

Отчёт по лабораторной работе №12

Операционные системы

Ильина Любовь Александровна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	12

Список таблиц

Список иллюстраций

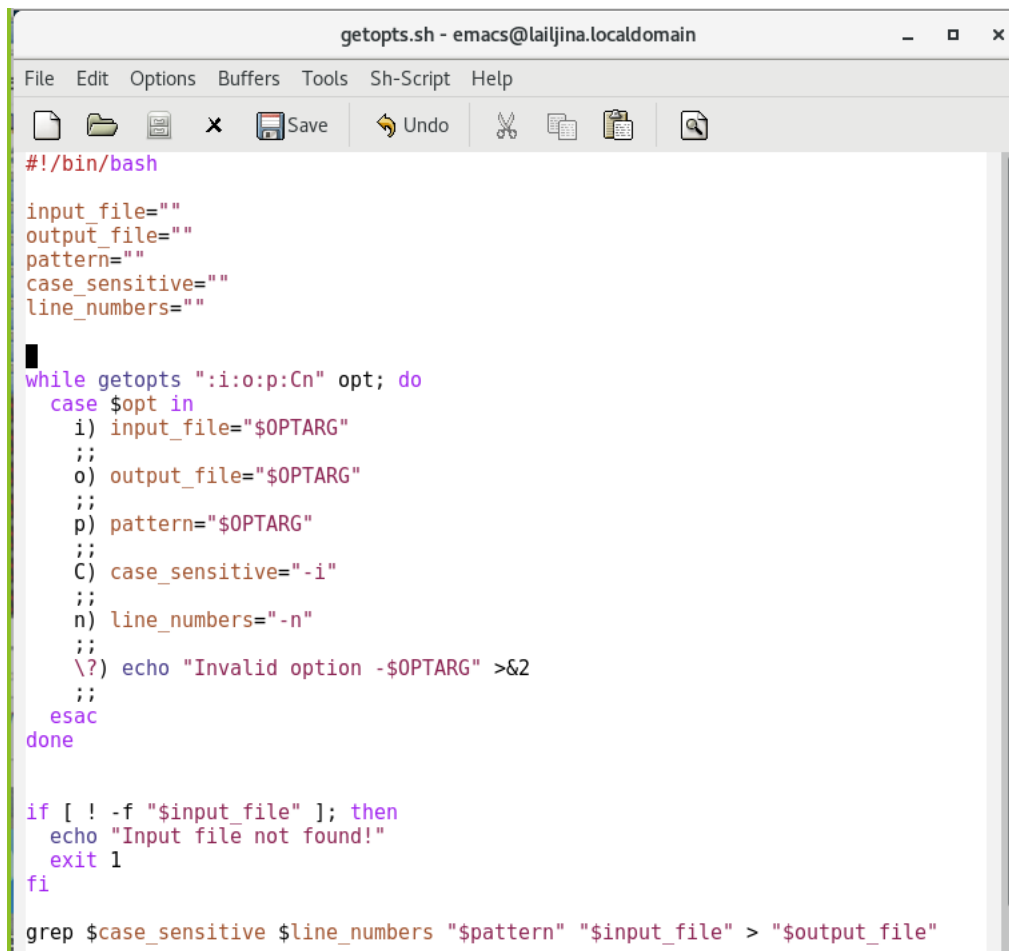
2.1 Командный файл	7
2.2 Проверка работы файла	7
2.3 Создание файлов	8
2.4 Код программы	8
2.5 Командный файл	8
2.6 Проверка работы файла	9
2.7 Командный файл	9
2.8 Проверка работы файла	9
2.9 Командный файл	10
2.10 Проверка работы файла	10

1. Цель работы

Изучение основ программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2. Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, напомним командный файл, который анализирует командную строку с ключами (рис. 2.1 - 2.2):
 - `-iinputfile` — прочитать данные из указанного файла;
 - `-ooutputfile` — вывести данные в указанный файл;
 - `-ршаблон` — 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`.



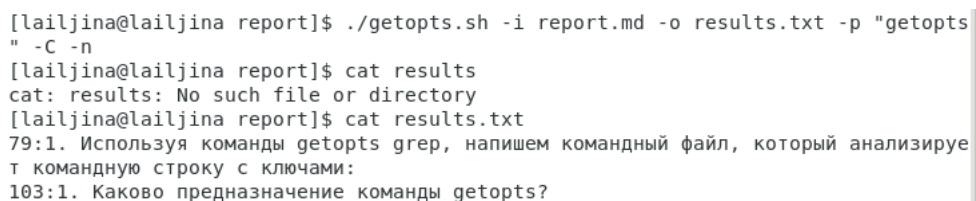
```
getopts.sh - emacs@lailjina.localdomain
File Edit Options Buffers Tools Sh-Script Help
Save Undo
#!/bin/bash
input_file=""
output_file=""
pattern=""
case_sensitive=""
line_numbers=""

while getopts ":i:o:p:Cn" opt; do
  case $opt in
    i) input_file="$OPTARG"
      ;;
    o) output_file="$OPTARG"
      ;;
    p) pattern="$OPTARG"
      ;;
    C) case_sensitive="-i"
      ;;
    n) line_numbers="-n"
      ;;
    \?) echo "Invalid option -$OPTARG" >&2
      ;;
  esac
done

if [ ! -f "$input_file" ]; then
  echo "Input file not found!"
  exit 1
fi

grep $case_sensitive $line_numbers "$pattern" "$input_file" > "$output_file"
```

Рис. 2.1: Командный файл



```
[lailjina@lailjina report]$ ./getopts.sh -i report.md -o results.txt -p "getopts" -C -n
[lailjina@lailjina report]$ cat results
cat: results: No such file or directory
[lailjina@lailjina report]$ cat results.txt
79:1. Используя команды getopts grep, напомним командный файл, который анализирует командную строку с ключами:
103:1. Каково предназначение команды getopts?
```

Рис. 2.2: Проверка работы файла

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. (рис. 2.3 - 2.5)

```
lailjina@lailjina report]$ vim vvod_chisla.c
lailjina@lailjina report]$ vim proverka_chisla.sh
lailjina@lailjina report]$ chmod u+x proverka_chisla.cpp
chmod: cannot access 'proverka_chisla.cpp': No such file or directory
lailjina@lailjina report]$ chmod u+x proverka_chisla.sh
```

Рис. 2.3: Создание файлов

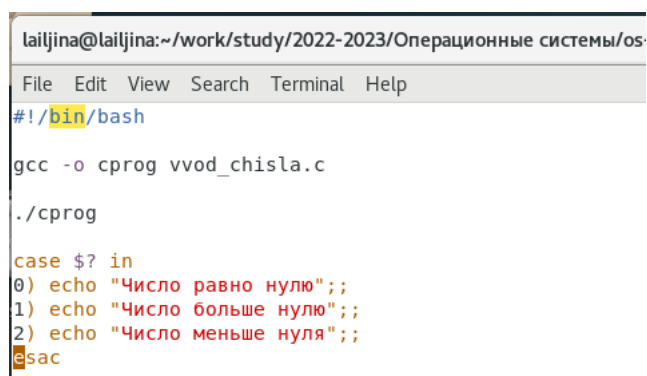


```
lailjina@lailjina:~/work/study/2022-2023/Опера
File Edit View Search Terminal Help
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int num;
    printf("Введите число: ");
    scanf("%d", &num);

    if (num > 0) {
        exit(1);
    } else if (num < 0) {
        exit(2);
    } else {
        exit(0);
    }
}
```

Рис. 2.4: Код программы



```
lailjina@lailjina:~/work/study/2022-2023/Операционные системы/os
File Edit View Search Terminal Help
#!/bin/bash

gcc -o cprog vvod_chisla.c

./cprog

case $? in
0) echo "Число равно нулю";;
1) echo "Число больше нуля";;
2) echo "Число меньше нуля";;
esac
```

Рис. 2.5: Командный файл

Проверяю работу файлов (рис. 2.6)


```
[lailjina@lailjina report]$ ./proverka_chisla.sh
Введите число: -4
Число меньше нуля
```

Рис. 2.6: Проверка работы файла

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). (рис. 2.7 - 2.8)

```
lailjina@lailjina:~/work/study/2022
File Edit View Search Termin
#!/bin/bash

N=$1
for ((i=1; i<=$N; i++))
do
if test -f "$i.tmp"
then rm "$i.tmp"
else touch "$i.tmp"
fi
done
~
~
```

Рис. 2.7: Командный файл

Проверяю работу файлов (рис. 2.8)

```
[lailjina@lailjina report]$ vim create_rm_files.sh
[lailjina@lailjina report]$ ./create_rm_files.sh 4
[lailjina@lailjina report]$ ls
1.tmp  4.tmp  create_rm_files.sh  Makefile          report.md
2.tmp  bib    getopt.sh          pandoc             results.txt
3.tmp  cprog  image             proverka_chisla.sh vvod_chisla.c
[lailjina@lailjina report]$ ./create_rm_files.sh 4
[lailjina@lailjina report]$ ls
bib      getopt.sh  pandoc      results.txt
cprog    image     proverka_chisla.sh  vvod_chisla.c
create_rm_files.sh  Makefile    report.md
```

Рис. 2.8: Проверка работы файла

4. Написать 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). (рис. 2.9 - 2.10)

```
lailjina@lailjina:~/work/study/2022-2023/Операционные системы/os-intro/labs/lab12
File Edit View Search Terminal Help
#!/bin/bash

directory="$1"

tar -cvzf archive.tar.gz $(find "$directory" -type f -mtime -7)
```

Рис. 2.9: Командный файл

Проверяю работу файлов (рис. 2.10)

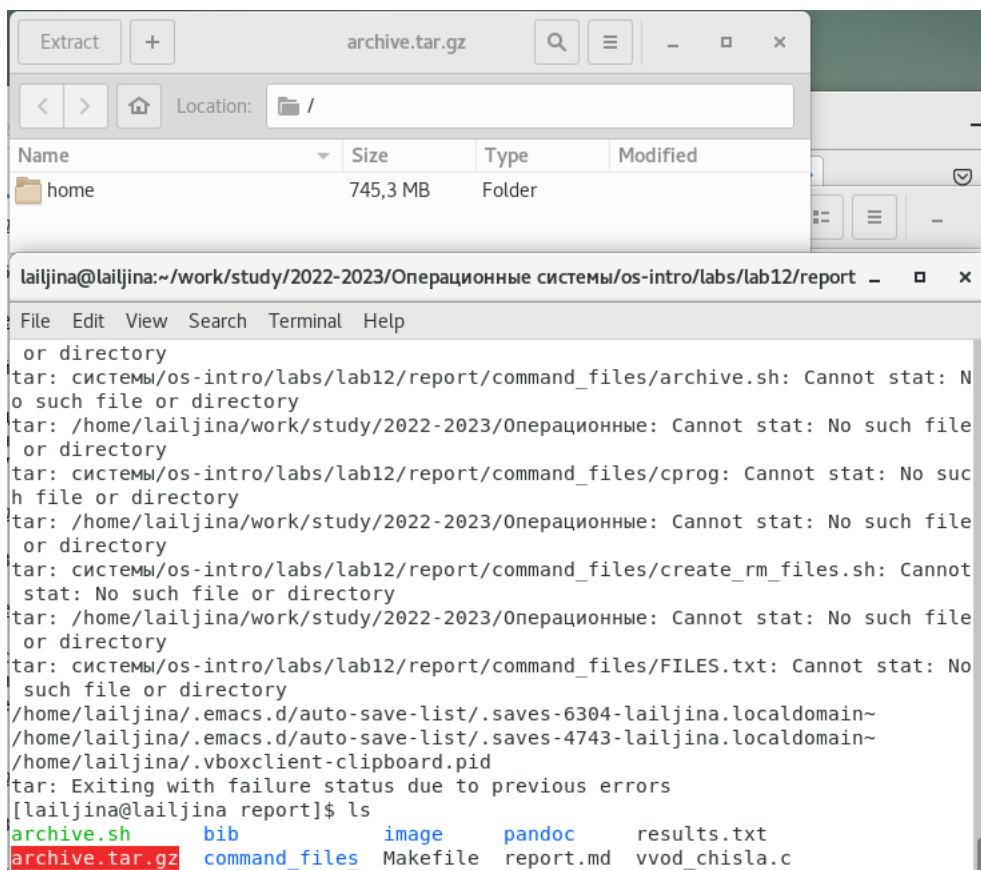


Рис. 2.10: Проверка работы файла

Контрольные вопросы

1. Каково предназначение команды `getopts`? `getopts` осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введённых пользователем данных.
2. Какое отношение метасимволы имеют к генерации имён файлов? Метасимволы позволяют генерировать имена файлов на основе заданного шаблона.
3. Какие операторы управления действиями вы знаете?
 - `test -f file` - возвращает нулевой код завершения (истина), если файл `file` существует, и ненулевой код завершения (ложь) в противном случае: – `test s` — истина, если аргумент `s` имеет значение истина; – `test -f file` — истина, если файл `file` существует; – `test -i file` — истина, если файл `file` доступен по чтению; – `test -w file` — истина, если файл `file` доступен по записи; – `test -e file` — истина, если файл `file` — исполняемая программа; – `test -d file` — истина, если файл `file` является каталогом.
4. Какие операторы используются для прерывания цикла? Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов.
5. Для чего нужны команды `false` и `true`? 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`, встреченная в командном файле? проверяет, существует ли файл с именем `man<значение переменной $s>/<значение переменной $i>`.<значение переменной \$s> в текущем каталоге
7. Объясните различия между конструкциями `while` и `until`. 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.

3. Выводы

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