

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №7**

**«Поиск файлов.**

**Перенаправление ввода-вывода.**

**Просмотр запущенных процессов»**

дисциплина: Операционные системы

Студентка:

Бочкарева Елена Дмитриевна

Студенческий билет номер №: 1032207514

Группа:

НПМбв-01-19

**МОСКВА**

2023

## **Содержание:**

Цель работы.....	4
Задания. Последовательность выполнения лабораторной работы.....	4
Пункт 1.....	5
Пункт 2.....	6-8
Пункт 3.....	9
Пункт 4.....	9
Пункт 5.....	9-10
Пункт 6.....	10-11
Пункт 7.....	11
Пункт 8.....	12
Пункт 9.....	13
Пункт 10.....	13-14
Пункт 11.....	14-18
Пункт 12.....	18-20
Ответы на контрольные вопросы.....	21-30
Выводы, согласованные с целью работы.....	31

## Список иллюстраций:

1 Рисунок 1.....	5
2 Рисунок 2.....	5
3 Рисунок 3.....	6
4 Рисунок 4.....	6
5 Рисунок 5.....	7
6 Рисунок 6.....	7
7 Рисунок 7.....	8
8 Рисунок 8.....	8
9 Рисунок 9.....	9
10 Рисунок 10.....	9
11 Рисунок 11.....	9
12 Рисунок 12.....	9
13 Рисунок 13.....	10
14 Рисунок 14.....	10
15 Рисунок 15.....	11
16 Рисунок 16.....	11
17 Рисунок 17.....	11
18 Рисунок 18.....	12
19 Рисунок 19.....	12
20 Рисунок 20.....	13
21 Рисунок 21.....	13
22 Рисунок 22.....	14
23 Рисунок 23.....	14
24 Рисунок 24.....	14
25 Рисунок 25.....	15
26 Рисунок 26.....	15
27 Рисунок 27.....	16
28 Рисунок 28.....	16
29 Рисунок 29.....	17
30 Рисунок 30.....	17
31 Рисунок 31.....	18
32 Рисунок 32.....	18
33 Рисунок 33.....	18
34 Рисунок 34.....	19
35 Рисунок 35.....	19
36 Рисунок 36.....	20

**Цель работы:** ознакомление с инструментами поиска файлов и фильтрации текстовых данных. Приобретение практических навыков: по управлению процессами (и заданиями), по проверке и использования диска и обслуживанию файловых систем.

**Задания: 4.3. Последовательность выполнения работы.**

1. Осуществите вход в систему, используя соответствующее имя пользователя.
2. Запишите в файл `file.txt` названия файлов, содержащихся в каталоге `/etc`. Допишите в этот же файл названия файлов, содержащихся в вашем домашнем каталоге.
3. Выведите имена всех файлов из `file.txt`, имеющих расширение `.conf`, после чего запишите их в новый текстовый файл `conf.txt`.
4. Определите, какие файлы в вашем домашнем каталоге имеют имена, начинавшиеся с символа `s`? Предложите несколько вариантов, как это сделать.
5. Выведите на экран (постранично) имена файлов из каталога `/etc`, начинающиеся с символа `h`.
6. Запустите в фоновом режиме процесс, который будет записывать в файл `~/logfile` файлы, имена которых начинаются с `log`.
7. Удалите файл `~/logfile`.
8. Запустите из консоли в фоновом режиме редактор `gedit`.
9. Определите идентификатор процесса `gedit`, используя команду `ps`, конвейер и фильтр `grep`. Можно ли определить этот идентификатор более простым способом?
10. Прочтите справку (`man`) команды `kill`, после чего используйте её для завершения процесса `gedit`.
11. Выполните команды `df` и `du`, предварительно получив более подробную информацию об этих командах, с помощью команды `man`.
12. Воспользовавшись справкой команды `find`, выведите имена всех директорий, имеющихся в вашем домашнем каталоге.

## Результаты выполнения задания:

### Пункт 1. Осуществите вход в систему, используя соответствующее имя пользователя.

Вход в систему осуществляю в графическом интерфейсе под именем пользователя **edbochkareva** (рис.1).

#### 1.1. Запускаем операционную систему:

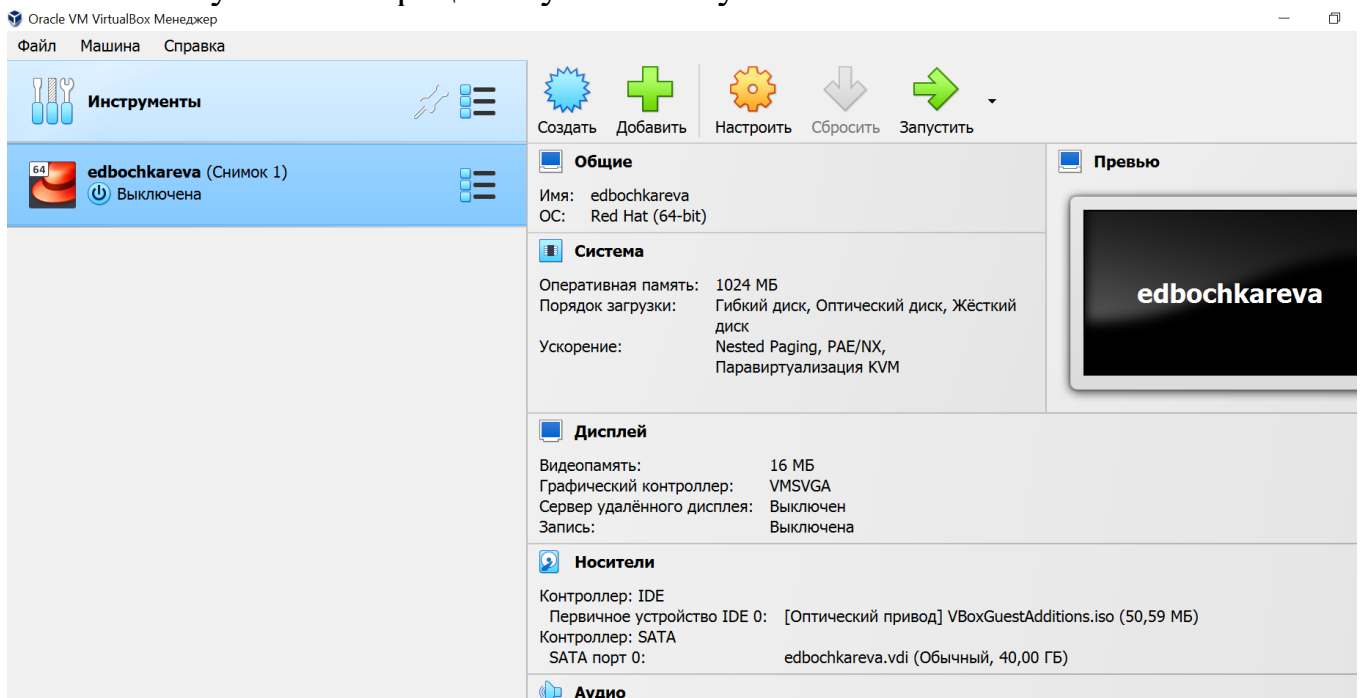


Рис.1: Рисунок 1

#### 1.2. Вхожу от имени пользователя edbochkareva. Ввожу пароль (рис.2).

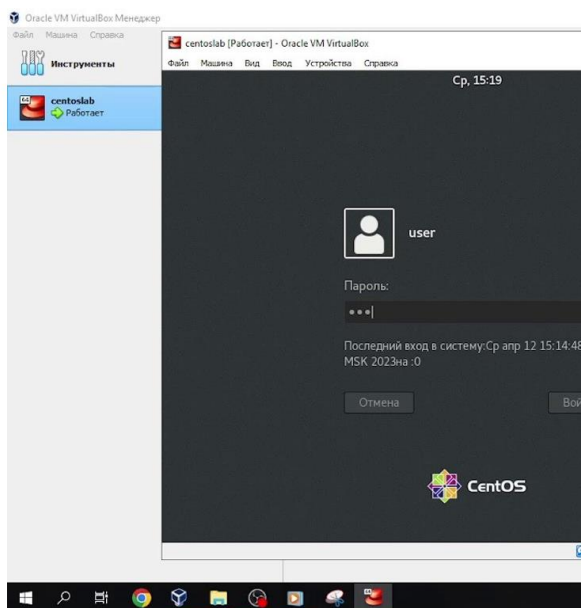
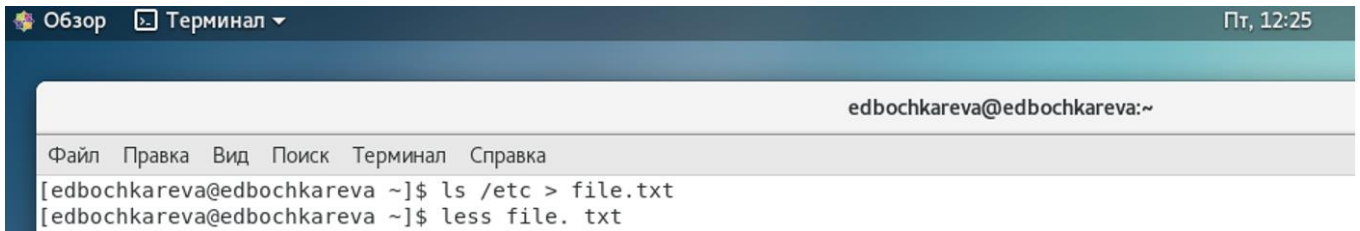


Рис.2: Рисунок 2

**Пункт 2. Запишите в файл file.txt названия файлов, содержащихся в каталоге /etc. Допишите в этот же файл названия файлов, содержащихся в вашем домашнем каталоге.**

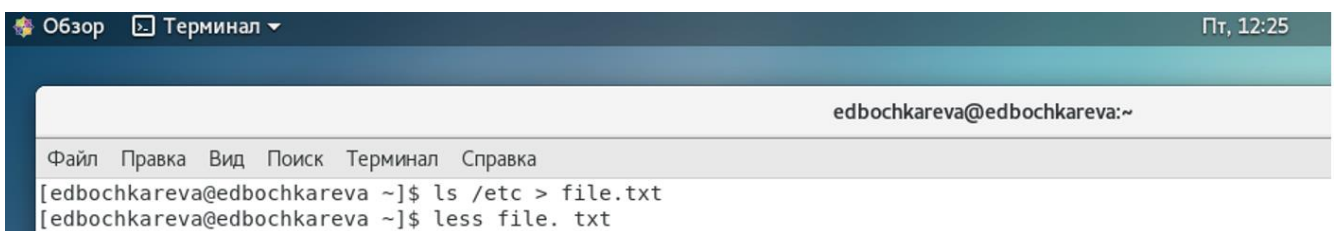
Для того, чтобы сохранить список файлов из каталога /etc в файле file.txt, выполняю команду: **ls /etc > file.txt** (рис.3).



```
Обзор Терминал ▼ Пт, 12:25
edbochkareva@edbochkareva:~
Файл Правка Вид Поиск Терминал Справка
[edbochkareva@edbochkareva ~]$ ls /etc > file.txt
[edbochkareva@edbochkareva ~]$ less file.txt
```

Рис.3: Рисунок 3

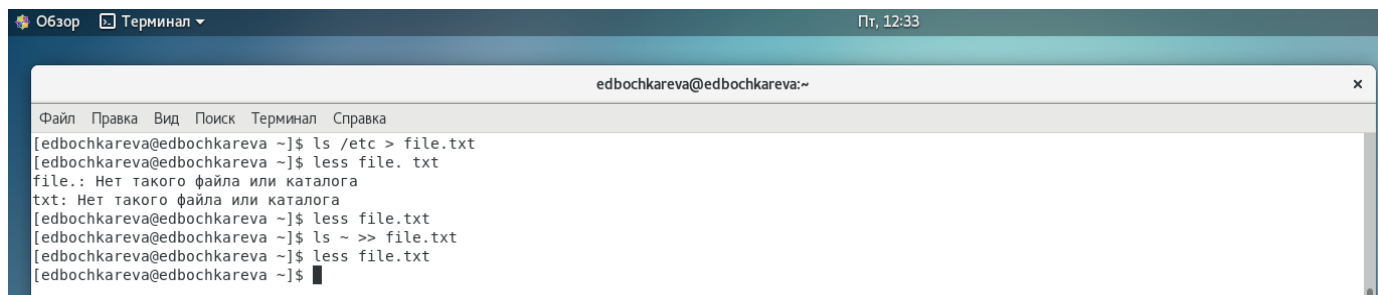
Ввод команды для сохранения в файл содержимого /etc (рис.4).



```
Обзор Терминал ▼ Пт, 12:25
edbochkareva@edbochkareva:~
Файл Правка Вид Поиск Терминал Справка
[edbochkareva@edbochkareva ~]$ ls /etc > file.txt
[edbochkareva@edbochkareva ~]$ less file.txt
```

Рис.4: Рисунок 4

Для того, чтобы добавить в файл file.txt список файлов из домашнего каталога, выполним команду: **ls ~ >> file.txt** (рис.5).



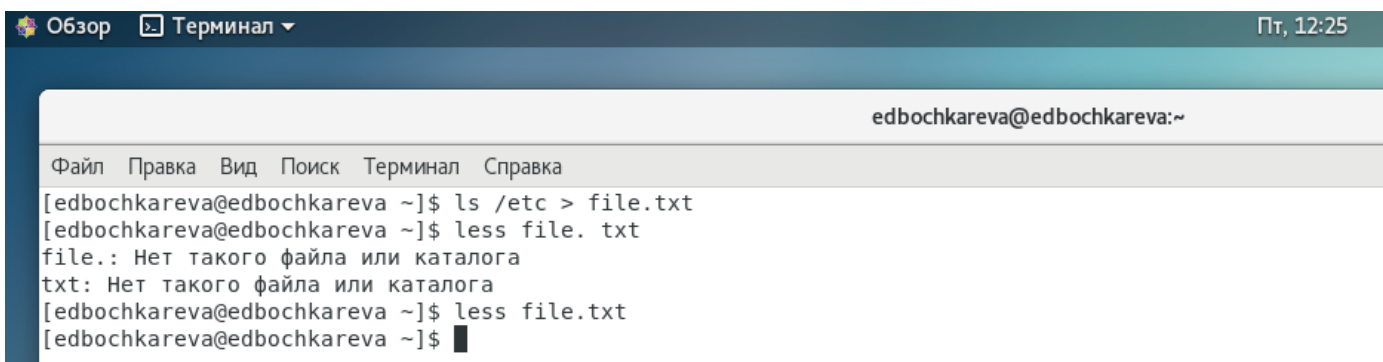
```
edbochkareva@edbochkareva:~  
Файл Правка Вид Поиск Терминал Справка  
[edbochkareva@edbochkareva ~]$ ls /etc > file.txt  
[edbochkareva@edbochkareva ~]$ less file.txt  
file.: Нет такого файла или каталога  
txt: Нет такого файла или каталога  
[edbochkareva@edbochkareva ~]$ less file.txt  
[edbochkareva@edbochkareva ~]$ ls ~ >> file.txt  
[edbochkareva@edbochkareva ~]$ less file.txt  
[edbochkareva@edbochkareva ~]$
```

Рис.5: Рисунок 5

Осуществляем ввод команды для сохранения в файл содержимого домашнего каталога пользователя.

Поскольку для **перенаправления вывода** используется **>>**, новые данные будут добавлены в конец файла.

Для просмотра содержимого файла file.txt выполним команду **less file.txt** (рис.6).



```
edbochkareva@edbochkareva:~  
Файл Правка Вид Поиск Терминал Справка  
[edbochkareva@edbochkareva ~]$ ls /etc > file.txt  
[edbochkareva@edbochkareva ~]$ less file.txt  
file.: Нет такого файла или каталога  
txt: Нет такого файла или каталога  
[edbochkareva@edbochkareva ~]$ less file.txt  
[edbochkareva@edbochkareva ~]$ ls ~ >> file.txt  
[edbochkareva@edbochkareva ~]$ less file.txt  
[edbochkareva@edbochkareva ~]$
```

Рис.6: Рисунок 6

Осуществляем проверку содержимого файла **file.txt**.  
Просматриваем содержимое файла (рис.7)

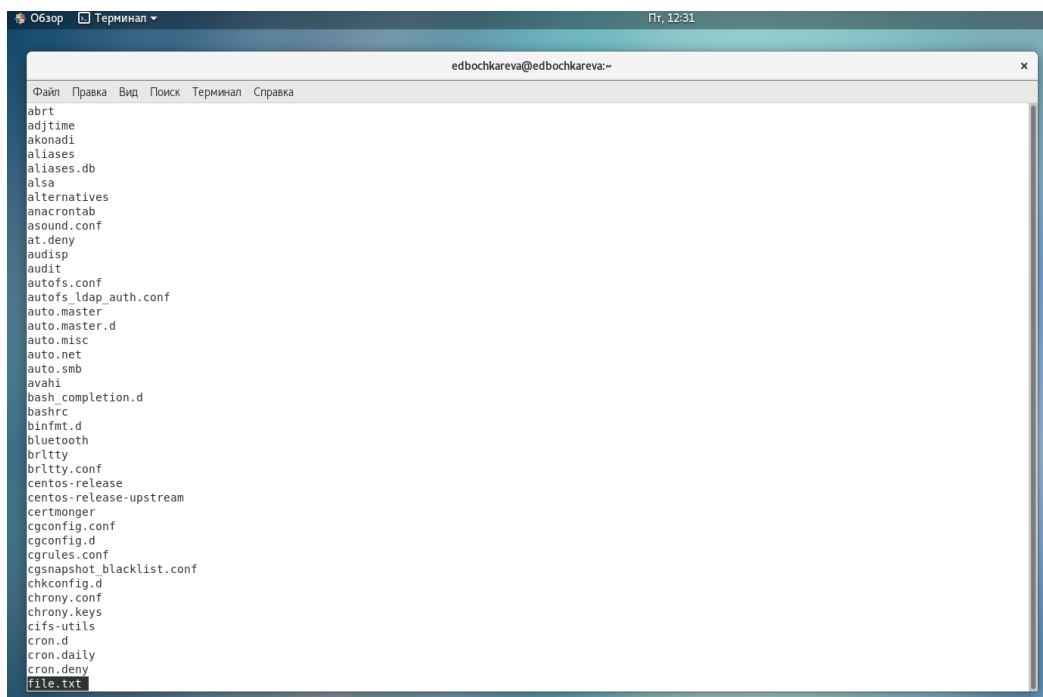


Рис.7: Рисунок 7

Содержимое файла **file.txt**.

Как видно, в конец файла были добавлены имена файлов из текущего каталога пользователя.

Осуществляем проверку выполнения новых команд в конце списка (рис.8).

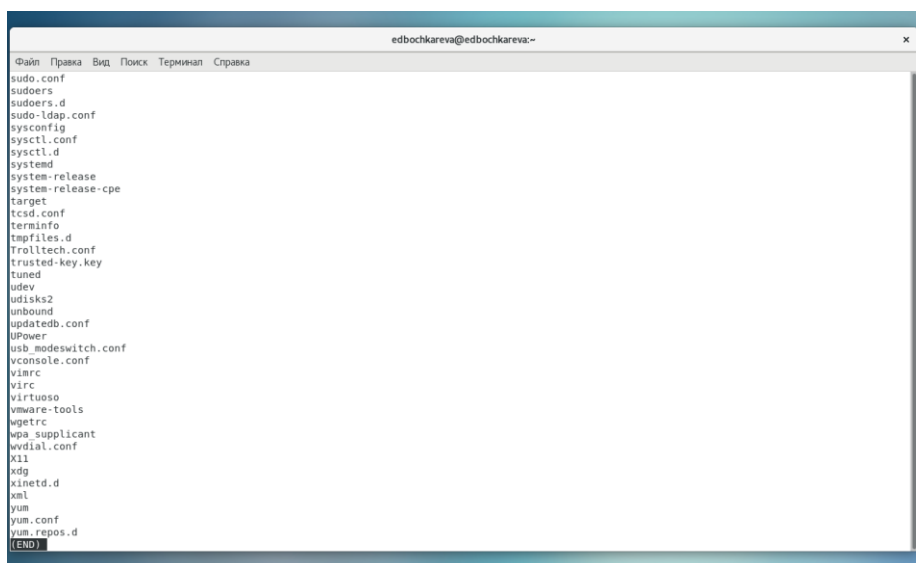


Рис.8: Рисунок 8



**Пункт 3. Выведите имена всех файлов из file.txt, имеющих расширение .conf, после чего запишите их в новый текстовый файл conf.txt.**

В каталоге txt ищем имена файлов, заканчивающихся на .conf (рис.9).

```
[edbochkareva@edbochkareva ~]$ cat file.txt | grep -E '\.conf$' > conf.txt
```

Рис.9: Рисунок 9

**Пункт 4. Определите, какие файлы в вашем домашнем каталоге имеют имена, начинавшиеся с символа с? Предложите несколько вариантов, как это сделать.**

Посмотрим имена файлов с помощью команды ls (рис.10).

```
[edbochkareva@edbochkareva ~]$ ls ~/c*  
/home/edbochkareva/conf.txt
```

Рис.10: Рисунок 10

Сделаем по аналогии выполненные выше действия при помощи команды find (рис.11).

```
[edbochkareva@edbochkareva ~]$ find ~ -maxdepth 1 -name 'c*'   
/home/edbochkareva/conf$.txt
```

Рис.11: Рисунок 11

**Пункт 5. Выведите на экран (постранично) имена файлов из каталога /etc, начинающиеся с символа h.**

Постранично просматриваем в каталоге etc имена файлов, начинающихся на h (рис.12).

```
[edbochkareva@edbochkareva ~]$ ls /etc/h* | less
```

Рис.12: Рисунок 12

В данном случае только одна страница таких файлов (рис.13).

```
/etc/host.conf
/etc/hostname
/etc/hosts
/etc/hosts.allow
/etc/hosts.deny

/etc/hp:
hplip.conf
(END)
```

Рис.13: Рисунок 13

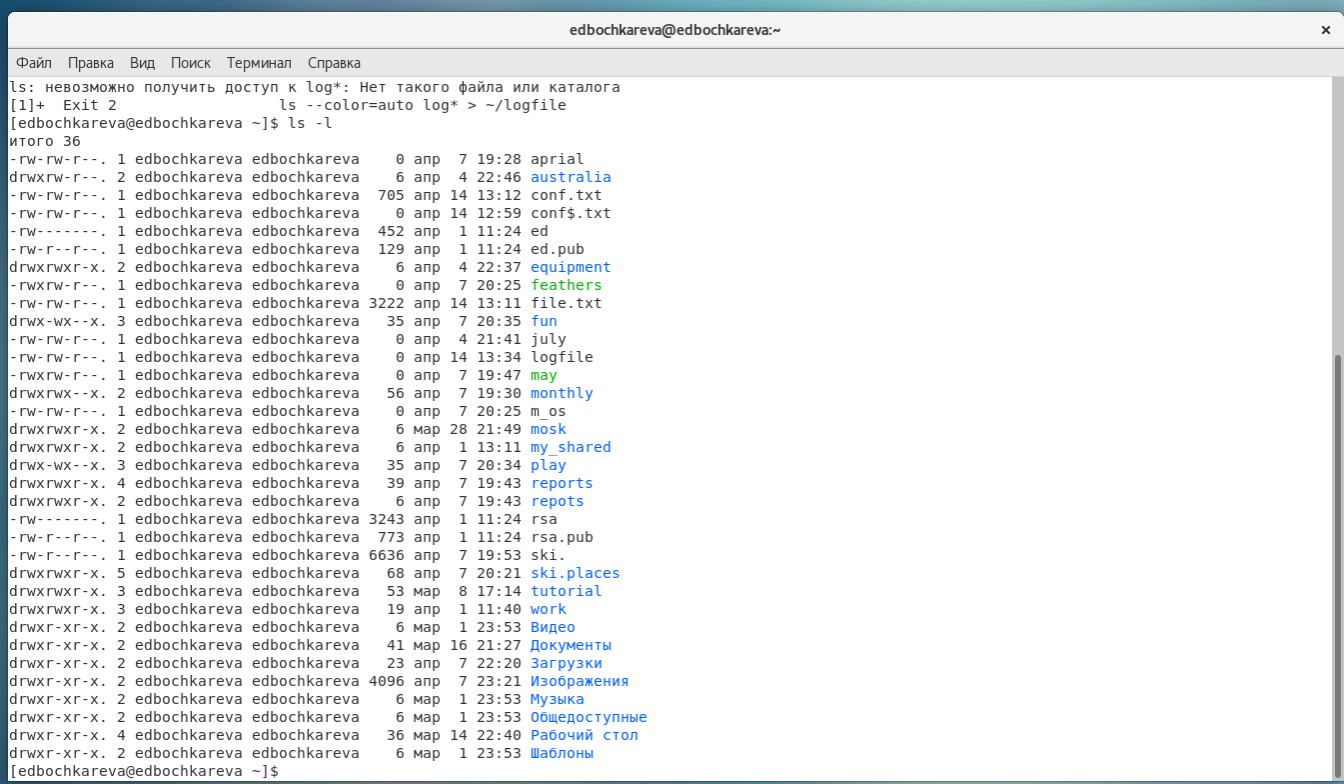
**Пункт 6. Запустите в фоновом режиме процесс, который будет записывать в файл ~/logfile файлы, имена которых начинаются с log.**

Запустим фоновый процесс, записывающий в файл logfile имена файлов, начинающиеся на log. При этом при первой попытке не будет записано ничего, так как таких файлов нет (рис.14).

```
[edbochkareva@edbochkareva ~]$ ls /etc/h | less
[edbochkareva@edbochkareva ~]$ ls /etc/h* | less
[edbochkareva@edbochkareva ~]$ ls log* > ~/logfile &
[1] 4829
ls: невозможно получить доступ к log*: Нет такого файла или каталога
[1]+  Exit 2                  ls --color=auto log* > ~/logfile
```

Рис.14: Рисунок 14

Введем команду `ls -l`. Команда `ls` выдаст список всех файлов из этих каталогов, кроме скрытых файлов (рис.15).



```
edbochkareva@edbochkareva:~  
Файл  Правка  Вид  Поиск  Терминал  Справка  
ls: невозможно получить доступ к log*: Нет такого файла или каталога  
[1]+  Exit 2                  ls --color=auto log* > ~/logfile  
edbochkareva@edbochkareva ~]$ ls -l  
итого 36  
-rw-rw-r--. 1 edbochkareva edbochkareva  0 апр  7 19:28 aprial  
drwxrwxr-x. 2 edbochkareva edbochkareva   6 апр  4 22:46 australia  
-rw-rw-r--. 1 edbochkareva edbochkareva 705 апр 14 13:12 conf.txt  
-rw-rw-r--. 1 edbochkareva edbochkareva   0 апр 14 12:59 conf$.txt  
-rw-rw-r--. 1 edbochkareva edbochkareva 452 апр  1 11:24 ed  
-rw-rw-r--. 1 edbochkareva edbochkareva 129 апр  1 11:24 ed.pub  
drwxrwxr-x. 2 edbochkareva edbochkareva   6 апр  4 22:37 equipment  
-rw-rw-r--. 1 edbochkareva edbochkareva   0 апр  7 20:25 feathers  
-rw-rw-r--. 1 edbochkareva edbochkareva 3222 апр 14 13:11 file.txt  
drwxrwxr-x. 3 edbochkareva edbochkareva  35 апр  7 20:35 fun  
-rw-rw-r--. 1 edbochkareva edbochkareva   0 апр  4 21:41 july  
-rw-rw-r--. 1 edbochkareva edbochkareva   0 апр 14 13:34 logfile  
-rw-rw-r--. 1 edbochkareva edbochkareva   0 апр  7 19:47 may  
drwxrwxr-x. 2 edbochkareva edbochkareva  56 апр  7 19:30 monthly  
-rw-rw-r--. 1 edbochkareva edbochkareva   0 апр  7 20:25 m_os  
drwxrwxr-x. 2 edbochkareva edbochkareva   6 мар 28 21:49 mosk  
drwxrwxr-x. 2 edbochkareva edbochkareva   6 апр  1 13:11 my_shared  
drwxrwxr-x. 3 edbochkareva edbochkareva  35 апр  7 20:34 play  
drwxrwxr-x. 4 edbochkareva edbochkareva  39 апр  7 19:43 reports  
drwxrwxr-x. 2 edbochkareva edbochkareva   6 апр  7 19:43 repots  
-rw-rw-r--. 1 edbochkareva edbochkareva 3243 апр  1 11:24 rsa  
-rw-rw-r--. 1 edbochkareva edbochkareva  773 апр  1 11:24 rsa.pub  
-rw-rw-r--. 1 edbochkareva edbochkareva 6636 апр  7 19:53 ski.  
drwxrwxr-x. 5 edbochkareva edbochkareva   68 апр  7 20:21 ski.places  
drwxrwxr-x. 3 edbochkareva edbochkareva   53 мар  8 17:14 tutorial  
drwxrwxr-x. 3 edbochkareva edbochkareva   19 апр  1 11:40 work  
drwxr-xr-x. 2 edbochkareva edbochkareva   6 мар  1 23:53 Видео  
drwxr-xr-x. 2 edbochkareva edbochkareva  41 мар 16 21:27 Документы  
drwxr-xr-x. 2 edbochkareva edbochkareva  23 апр  7 22:20 Загрузки  
drwxr-xr-x. 2 edbochkareva edbochkareva 4096 апр  7 23:21 Изображения  
drwxr-xr-x. 2 edbochkareva edbochkareva   6 мар  1 23:53 Музыка  
drwxr-xr-x. 2 edbochkareva edbochkareva   6 мар  1 23:53 Общедоступные  
drwxr-xr-x. 4 edbochkareva edbochkareva   36 мар 14 22:40 Рабочий стол  
drwxr-xr-x. 2 edbochkareva edbochkareva   6 мар  1 23:53 Шаблоны  
edbochkareva@edbochkareva ~]$
```

Рис. 15: Рисунок 15

При второй попытке ввода `ls` в данный файл попадет единственное имя: имя этого же файла (рис.16).

```
[edbochkareva@edbochkareva ~]$ ls log* > ~/logfile &  
[1] 5066  
[1]+ Done                  ls --color=auto log* > ~/logfile  
[edbochkareva@edbochkareva ~]$ cat logfile  
logfile  
[edbochkareva@edbochkareva ~]$ █
```

Рис.16: Рисунок 16

## Пункт 7. Удалите файл `~/logfile`.

Удаляем файл `logfile` (рис.17).

```
[edbochkareva@edbochkareva ~]$ rm logfile  
[edbochkareva@edbochkareva ~]$ █
```

Рис.17: Рисунок 17

## Пункт 8. Запустите из консоли в фоновом режиме редактор gedit.

Запускаем из консоли в фоновом режиме редактор gedit (рис 18.).

```
[edbochkareva@edbochkareva ~]$ gedit  
gedit &
```

Рис.18: Рисунок 18

Gedit - свободный текстовый редактор для среды GNOME

Открывается «Безымянный документ»: окно программы gedit, запустим командой gedit &. Знак амперсанд означает, что программа будет запущена в фоновом режиме, но это относится к тому, что происходит в самой консоли.

Просматриваем окно «Безымянный документ» (рис.19).

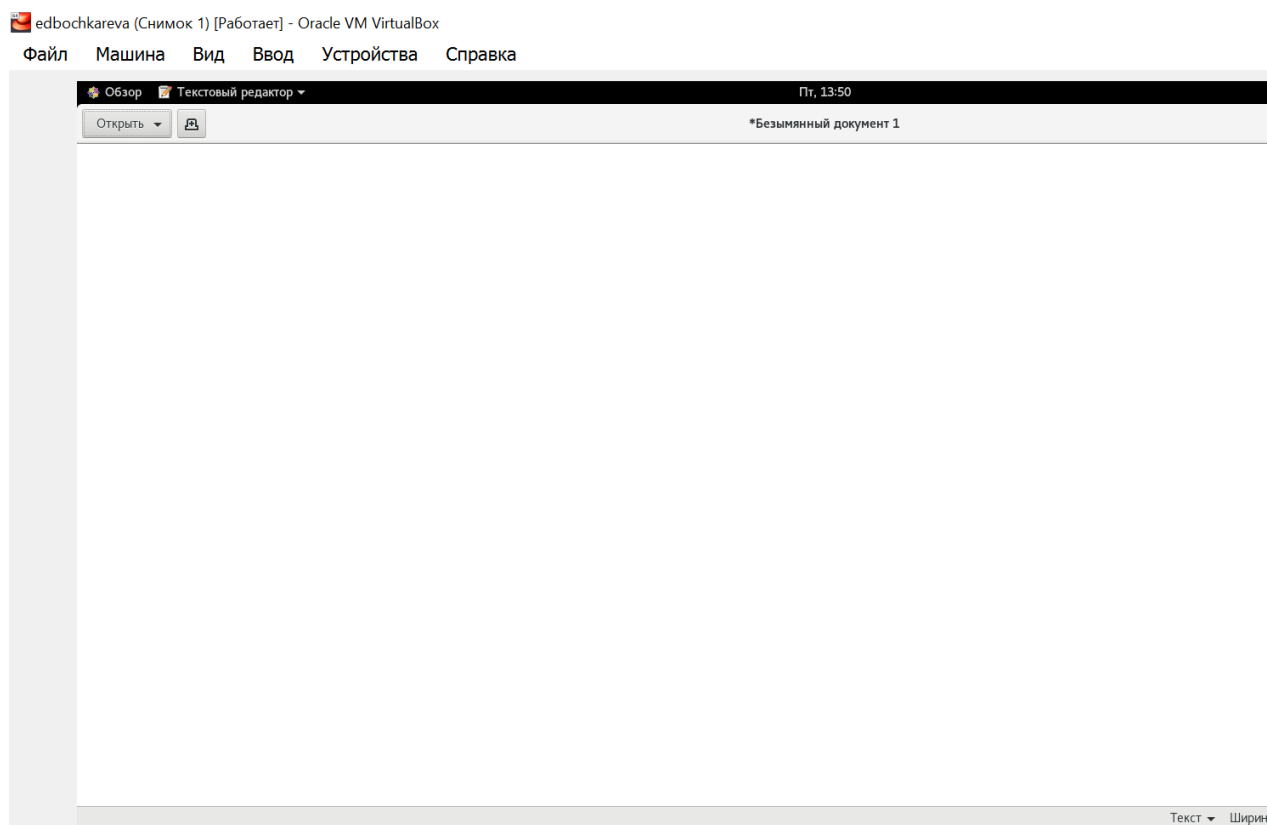


Рис.19: Рисунок 19

**Пункт 9. Определите идентификатор процесса gedit, используя команду ps, конвейер и фильтр grep. Можно ли определить этот идентификатор более простым способом?**

Определяем идентификатор процесса программы GEDIT при помощи команды ps (рис.20).

```
[edbochkareva@edbochkareva ~]$ ps -AF | grep gedit
edbochk+  6574  2916  0 28208  980  0 14:01 pts/0    00:00:00 grep --color=auto gedit
[edbochkareva@edbochkareva ~]$
```

Рис.20: Рисунок 20

**Команда ps** выводит сведения о процессах в статическом виде.

**Grep** — это команда для поиска внутри текстовых файлов.

Определяем идентификатор процесса программы gedit при помощи команды **pidof** (рис.21).

```
[edbochkareva@edbochkareva ~]$ pidof gedit
```

Рис. 21: Рисунок 21

**Команда pidof** используется для определения идентификаторов PID конкретной запущенной программы.

**Пункт 10. Прочтите справку (man) команды kill, после чего используйте её для завершения процесса gedit.**

## Просматриваем справку по команде `man kill` (рис.22).

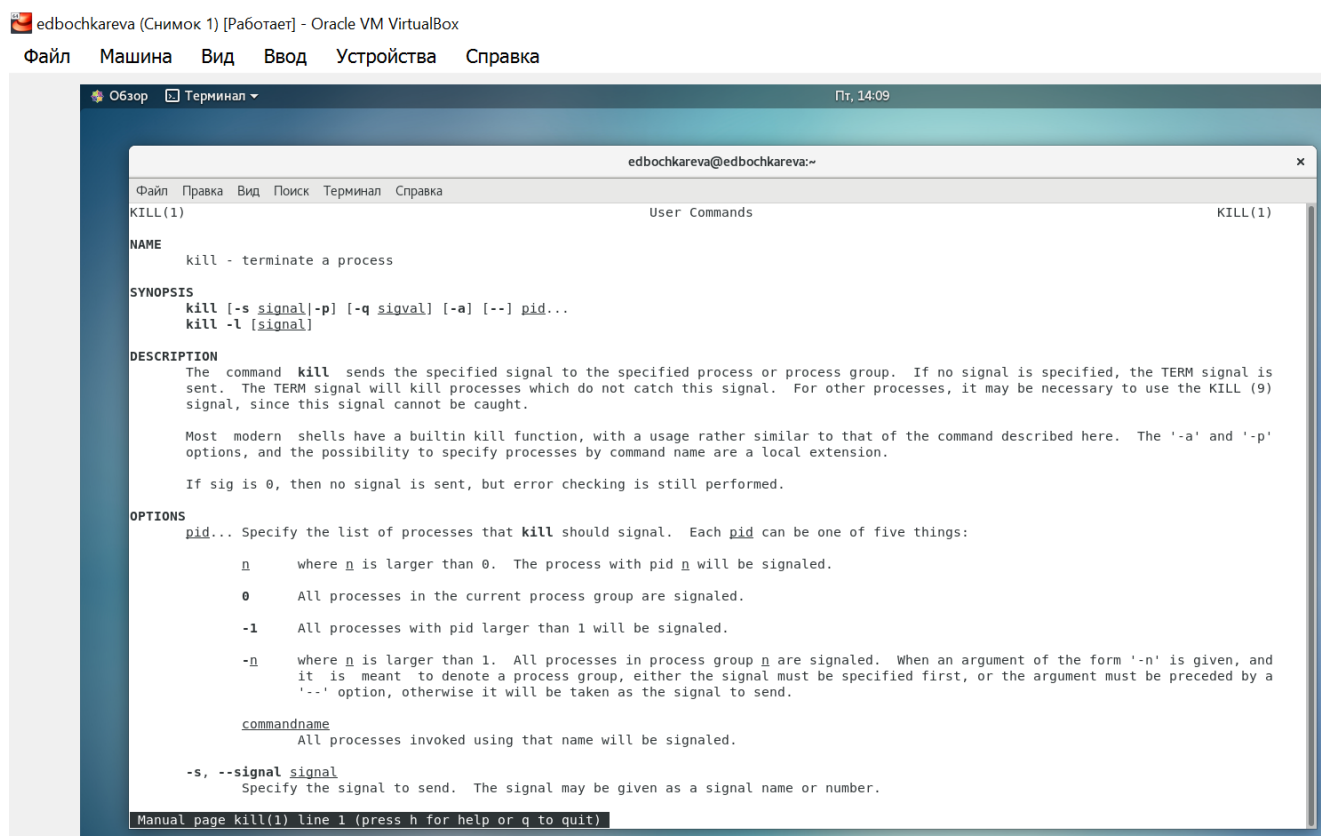


Рис.22: Рисунок 22

**Команда `kill`** является встроенной командой командной оболочки, предназначенной для отправки системных сигналов определенным процессам. Команда принимает числовые идентификаторы процессов, а также числовые или текстовые идентификаторы сигналов.

## Закрываем `gedit` при помощи команды `kill` (рис.23).

```
[edbochkareva@edbochkareva ~]$ kill 2916
```

Рис.23: Рисунок 23

**Пункт 11. Выполните команды `df` и `du`, предварительно получив более подробную информацию об этих командах, с помощью команды `man`.**

Выполним команду `df`, предварительно получив более подробную информацию об этой команде, с помощью команды `man` (рис.24).

```
[edbochkareva@edbochkareva ~]$ man df
[edbochkareva@edbochkareva ~]$
```

Рис.24: Рисунок 24

**df** (аббревиатура от **disk free**) — утилита в UNIX и UNIX-подобных системах, показывает список всех файловых систем по именам устройств, сообщает их размер, занятое и свободное пространство и точки монтирования:

Получаем более подробную справочную информацию о команде **df** (рис.25).

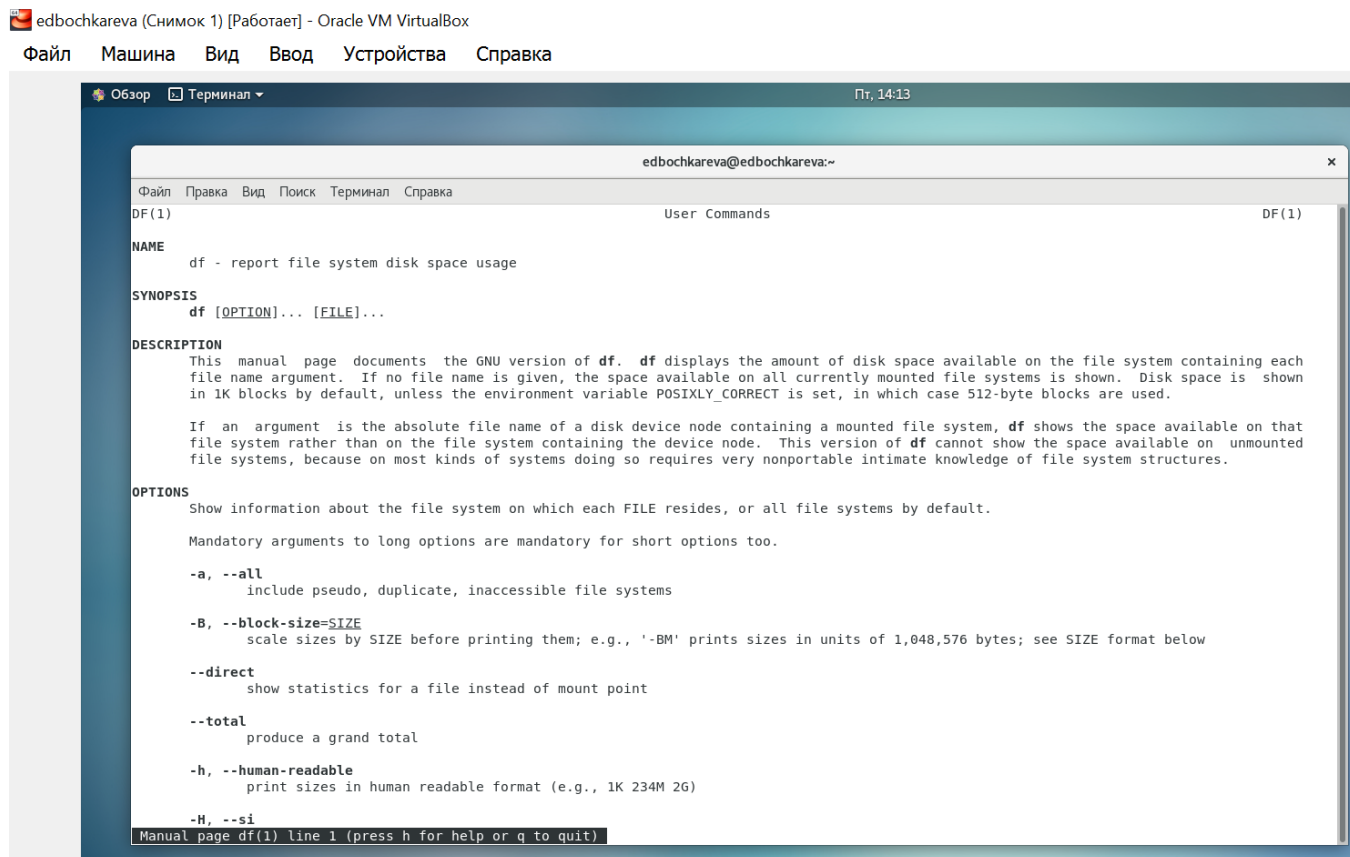


Рис.25: Рисунок 25

Запускаем команду **df** для того, чтобы посмотреть «свободное место» (рис.26).

Файловая система	1K-блоков	Использовано	Доступно	Использовано%	Смонтировано в
devtmpfs	490380	0	490380	0%	/dev
tmpfs	507360	0	507360	0%	/dev/shm
tmpfs	507360	7812	499548	2%	/run
tmpfs	507360	0	507360	0%	/sys/fs/cgroup
/dev/mapper/centos_edbochkareva-root	38770180	8113256	30656924	21%	/
/dev/loop1	56960	56960	0	100%	/var/lib/napd/snap/core18/2721
/dev/loop2	37888	37888	0	100%	/var/lib/napd/snap/gh/502
/dev/loop0	51072	51072	0	100%	/var/lib/napd/snap/snapd/18596
/dev/sda1	1038336	177244	861092	18%	/boot
shared	123741180	123441188	299992	100%	/media/sf_shared
tmpfs	101472	32	101440	1%	/run/user/1000
/dev/sr0	51806	51806	0	100%	/run/media/edbochkareva/VBox_GAs_7.0.6

```

[edbochkareva@edbochkareva ~]$
  
```

Рис.26: Рисунок 26

Выполним команду **man du**, предварительно получив более подробную информацию об этой команде, с помощью команды **man**:

```

[edbochkareva@edbochkareva ~]$ man du
  
```

**Команда du** позволяет задействовать одноименную утилиту, предназначенную для вывода информации об объеме дискового пространства, занятого файлами и директориями.

**man** (от англ. manual — руководство) — команда Unix, предназначенная для форматирования и вывода справочных страниц.

Просматриваем подробную информацию о команде **du** (рис.27)

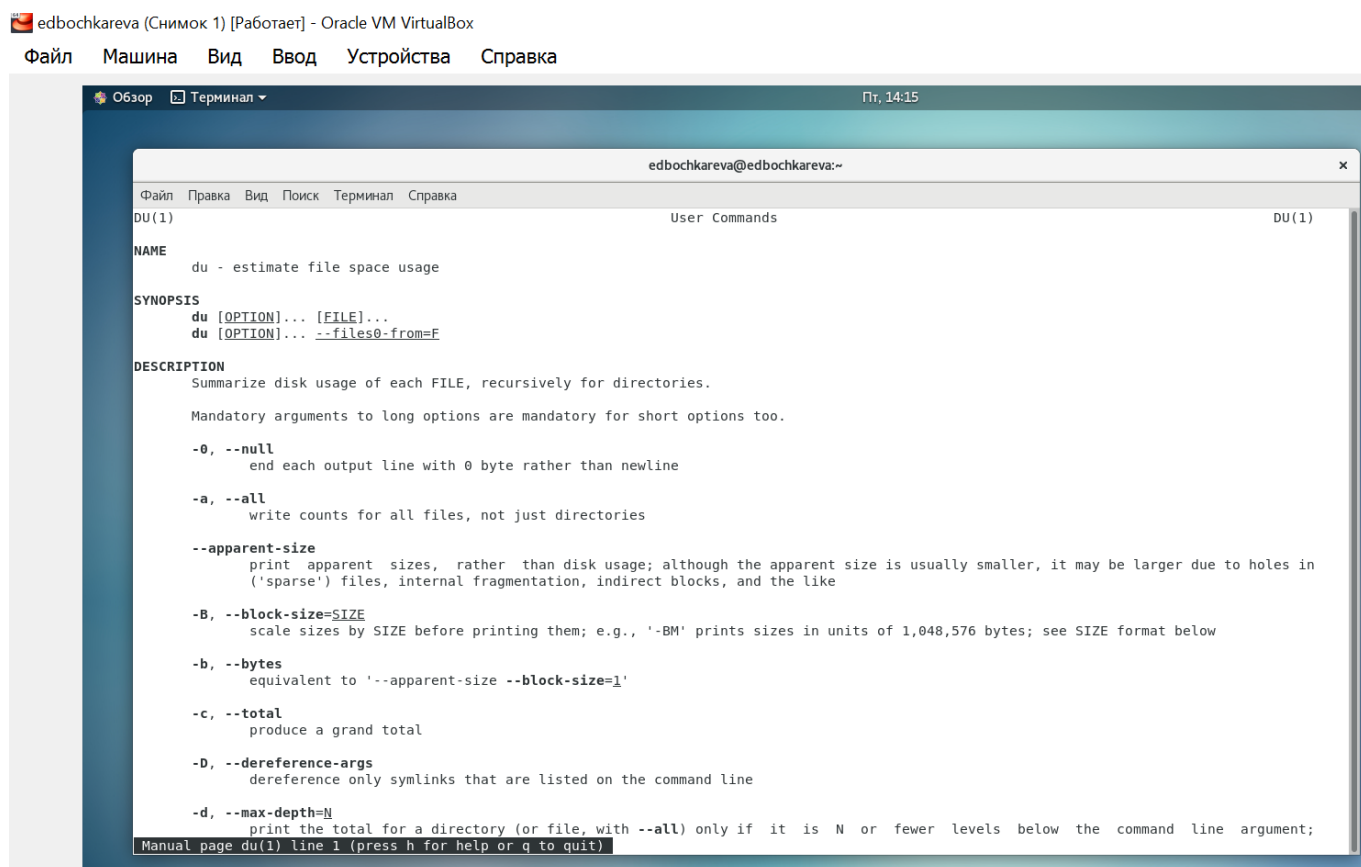


Рис.26: Рисунок 27

При помощи команды **du** определим «используемое место» (рис.28)

```
[edbochkareva@edbochkareva ~]$ du
```

Рис.28: Рисунок 28



## Просмотр содержимого после выполнения команды **du** (рис.29).

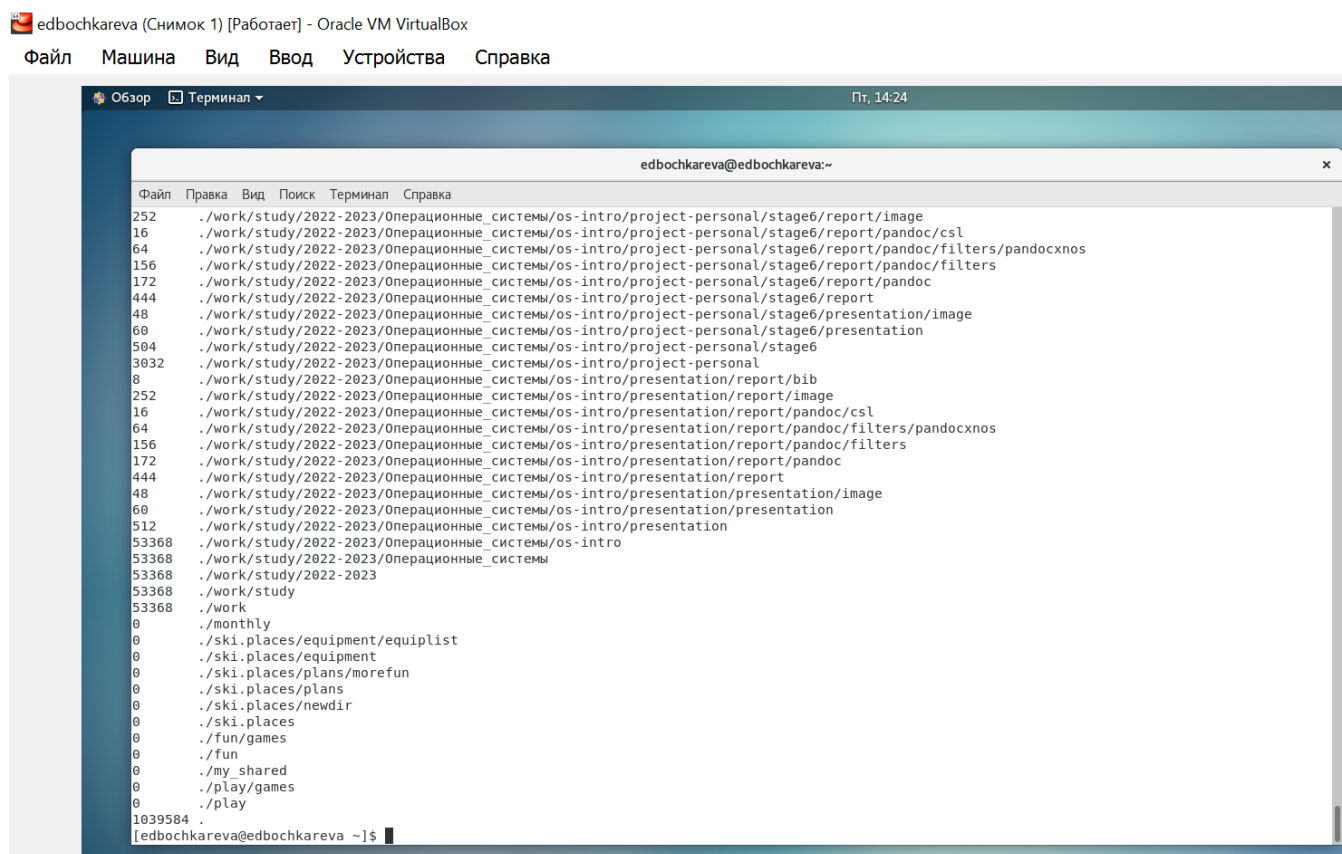


Рис.29: Рисунок 29

## Определим «используемое место» при помощи команды **du | less** (рис.30).

```
[edbochkareva@edbochkareva ~]$ du | less
[edbochkareva@edbochkareva ~]$
```

Рис.30: Рисунок 30

**Команда less** позволяет перематывать текст не только вперёд, но и назад, осуществлять поиск в обоих направлениях, переходить сразу в конец или в начало файла. Особенность less заключается в том, что команда не считывает текст полностью, а загружает его небольшими фрагментами.

Просмотр содержимого после выполнения команды `du | less` (рис. 31).

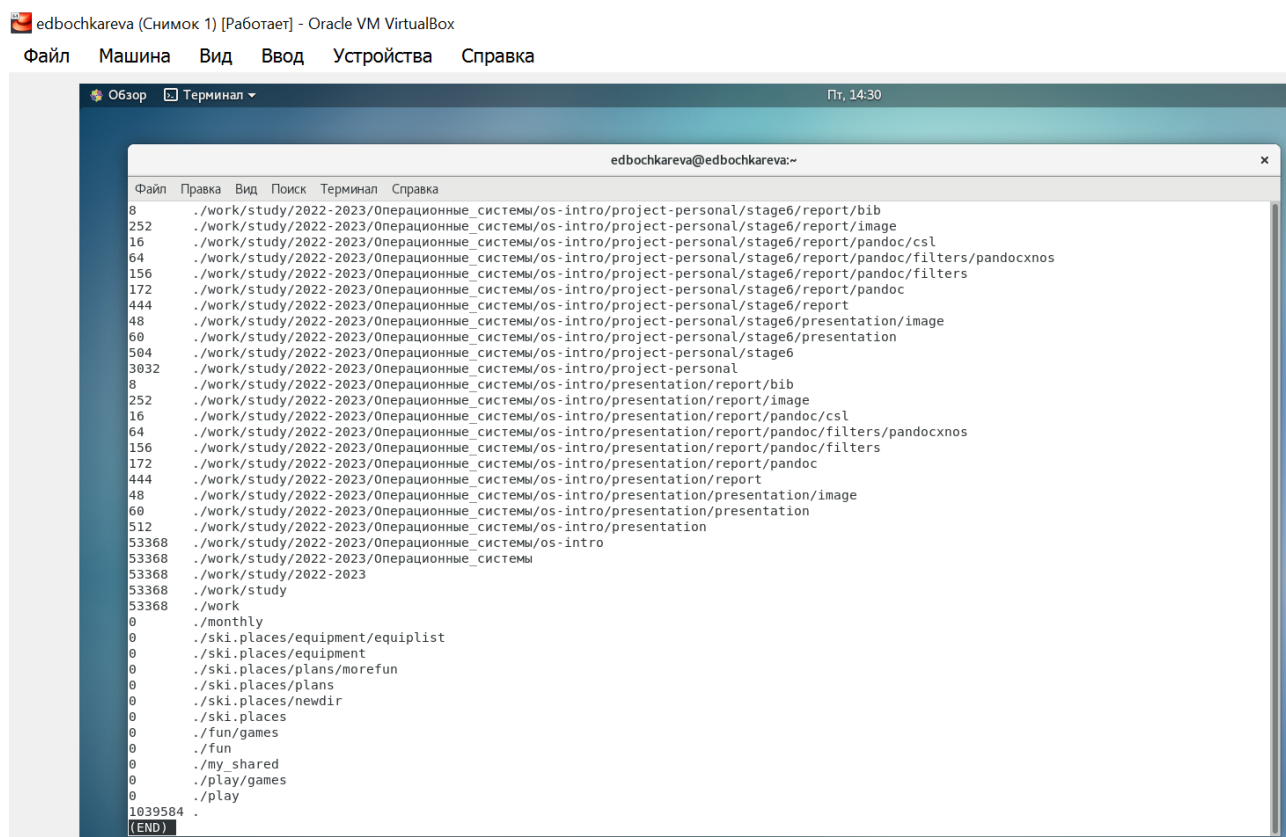


Рис.31: Рисунок 31

Определим используемое место при помощи команды `du -hs` (рис.32).

```
[edbochkareva@edbochkareva ~]$ du -hs .  
1016M  .  
[edbochkareva@edbochkareva ~]$ █
```

Рис.32: Рисунок 32

**Пункт 12. Воспользовавшись справкой команды `find`, выведите имена всех директорий, имеющихя в вашем домашнем каталоге.**

При помощи команды `find` найдем все каталоги в текущем каталоге (рис.33).

```
[edbochkareva@edbochkareva ~]$ find ~ -type d
```

Рис.33: Рисунок 33

Просматриваем выполнение команды **find**, найдем все каталоги в текущем каталоге (рис.34)

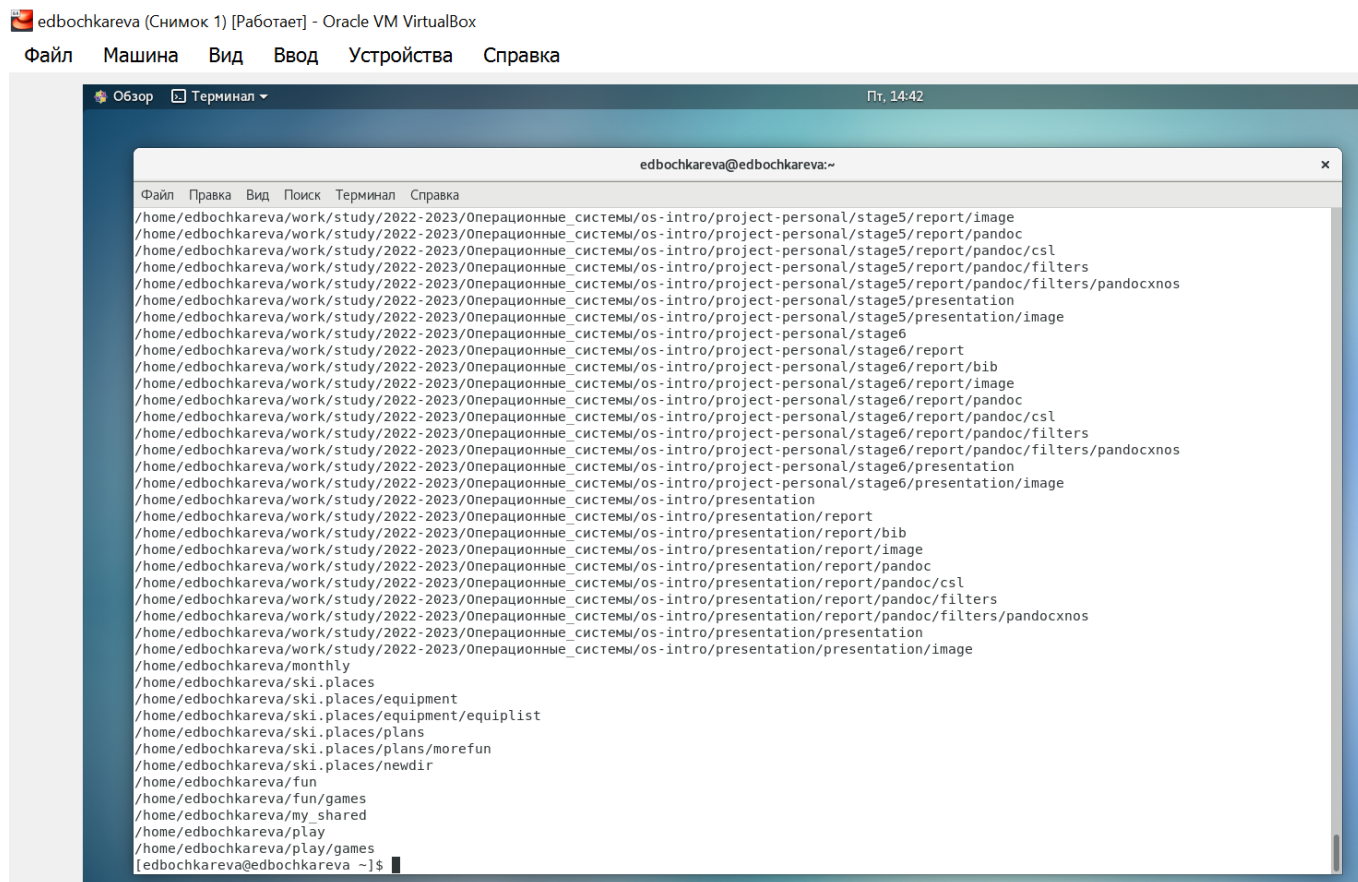


Рис.34: Рисунок 34

При помощи команды **find ~ -type d | less** найдем все каталоги в текущем каталоге (рис.35).

```
[edbochkareva@edbochkareva ~]$ find ~ -type d | less
[edbochkareva@edbochkareva ~]$
```

Рис.35: Рисунок 35

Просматриваем выполнение команды `find ~ -type d | less` до конца списка (рис.36).

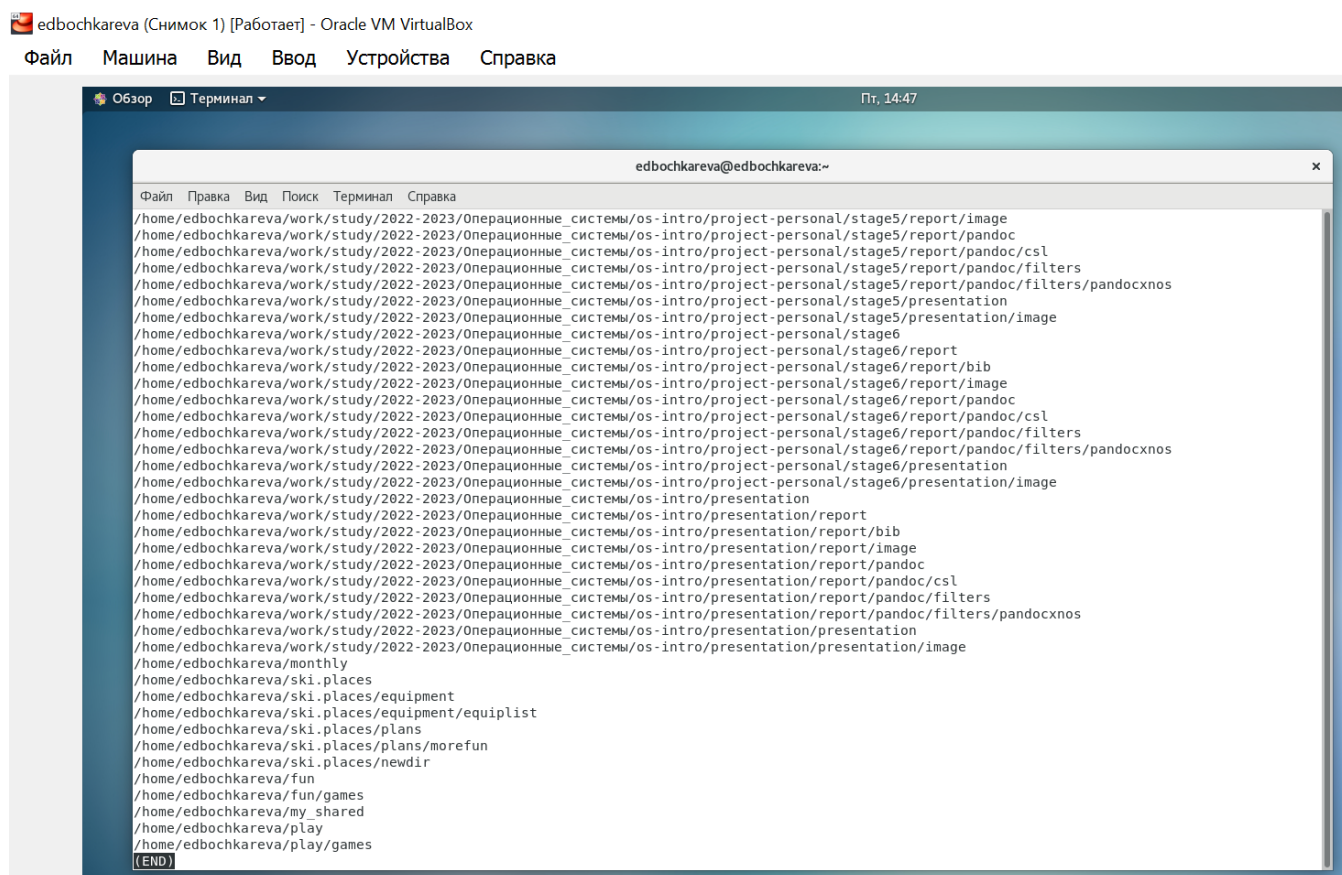


Рис.36: Рисунок 36

## Ответы на контрольные вопросы:

### 1. Какие потоки ввода вывода вы знаете?

**Ответ:** Существуют потоки стандартного ввода, вывода и вывода ошибок.

Стандартный ввод при работе пользователя в терминале передается через клавиатуру.

Стандартный вывод и стандартная ошибка отображаются на дисплее терминала пользователя в виде текста.

Ввод и вывод распределяется между тремя стандартными потоками:

- `stdin` — стандартный ввод (клавиатура),
- `stdout` — стандартный вывод (экран),
- `stderr` — стандартная ошибка (вывод ошибок на экран).

Потоки также пронумерованы:

- `stdin` — 0,
- `stdout` — 1,
- `stderr` — 2.

Из стандартного ввода команда может только считывать данные, а два других потока могут использоваться только для записи.

Данные выводятся на экран и считываются с клавиатуры, так как стандартные потоки по умолчанию ассоциированы с терминалом пользователя.

### 2. Объясните разницу между операцией `>` и `>>`.

**Ответ:** Операция `>` означает, что вывод команды будет записан в файл, прежнее содержимое которого будет уничтожено (если ранее существовало). При внесении изменений в файл, когда вы хотите перезаписать существующие данные, использую оператор `<>`.

Операция `>>` дописывает вывод команды в конец файла (или создаёт новый, если такого файла нет). Если я хочу что-то добавить в этот файл, использую оператор `<>>`.

Оба оператора являются операторами направления вывода. Основное отличие указано ниже:

- > : Перезаписывает существующий файл или создает файл, если файл с указанным именем отсутствует в каталоге.
- >> : добавляет существующий файл или создает файл, если файл с указанным именем отсутствует в каталоге.

### **Примеры:**

```
$ echo «Добро пожаловать в ElenaEx» > my_file_1.txt
```

```
$ echo «Добро пожаловать в ElenaEx» >> my_file_2.txt
```

### 3. Что такое «Конвейер?»

**Ответ:** **Конвейер** (англ. *pipeline*) в терминологии операционных систем семейства Unix — некоторое множество процессов, для которых выполнено следующее перенаправление ввода-вывода: то, что выводит на поток стандартного вывода предыдущий процесс, попадает в поток стандартного ввода следующего процесса. **Конвейер** - это передача с вывода одной команды на стандартный ввод следующей, соединение стандартного вывода одной команды со стандартным вводом другой.

Осуществляется при помощи символа | между соответствующими командами, **пример: ls | less**

Можем сделать это, указав команды в нужном порядке и разделив их вертикальной чертой '|' (иногда называемой ``трубой (pipe)``).

**Запуск конвейера реализован с помощью системного вызова `pipe()`.**

Код возврата конвейера равен коду возврата последней команды.

В bash можно изменить это поведение, включив опцию `pipefail`:

```
set -o pipefail
```

## 2. Что такое процесс? Чем это понятие отличается от программы?

Процесс - это запущенная программа. При выполнении одной программы может быть запущено несколько принадлежащих к ней процессов.

Термин "процесс" впервые появился при разработке операционной системы Multix и имеет несколько определений, которые используются в зависимости от контекста.

Процесс - это:

1. программа на стадии выполнения
2. "объект", которому выделено процессорное время
3. асинхронная работа

Для описания состояний процессов используется несколько моделей.

Самая простая модель - это модель трех состояний. Модель состоит из:

1. состояния выполнения
2. состояния ожидания
3. состояния готовности

## 5. Что такое PID и GID?

**PID** - это идентификатор процесса.

**GID** - идентификатор группы.

**PID** - идентификатор процесса **PPID** - идентификатор процесса, породившего данный **UID** и **GID** - идентификаторы прав процесса (соответствует **UID** и **GID** пользователя, от которого запущен процесс).

**PID** обозначает идентификационный номер процесса, который обычно используется большинством ядер операционной системы, таких как Linux, Unix, macOS и Windows. Это уникальный идентификационный номер, который автоматически присваивается каждому процессу, когда он создается в операционной системе.

### Идентификатор группы (GID)

Кроме идентификационного номера пользователя с учётной записью связан идентификатор группы. Группы пользователей применяются для организации доступа нескольких пользователей к некоторым ресурсам. У группы, так же, как и у пользователя, есть имя и идентификационный номер — **GID (Group ID)**.

## 6. Что такое задачи и какая команда позволяет ими управлять?

**Задача** - это программа, запущенная в фоне.

Для управления задачами используется программа **jobs**:  
**jobs** — просмотр списка собственных задач (процессов).

Запущенные из консоли с помощью амперсанда команды, работают в фоновом режиме и называются *задачами* (**jobs**). Можно сказать, что *задачи* это процессы, привязанные к командному интерпретатору. Такие задачи помимо традиционного **PID** имеют еще свою нумерацию начинающуюся с единицы.

Просмотреть запущенные *задачи* интерпретатора, можно



командой **jobs**.

**В примере** ниже показана ситуация когда есть две *задачи* и выполнение одного из них остановлено:

```
1 $ jobs
2 [1]+  Stopped                  top
3 [2]-  Running                  sleep 100 &
```

## 7.Найдите информацию об утилитах **top** и **htop** . Каковы их функции?

Утилиты **top** и **htop** используются для просмотра списка запущенных процессов.

Функции утилит **top** и **htop**: в частности они позволяют сортировать процессы по потреблению ресурсов процессора и памяти, что позволяет, например, находить процессы, которые потребляют значительную часть ресурсов системы.

**Команда top.** Она немного проще чем та же утилита **htop**, но в отличие от **ps** позволяет выводить информацию о системе, а также список процессов динамически обновляя информацию о потребляемых ими ресурсах. Утилита не всегда установлена по умолчанию, для её установки в Ubuntu используйте команду:

**sudo apt install top**

Затем для запуска просто выполните в терминале:

**top**

**НТОР - монитор процессов.** **htop** — продвинутый монитор процессов, написанный для Linux. Он был задуман заменить стандартную программу **top**. **Нтор** показывает динамический список системных процессов, список обычно выравнивается по использованию ЦПУ. В отличие от **top**, **htop** показывает все процессы в системе. Также показывает время непрерывной работы, использование процессоров и памяти. **Нтор** часто применяется в тех случаях, когда информации даваемой утилитой **top** недостаточно, например при поиске утечек памяти в процессах.

Установка:

```
sudo apt-get install htop
```

Запускаем:

```
sudo htop
```

## **8. Назовите и дайте характеристику команде поиска файлов. Приведите примеры использования этой команды.**

Для поиска файлов может использоваться программа **find**.

Пример использования программы, для поиска в текущем каталоге файлов, которые содержат в названии слово `user` в любом регистре:

```
find . -iname '*user*'
```

Рассмотрим несколько наиболее популярных способов поиска файлов в Linux, используя терминал.

**find:** для поиска файлов из командной строки вы можете использовать команду “find”.

У этой команды следующий синтаксис:

**“path”** - Секция для указания директории поиска. Если ничего не указано поиск идет по текущей директории.

**“criteria”** - Опции поиска.

**“action”** - Опции, которые влияют на состояние поиска или контролируют его, например,

**“-print”**

```
pavs@uberhaxor:/$ find /usr/share/doc/ -name "*.txt"
/usr/share/doc/alsa-base/driver/Bt87x.txt
/usr/share/doc/alsa-base/driver/ControlNames.txt
/usr/share/doc/alsa-base/driver/Joystick.txt
/usr/share/doc/alsa-base/driver/MIXART.txt
/usr/share/doc/alsa-base/driver/VIA82xx-mixer.txt
/usr/share/doc/alsa-base/driver/emul0k1-jack.txt
/usr/share/doc/alsa-base/driver/serial-ul6550.txt
/usr/share/doc/appport/package-hooks.txt
/usr/share/doc/bittorrent/credits.txt
/usr/share/doc/bitchx/documentation/mirc-colors.txt
/usr/share/doc/console-tools/contrib/keysyms.h.txt
/usr/share/doc/gcc-4.1-base/C++/libstdc++_symbols.txt
/usr/share/doc/gnome-keyring/file-format.txt
/usr/share/doc/hpijs/users-guide.txt
/usr/share/doc/hplip/users-guide.txt
/usr/share/doc/kde/HTML/en/kppp/ttyS-cua.txt
/usr/share/doc/libgettext-ruby1.8/examples/rails/public/robots.txt
/usr/share/doc/libruby1.8/enumerator/enumerator.txt
/usr/share/doc/libruby1.8/etc/etc.txt
/usr/share/doc/libruby1.8/syslog/syslog.txt
/usr/share/doc/qstat/README.txt
/usr/share/doc/qstat/UT2003.txt
/usr/share/doc/qstat/examples/README.txt
/usr/share/doc/nmap/leet-nmap-ascii-art.txt
/usr/share/doc/vlc/bugreport-howto.txt
/usr/share/doc/alien/gendiff.txt
/usr/share/doc/gawk/examples/network/stoxdata.txt
/usr/share/doc/rrdtool/txt/rrddump.txt
/usr/share/doc/rrdtool/txt/rrdrestore.txt
/usr/share/doc/rrdtool/txt/rrdfirst.txt
/usr/share/doc/rrdtool/txt/rrdinfo.txt
/usr/share/doc/rrdtool/txt/rrdlast.txt
/usr/share/doc/rrdtool/txt/rrdlastupdate.txt
/usr/share/doc/rrdtool/txt/rrdresize.txt
```

## 9. Можно ли по контексту (содержанию) найти файл? Если да, то как?

Для поиска по содержимому может использоваться команда **grep**, которой передаётся, первым параметром, строка для поиска, а далее имена файлов, в которых нужно искать.

Например, для поиска строки `test` в файлах в текущем каталоге можно использовать такую команду:

```
grep "test" ./*
```

Для поиска файла по содержимому проще всего воспользоваться **командой `grep`** (вместо `find`).

**Пример:**

```
grep -r строка_поиска каталог
```

или:

```
grep -lir 'class List' /home/balancer/programming/java/jbforth
```

## 10. Как определить объем свободной памяти на жёстком диске?

Для того, чтобы определить объём памяти на диске, можно **воспользоваться командой `df`**, в качестве параметра можно передать путь, тогда будет выведено свободное место на соответствующем разделе (путь не обязательно должен указывать на точку монтирования раздела).

**`df`** – это команда позволяет отобразить информацию о свободном/доступном месте на диске, файловой системы раздела. Чтобы листинг команды был более читабелен, нужно использовать ее с опциями. Например:

```
# df -h
```

```
[root@server ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        899M   0    899M   0% /dev
tmpfs           914M   0    914M   0% /dev/shm
tmpfs           914M  97M   817M  11% /run
tmpfs           914M   0    914M   0% /sys/fs/cgroup
/dev/vda2       39G   3.6G   34G   10% /
/dev/vda1       488M  176M  277M   39% /boot
tmpfs          183M   0    183M   0% /run/user/0
[root@server ~]#
```

Описание столбцов:

- `Filesystems` – имя файловой системы
- `Size` – размер раздела

- **Used** – используемое дисковое пространство
- **Avail** – доступное дисковое пространство(свободное)
- **Use%** — занятое дисковое пространство в процентах
- **Mounted on** – смонтировано. Указывает директорию, к которой примонтирован раздел.

**При использовании опции -h дисковое пространство выводится в Гб. Если размер меньше 1Гб, то вывод будет в Мб.**

**Дополнительные опции:**

**df -m** – информация будет отображена в Мб;

**df -k** — информация будет отображена в Кб;

**df -T** – к выводу добавиться тип файловой системы.

## **11. Как определить объем вашего домашнего каталога?**

В операционных системах на базе Linux посмотреть размер папки (директории) можно с помощью команды **du**.

Эта команда, выполняемая в консоли, позволяет оценить используемый объем места на жестком диске отдельно по папкам и файлам, просуммировать результат, узнать общий размер папки.

Чтобы определить объём домашнего каталога, можно выполнить следующую команду:

**du -hs ~**

**Общий синтаксис команды du следующий:**

**du**

**du имяПапки**

**du [ключи] имяПапки**

**Пример.** Без передачи каких-либо параметров команда **du** выводит название и размер каждой папки по текущему пути, при этом включая в вывод все подкаталоги:

```
du
```

Пример вывода:

```
8      ./share/gegl-0.2/plugin-ins
12     ./share/gegl-0.2
16     ./share/rhythmbox
56     ./share/icons/hicolor/16x16/apps
60     ./share/icons/hicolor/16x16
208    ./share/icons/hicolor/256x256/apps
212    ./share/icons/hicolor/256x256
64     ./share/icons/hicolor/48x48/apps
68     ./share/icons/hicolor/48x48
56     ./share/icons/hicolor/32x32/apps
60     ./share/icons/hicolor/32x32
```

**Ключ -h** обеспечит вывод в удобном для восприятия человеком виде, а, благодаря ключу **-s**, будет выведен суммарный объём всего каталога.

## 12. Как удалить зависший процесс?

Для того, чтобы удалить зависший процесс, необходимо, зная его идентификатор, выполнить команду **kill** с ключом **-9** и идентификатором процесса, переданном в качестве следующего.

Для завершения *процесса* нужно вызвать утилиту **kill** с параметром **"-9"**.

Когда известен PID процесса, мы можем убить его **командой kill**.

Команда **kill** принимает в качестве параметра PID процесса.

Например, убьем процесс с номером 25609:

```
kill 25609
```

## **Выводы, согласованные с целью работы:**

В процессе выполнения лабораторной работы были использованы приложения, предназначенные для поиска файлов (`find`, `ls`), фильтрации данных (`grep`), управлению процессами (`ps`, `pidof`, `kill`) и проверке использования диска (команды `df`, `du`).

Данные приложения позволяют управлять работой системы в консоли, что, в свою очередь, позволяет администрировать систему, подключаясь к ней без использования графического интерфейса.

Вышеописанные действия администрируют систему, в том числе используя относительно медленные соединения, где работа используя графический интерфейс может быть затруднена или некомфортна.