-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  $n$  (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).
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

Создаю файл, который будет анализировать командную строку с ключами (рис. 3.1).

```
kamilla@fedora:~$ touch 1.sh
kamilla@fedora:~$ chmod +x 1.sh
kamilla@fedora:~$ gedit 1.sh
kamilla@fedora:~$ bash 1.sh -p эды -i input.txt -o output.txt -c -n
grep: input.txt: Нет такого файла или каталога
kamilla@fedora:~$ bash 1.sh -p эды -i text.txt -o output.txt -c -n
```

Рис. 3.1: Создание файла

Пример кода, который будет анализировать командную строку (рис. 3.2).

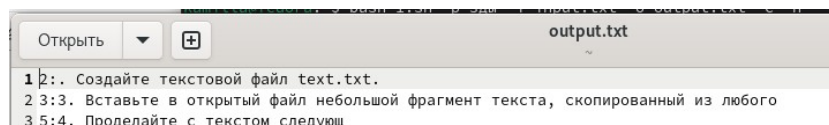
```

1 #!/bin/bash
2
3 while getopts i:o:p:cn optletter
4 do
5     case $optletter in
6         i) iflag=1; ival=$OPTARG;;
7         o) oflag=1; oval=$OPTARG;;
8         p) pflag=1; pval=$OPTARG;;
9         c) cflag=1;;
10        n) nflag=1;;
11        *) echo Illegal option $optletter;;
12    esac
13 done
14
15 if ! test $cflag
16 then
17     cf=-i
18 fi
19
20 if test $nflag
21 then
22     nf=-n
23 fi
24
25 grep $cf $nf $pval $ival >> $oval

```

Рис. 3.2: код

Результат выполнения команды (рис. 3.3).



The screenshot shows a terminal window with the title 'output.txt'. The content of the terminal is as follows:

```

1 2: . Создайте текстовый файл text.txt.
2 3:3. Вставьте в открытый файл небольшой фрагмент текста, скопированный из любого
3 5:4. Прodelайте с текстом следующ

```

Рис. 3.3: Результат

Создаю файл для программы на языке C, которая будет определять, является ли число больше нуля (рис. 3.4).



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

```

kamilla@fedora:~$ touch 2.sh
kamilla@fedora:~$ chmod +x 2.sh
kamilla@fedora:~$ touch 2.cpp
kamilla@fedora:~$ gedit 2.cpp

```

Рис. 3.4: Создание файла



Пример кода для этого пункта на Си (рис. 3.5).

```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int main () {
5     int n;
6     printf ("Введите число: ");
7     scanf ("%d", &n);
8     if(n>0){
9         exit(1);
10    }
11    else if (n==0) {
12        exit(0);
13    }
14    else {
15        exit(2);
16    }
17 }
```

Рис. 3.5: код

Код, анализирующий (рис. 3.6).

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

Рис. 3.6: код

Результат выполнения программы (рис. 3.7).

```
kamilla@fedora:~$ bash 2.sh
Введите число: 5
Число больше нуля
```

Рис. 3.7: результат

Создаю файл для третьего пункта, пишу программу, которая будет создавать и удалять файлы (рис. 3.8).

```
1 #!/bin/bash
2 for((i=1; i<=$*; i++))
3 do
4 if test -f "$i".tmp
5 then rm "$i".tmp
6 else touch "$i.tmp"
7 fi
8 done
```

Рис. 3.8: код

Результат выполнения (рис. 3.9).

```

kamilla@fedora:~$ bash 3.sh 4
kamilla@fedora:~$ ls
1.sh      Arseny      file.txt
1.tmp     australia   fun
2.c       backup      git-extens
2.cpp     bin         image
2.sh      conf.txt    image.zip
2.tmp     cprog      _index.md
3.sh      Documents   '#lab07.sh
3.tmp     doklad      lab07.sh
4.tmp     Downloads   lab07.sh~
abc1      feathers    LICENSE
kamilla@fedora:~$ bash 3.sh 4
kamilla@fedora:~$ ls
1.sh      conf.txt    ima
2.c       cprog      _i

```

Рис. 3.9: результат

Создаю файл для последней программы и вписываю в него код (рис. 3.10).

```

1 #! /bin/bash
2 find $* -mtime -7 -mtime +0 -type f > FILES.txt
3 tar -cf archive.tar -T FILES.txt

```

Рис. 3.10: код

Результат программы - архив файлов (рис. ??).

```
kamilla@fedora:~$ gedit 4.sh  
kamilla@fedora:~$ bash 4.sh  
kamilla@fedora:~$
```

Рис. 3.11: результат

## **4 Выводы**

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

## 5 Ответы на контрольные вопросы

### 1. Каково предназначение команды getopt?

Осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable`. Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` – это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Предположим, необходимо распознать командную строку следующего формата: `testprog -infile_in.txt -outfile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае: 

```
while
getopts o:i:Ltr optletter do
case optletter in
o) iflag = 1; oval =OPTARG;;
i) iflag=1;
ival=OPTARG;;
L) Lflag=1;;
t) tflag=1;;
r) rflag=1;;
*) echo Illegal option $optletter
esac
done
```

 Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента (будет равна `file_in.txt` для опции `i` и `file_out.doc` для опции `o`). `OPTIND` является числовым индексом на упомянутый аргумент. Функция `getopts` также понимает переменные типа массив, следовательно, можно

использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

## 2. Какое отношение метасимволы имеют к генерации имён файлов?

При перечислении имён файлов текущего каталога можно использовать следующие символы: – соответствует произвольной, в том числе и пустой строке; ? – соответствует любому одинарному символу; [с1-с2] – соответствует любому символу, лексикографически находящемуся между символами с1 и с2. Например, `echo *` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `ls .c` – выведет все файлы с последними двумя символами, совпадающими с `.c`. `echo prog.?` – выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.` [a-z] – соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

## 3. Какие операторы управления действиями вы знаете?

Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости отрезультатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути, являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.

#### 4. Какие операторы используются для прерывания цикла?

Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

#### 5. Для чего нужны команды `false` и `true`?

Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования `bash`: это команда `true`, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда `false`, которая всегда возвращает код завершения, не равный нулю (т. е. ложь).

#### 6. Что означает строка `if test -f mans/i.$s`, встречаемая в командном файле?

Строка `if test -f mans/i.s, mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).

#### 7. Объясните различия между конструкциями `while` и `until`.

Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда



из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.