

Отчёт по лабораторной работе №13

Операционные системы

Чистов Даниил Максимович

Содержание

1	Цель работы	4
2	Задания	5
3	Выполнение лабораторной работы	6
4	Программа 1	7
5	Программа 2	9
6	Программа 3	12
7	Программа 4	14
8	Выводы	15
	Список литературы	16

Список иллюстраций

4.1	Задание 1 - выполнено	7
4.2	Задание 1 - программа	8
5.1	Задание 2 - выполнено	9
5.2	Задание 2 - программа 1	10
5.3	Задание 2 - программа 2	11
6.1	Задание 3 - выполнено	12
6.2	Задание 3 - программа	13
7.1	Задание 4 - готово	14
7.2	Задание 4 - программа	14

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

2 Задания

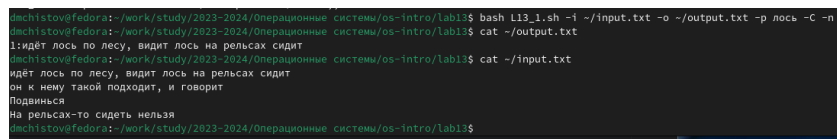
1. Программа 1
2. Программа 2
3. Программа 3
4. Программа 4

3 Выполнение лабораторной работы

4 Программа 1

Задание: “Используя команды `getopts` `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`.”

Приступаю к выполнению работы. Создаю файл с кодом и пишу код (сама программа будет показана позже), выполняю - готово (рис. 4.1).



```
lmchistov@fedora: ~/work/study/2023-2024/Операционные системы/os-intro/lab1$ bash L13_1.sh -i ~/input.txt -o ~/output.txt -p лось -C -n
lmchistov@fedora: ~/work/study/2023-2024/Операционные системы/os-intro/lab1$ cat ~/output.txt
1идёт лось по лесу, видит лось на рельсах сидит
2идёт лось по лесу, видит лось на рельсах сидит
3он к нему такой подходит, и говорит
4Подвинься
5на рельсах-то сидеть нельзя
lmchistov@fedora: ~/work/study/2023-2024/Операционные системы/os-intro/lab1$
```

Рис. 4.1: Задание 1 - выполнено

Код программы 1: утилитой `getopts` считываем коды, а дальше утилитой `case` рассматриваем каждый случай. Затем утилитой `grep` считываем и используем `output.txt` (рис. 4.2).

```

1  #!/bin/bash
2
3  while getopts ":i:o:p:Cn" opt;
4  do
5      case ${opt} in
6          i ) iinputfile=$OPTARG;;
7          o ) ooutputfile=$OPTARG;;
8          p ) pattern=$OPTARG;;
9          C ) case_sens="-i";;
10         n ) s_num="-n";;
11         * ) echo "weird operation" exit 1;;
12     esac
13 done
14
15 grep $case_sens $s_num "$pattern" "$iinputfile" > "$ooutputfile"
16

```

Рис. 4.2: Задание 1 - программа

5 Программа 2

Задание: “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. Команд ный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено”

Приступаю к выполнению - создаю, даю права, пишу код, исправляю ошибки, запускаю - работает (рис. 5.1).

```
dmchistov@fedora: ~/work/study/2023-2024/Операционные системы/os-intro/lab13$ bash l13_2_1.sh
Enter the number: 4
Число больше нуля
dmchistov@fedora: ~/work/study/2023-2024/Операционные системы/os-intro/lab13$ bash l13_2_1.sh
Enter the number: -1
Число меньше нуля
dmchistov@fedora: ~/work/study/2023-2024/Операционные системы/os-intro/lab13$ bash l13_2_1.sh
Enter the number: 0
Число равно нулю
dmchistov@fedora: ~/work/study/2023-2024/Операционные системы/os-intro/lab13$
```

Рис. 5.1: Задание 2 - выполнено

Код программы 2_1 (bash): открываем файл с кодом на C, запускаем его и получаем код, утилитой `case` в зависимости от кода возвращаем нужное сообщение (рис. 5.2).

```
1  #!/bin/bash
2
3  gcc -o cprog L13_2_2.cpp
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. 5.2: Задание 2 - программа 1

Программа 2_2: Если число больше нуля возвращаем 1, если меньше - 2, если равно - 0 (рис. 5.3).

```

1  #include <stdlib.h>
2  #include <stdio.h>
3
4  int main()
5  {
6      int n;
7      printf("Enter the numner: ");
8      scanf("%d", &n);
9      if (n > 0){
10         exit(1);
11     }
12     else if (n == 0){
13         exit(0);
14     }
15     else {
16         exit(2);
17     }
18
19
20
21 }

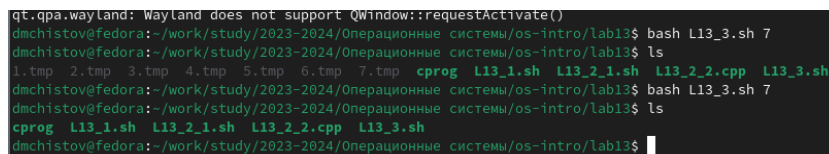
```

Рис. 5.3: Задание 2 - программа 2

6 Программа 3

Задание: “Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до (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).”

Создаю файл, даю права, начинаю писать код, всё работает (рис. 6.1).



```
qt.qpa.wayland: Wayland does not support QWindow::requestActivate()
dmchistov@fedora:~/work/study/2023-2024/Операционные системы/os-intro/lab13$ bash L13_3.sh 7
dmchistov@fedora:~/work/study/2023-2024/Операционные системы/os-intro/lab13$ ls
1.tmp 2.tmp 3.tmp 4.tmp 5.tmp 6.tmp 7.tmp cprog L13_1.sh L13_2_1.sh L13_2_2.cpp L13_3.sh
dmchistov@fedora:~/work/study/2023-2024/Операционные системы/os-intro/lab13$ bash L13_3.sh 7
dmchistov@fedora:~/work/study/2023-2024/Операционные системы/os-intro/lab13$ ls
cprog L13_1.sh L13_2_1.sh L13_2_2.cpp L13_3.sh
dmchistov@fedora:~/work/study/2023-2024/Операционные системы/os-intro/lab13$
```

Рис. 6.1: Задание 3 - выполнено

Код программы 3: Иду по циклу от 1 до заданного числа, проверяю утилитой test есть ли такие файлы под названием “Номер”.tmp. Если есть удаляю, если нет, создаю (рис. 6.2).

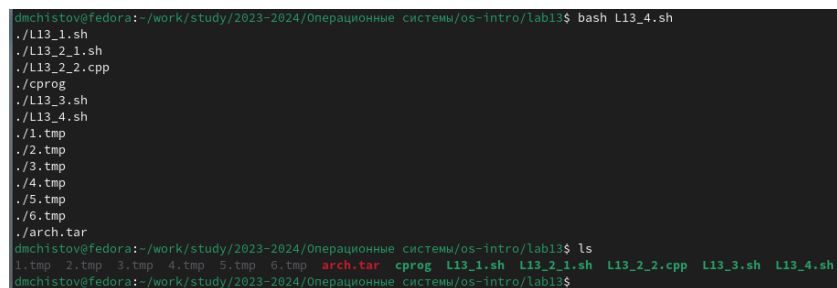
```
1  #!/bin/bash
2
3  for((i=1; i<=*$*; i++))
4  do
5  if test -f "$i".tmp
6  then rm "$i".tmp
7  else touch "$i".tmp
8  fi
9  done
10
```

Рис. 6.2: Задание 3 - программа

7 Программа 4

Задание: “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).”

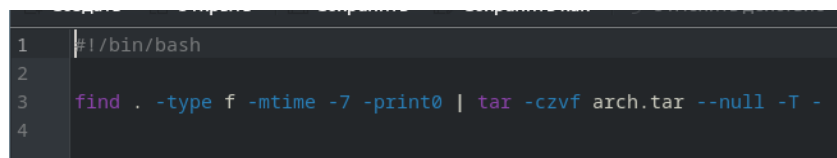
Создаю файл, даю права, начинаю писать код. Готово (рис. 7.1).



```
dmchistov@fedora:~/work/study/2023-2024/Операционные системы/os-intro/lab13$ bash L13_4.sh
./L13_1.sh
./L13_2_1.sh
./L13_2_2.cpp
./cprog
./L13_3.sh
./L13_4.sh
./1.tmp
./2.tmp
./3.tmp
./4.tmp
./5.tmp
./6.tmp
./arch.tar
dmchistov@fedora:~/work/study/2023-2024/Операционные системы/os-intro/lab13$ ls
1.tmp 2.tmp 3.tmp 4.tmp 5.tmp 6.tmp arch.tar cprog L13_1.sh L13_2_1.sh L13_2_2.cpp L13_3.sh L13_4.sh
dmchistov@fedora:~/work/study/2023-2024/Операционные системы/os-intro/lab13$
```

Рис. 7.1: Задание 4 - готово

Код программы 4: Использую конвеер. В первой части ищу утилитой find все файлы, которые были модифицированы менее 7 дней назад (-mtime -7), вывожу их. Во второй части принимаю этот вывод, архивирую (рис. 7.2).



```
1 #!/bin/bash
2
3 find . -type f -mtime -7 -print0 | tar -czvf arch.tar --null -T -
4
```

Рис. 7.2: Задание 4 - программа

8 Выводы

В результате выполнения данной работы я изучил основы программирования в оболочке ОС UNIX/Linux. Научился писать небольшие командные файлы.

Список литературы

Лабораторная работы №13 Команда find