

Лабораторная работа №11

Дисциплина: Операционные системы

Гибшер Кирилл Владимирович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
4	Выполнение лабораторной работы	10
5	Вывод	17

Список иллюстраций

4.1	Скрипт первого задания	10
4.2	Запуск	11
4.3	input.txt	11
4.4	output.txt	12
4.5	Файл си	12
4.6	Командный файл	13
4.7	Запуск и проверка работоспособности кода	13
4.8	Скрипт третьего задания	14
4.9	Запуск и проверка	14
4.10	Скрипт последнего задания	15
4.11	Запуск исполняемого файла	15
4.12	Результат	16

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;
- `-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`

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 ∞ (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 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 Теоретическое введение

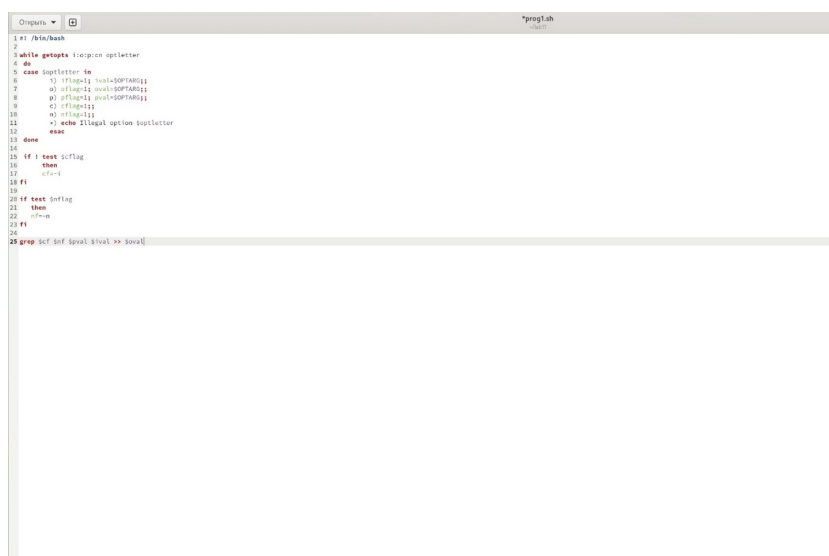
Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: - оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;

- C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).
- POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим

основные элементы программирования в оболочке `bash`. В других оболочках большинство команд будет совпадать с описанными ниже.

4 Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, напомним скрипт для первого задания (рис. [4.1])



```
1 #!/bin/bash
2
3 while getopts :f:p:c:n optletter
4 do
5     case $optletter in
6         f) iflag=$OPTARG ;;
7         p) pflag=$OPTARG ;;
8         c) cflag=$OPTARG ;;
9         n) nflag=$OPTARG ;;
10        *) echo "Illegal option $optletter"
11           ;;
12    esac
13 done
14
15 if [ -f $iflag ]
16 then
17     echo "f"
18 fi
19
20 if [ -f $pflag ]
21 then
22     echo "p"
23 fi
24
25 grep -c $cflag $nflag
```

Рис. 4.1: Скрипт первого задания

2. Запустим командный файл и проверим его работоспособность, прописав соответствующие опции в команде. Результат работы проверим в файле `output` (рис. [4.2])

```
[kvgibsher@kvgibsher lab11]$ bash prog1.sh -p ку -i input.txt -o output.txt -cn  
[kvgibsher@kvgibsher lab11]$
```

Рис. 4.2: Запуск

3. Содержимое файла input (рис. [4.3])

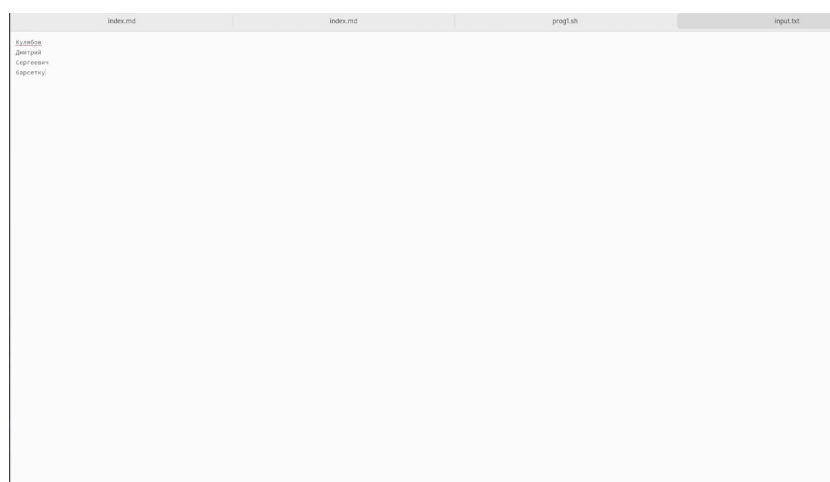


Рис. 4.3: input.txt

4. Содержимое файла output (рис. [4.4])

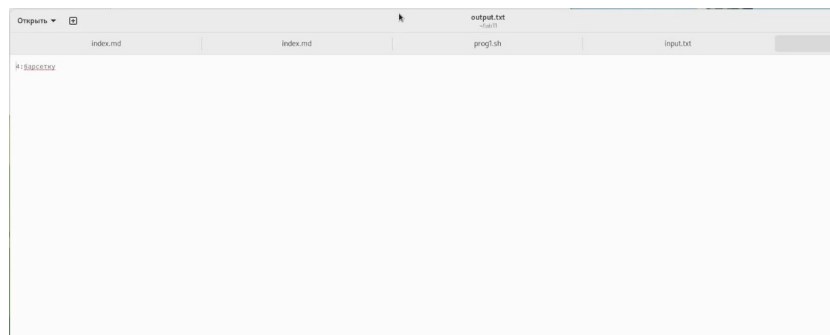


Рис. 4.4: output.txt

5. Напишем код для файла с поддержкой Си (рис. [4.5])

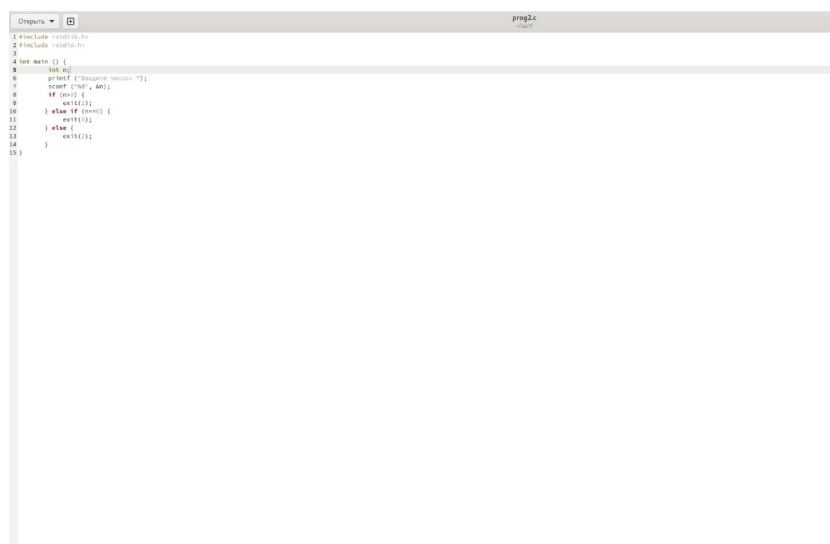


Рис. 4.5: Файл си

6. Используя ранее написанный код СИ , напишем скрипт для командного файла, который будет удовлетворять условия задания 2 (рис. [4.6])

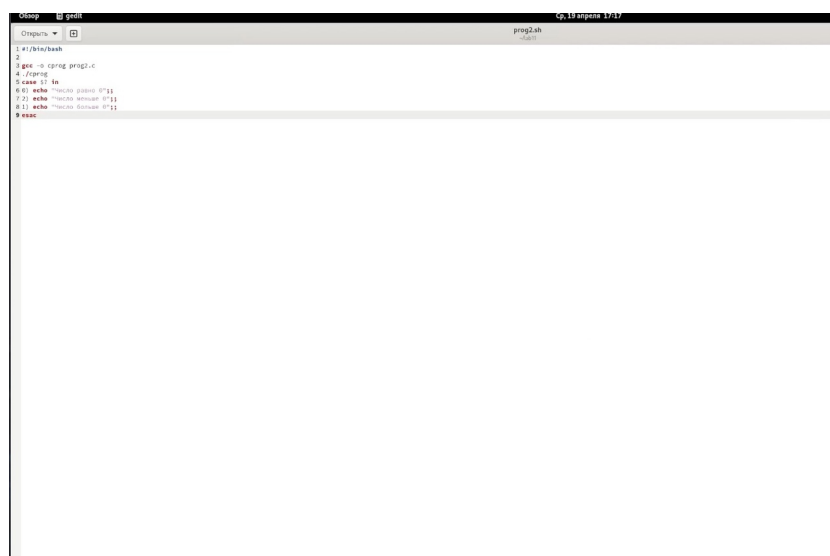


Рис. 4.6: Командный файл

7. Запуск командного файла и проверка работоспособности кода. Действительно, все необходимые проверки над вводимыми пользователем числами проводятся. Задание выполнено (рис. [4.7])

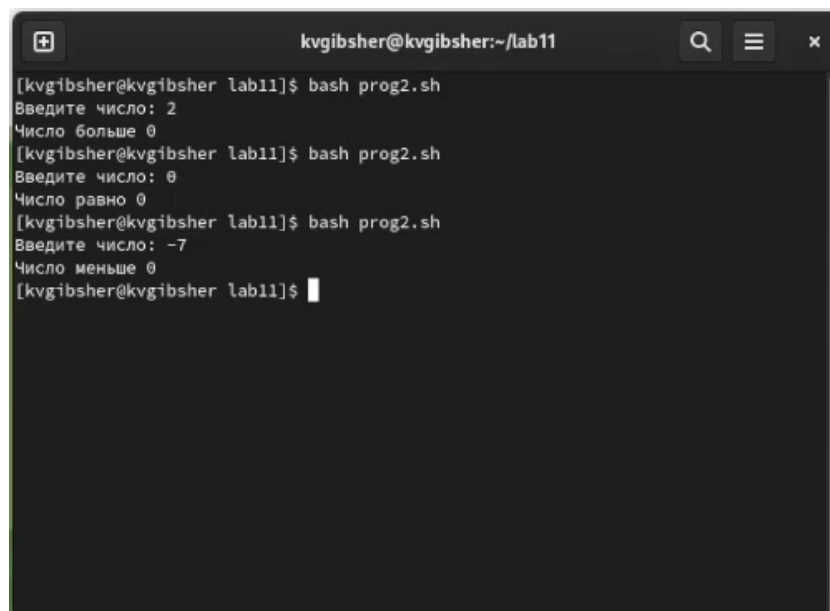
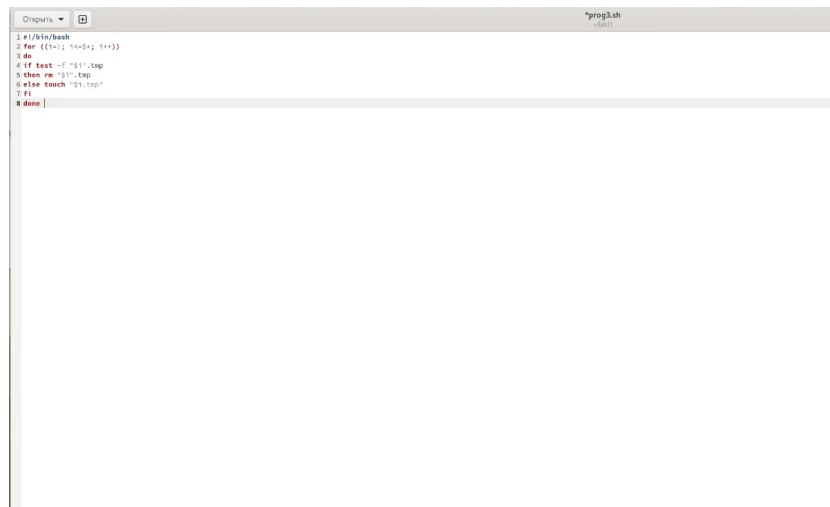


Рис. 4.7: Запуск и проверка работоспособности кода

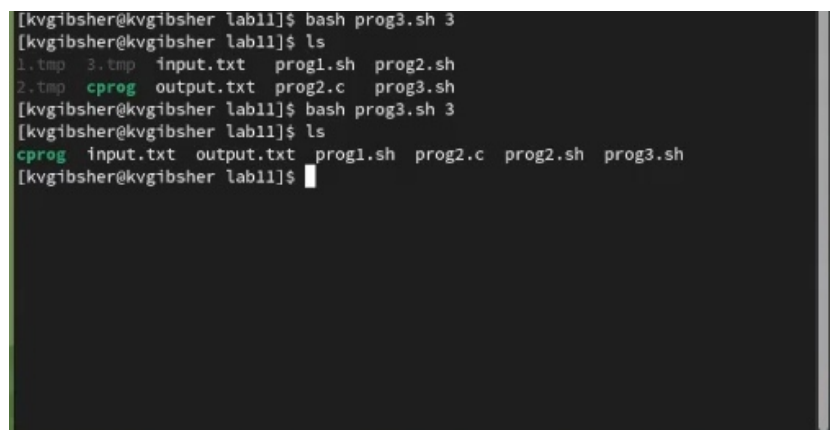
8. Напишем скрипт для третьего задания (рис. [4.8])



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

Рис. 4.8: Скрипт третьего задания

9. Запустим командный файл и увидим, что при первом запуске создается 3 файла tmp и при повторном запуске, созданные им файлы удаляются, тем самым программа соответствует условиям задания. (рис. [4.9])



```
[kvgibsher@kvgibsher lab11]$ bash prog3.sh 3
[kvgibsher@kvgibsher lab11]$ ls
1.tmp 3.tmp input.txt prog1.sh prog2.sh
2.tmp cprog output.txt prog2.c prog3.sh
[kvgibsher@kvgibsher lab11]$ bash prog3.sh 3
[kvgibsher@kvgibsher lab11]$ ls
cprog input.txt output.txt prog1.sh prog2.c prog2.sh prog3.sh
[kvgibsher@kvgibsher lab11]$
```

Рис. 4.9: Запуск и проверка

10. Напишем скрипт для последнего задания (рис. [4.10])



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

Рис. 4.10: Скрипт последнего задания

11. Запустим командный файл и увидим, что при запуске архив и файл FILES.txt были успешно созданы. (рис. [4.11])

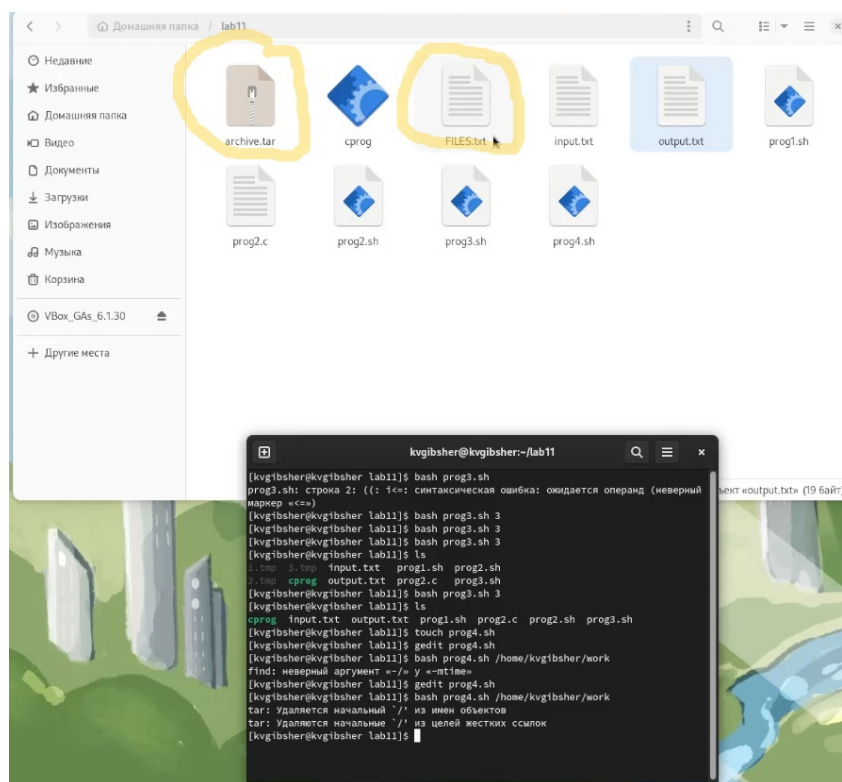


Рис. 4.11: Запуск исполняемого файла

12. Проверим результат, открыв файл FILES.txt , в которой и указываются все архивированные файлы (рис. [4.12])

index	index.md	project	input.txt	output.txt	FILE SIZE
1	./index/1/	./index/1/	./index/1/	./index/1/	1.00 MB
2	./index/2/	./index/2/	./index/2/	./index/2/	2.00 MB
3	./index/3/	./index/3/	./index/3/	./index/3/	3.00 MB
4	./index/4/	./index/4/	./index/4/	./index/4/	4.00 MB
5	./index/5/	./index/5/	./index/5/	./index/5/	5.00 MB
6	./index/6/	./index/6/	./index/6/	./index/6/	6.00 MB
7	./index/7/	./index/7/	./index/7/	./index/7/	7.00 MB
8	./index/8/	./index/8/	./index/8/	./index/8/	8.00 MB
9	./index/9/	./index/9/	./index/9/	./index/9/	9.00 MB
10	./index/10/	./index/10/	./index/10/	./index/10/	10.00 MB
11	./index/11/	./index/11/	./index/11/	./index/11/	11.00 MB
12	./index/12/	./index/12/	./index/12/	./index/12/	12.00 MB
13	./index/13/	./index/13/	./index/13/	./index/13/	13.00 MB
14	./index/14/	./index/14/	./index/14/	./index/14/	14.00 MB
15	./index/15/	./index/15/	./index/15/	./index/15/	15.00 MB
16	./index/16/	./index/16/	./index/16/	./index/16/	16.00 MB
17	./index/17/	./index/17/	./index/17/	./index/17/	17.00 MB
18	./index/18/	./index/18/	./index/18/	./index/18/	18.00 MB
19	./index/19/	./index/19/	./index/19/	./index/19/	19.00 MB
20	./index/20/	./index/20/	./index/20/	./index/20/	20.00 MB
21	./index/21/	./index/21/	./index/21/	./index/21/	21.00 MB
22	./index/22/	./index/22/	./index/22/	./index/22/	22.00 MB
23	./index/23/	./index/23/	./index/23/	./index/23/	23.00 MB
24	./index/24/	./index/24/	./index/24/	./index/24/	24.00 MB
25	./index/25/	./index/25/	./index/25/	./index/25/	25.00 MB
26	./index/26/	./index/26/	./index/26/	./index/26/	26.00 MB
27	./index/27/	./index/27/	./index/27/	./index/27/	27.00 MB
28	./index/28/	./index/28/	./index/28/	./index/28/	28.00 MB
29	./index/29/	./index/29/	./index/29/	./index/29/	29.00 MB
30	./index/30/	./index/30/	./index/30/	./index/30/	30.00 MB
31	./index/31/	./index/31/	./index/31/	./index/31/	31.00 MB
32	./index/32/	./index/32/	./index/32/	./index/32/	32.00 MB
33	./index/33/	./index/33/	./index/33/	./index/33/	33.00 MB
34	./index/34/	./index/34/	./index/34/	./index/34/	34.00 MB
35	./index/35/	./index/35/	./index/35/	./index/35/	35.00 MB
36	./index/36/	./index/36/	./index/36/	./index/36/	36.00 MB
37	./index/37/	./index/37/	./index/37/	./index/37/	37.00 MB
38	./index/38/	./index/38/	./index/38/	./index/38/	38.00 MB
39	./index/39/	./index/39/	./index/39/	./index/39/	39.00 MB
40	./index/40/	./index/40/	./index/40/	./index/40/	40.00 MB
41	./index/41/	./index/41/	./index/41/	./index/41/	41.00 MB
42	./index/42/	./index/42/	./index/42/	./index/42/	42.00 MB
43	./index/43/	./index/43/	./index/43/	./index/43/	43.00 MB
44	./index/44/	./index/44/	./index/44/	./index/44/	44.00 MB
45	./index/45/	./index/45/	./index/45/	./index/45/	45.00 MB
46	./index/46/	./index/46/	./index/46/	./index/46/	46.00 MB
47	./index/47/	./index/47/	./index/47/	./index/47/	47.00 MB
48	./index/48/	./index/48/	./index/48/	./index/48/	48.00 MB
49	./index/49/	./index/49/	./index/49/	./index/49/	49.00 MB
50	./index/50/	./index/50/	./index/50/	./index/50/	50.00 MB
51	./index/51/	./index/51/	./index/51/	./index/51/	51.00 MB
52	./index/52/	./index/52/	./index/52/	./index/52/	52.00 MB
53	./index/53/	./index/53/	./index/53/	./index/53/	53.00 MB
54	./index/54/	./index/54/	./index/54/	./index/54/	54.00 MB
55	./index/55/	./index/55/	./index/55/	./index/55/	55.00 MB
56	./index/56/	./index/56/	./index/56/	./index/56/	56.00 MB
57	./index/57/	./index/57/	./index/57/	./index/57/	57.00 MB
58	./index/58/	./index/58/	./index/58/	./index/58/	58.00 MB
59	./index/59/	./index/59/	./index/59/	./index/59/	59.00 MB
60	./index/60/	./index/60/	./index/60/	./index/60/	60.00 MB
61	./index/61/	./index/61/	./index/61/	./index/61/	61.00 MB

Рис. 4.12: Результат

5 Вывод

Я изучил основы программирования в оболочке ОС UNIX. Научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.