

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

**Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей**

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплина: Операционные системы

Студент: Кашкин Иван

Группа: НБИбд-01-21

Ст. билет №: 1032212958

Москва

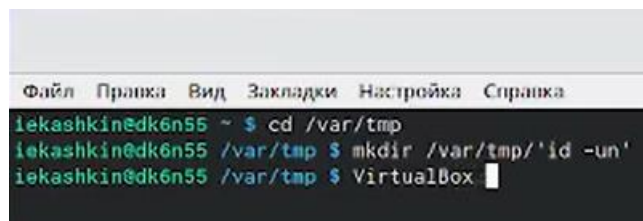
2022 г.

Цель работы:

Целью данной работы является приобретение практических навыков установки операционной системы на виртуальную машину, настройки минимально необходимых для дальнейшей работы сервисов.

Ход работы:

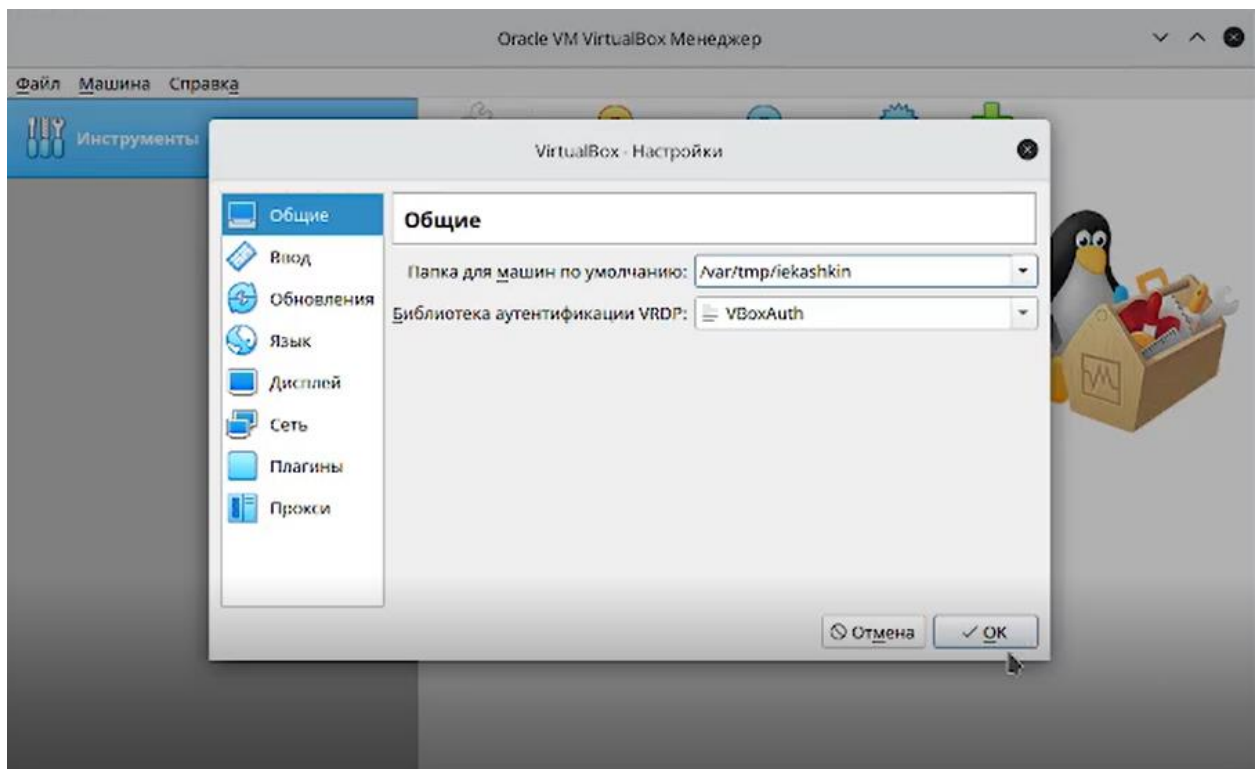
Загрузите в дисплейном классе операционную систему Linux. Осуществим вход в систему. Запустим терминал. Перейдем в каталог /var/tmp:, создадим каталог с именем пользователя и откроем виртуальную машину(рис. 1):



```
Файл  Правка  Вид  Закладки  Настройка  Справка
iekashkin@dk6n55 ~ $ cd /var/tmp
iekashkin@dk6n55 /var/tmp $ mkdir /var/tmp/'id -un'
iekashkin@dk6n55 /var/tmp $ VirtualBox
```

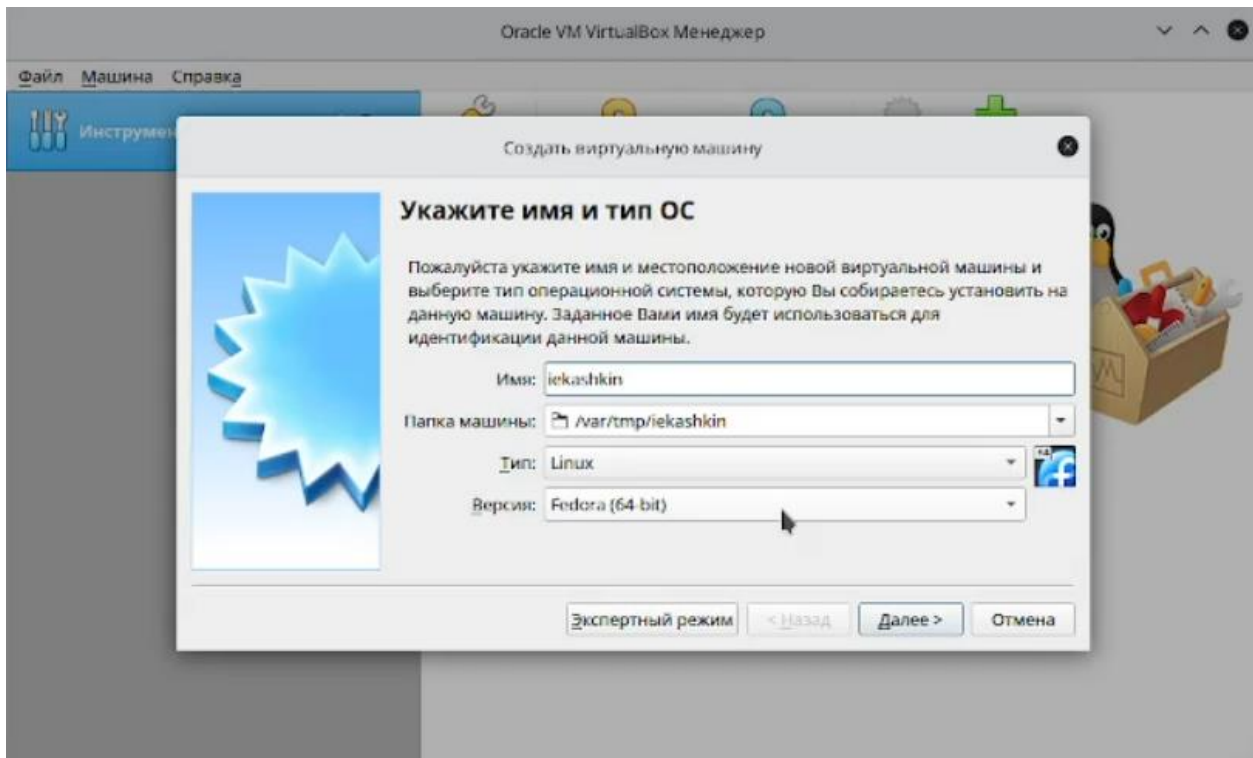
(Рис.1)

Проверим в свойствах VirtualBox месторасположение каталога для виртуальных машин. Был указан другой каталог, поэтому потребовалось изменить его, как указано на рис.2:



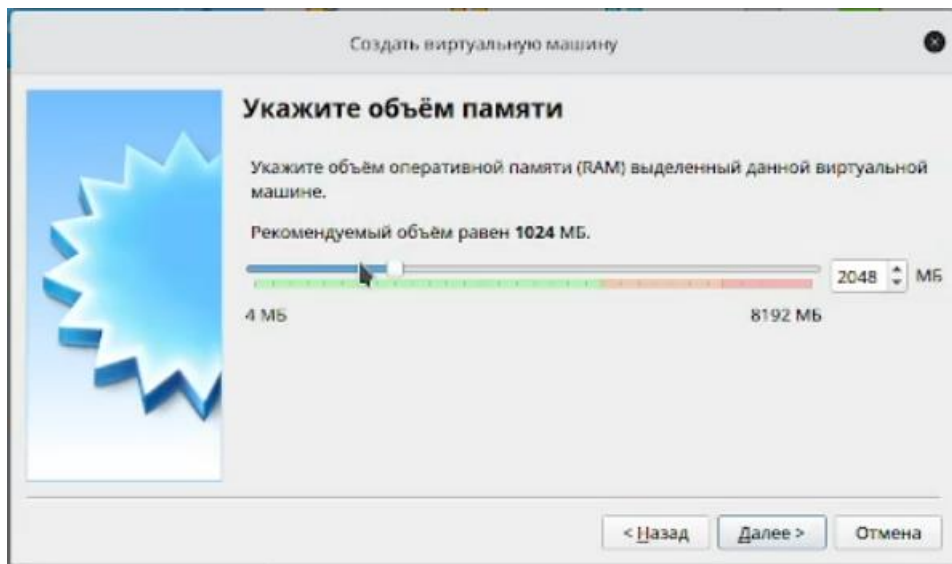
(Рис.2)

Создаем новую виртуальную машину. Укажем имя виртуальной машины – это имя пользователя, тип операционной системы — Linux, Fedora(рис.3)



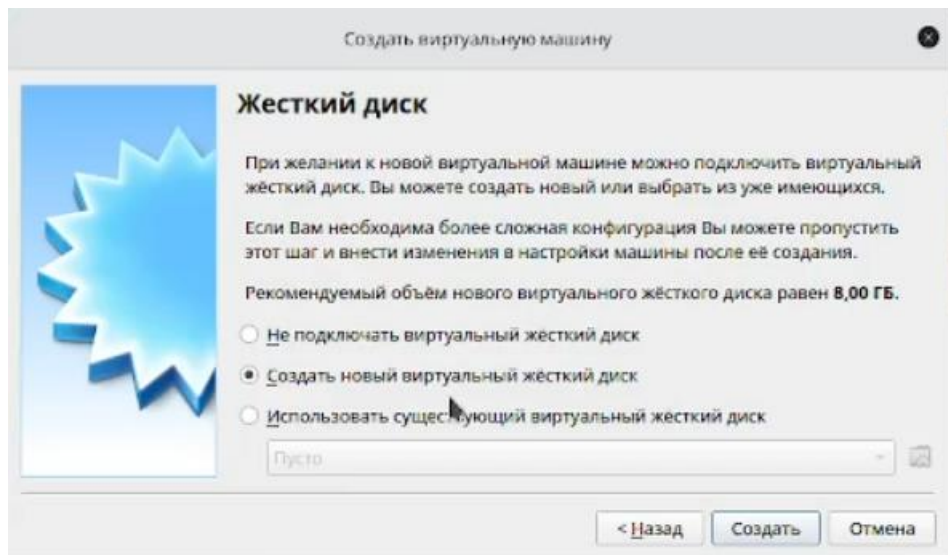
(Рис.3)

Указываем размер основной памяти виртуальной машины — от 2048 МБ (рис.4)

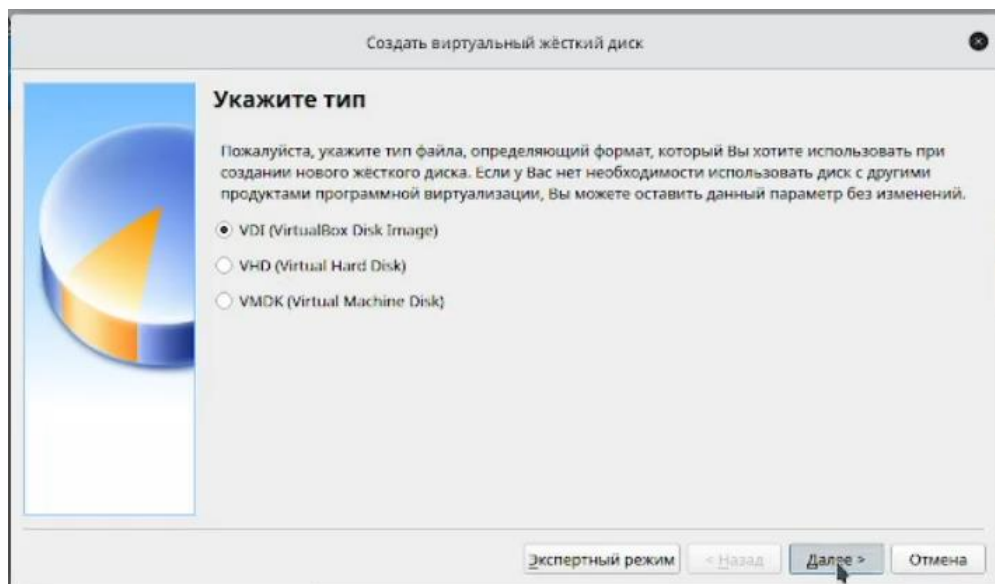


(Рис.4)

Задаем конфигурацию жёсткого диска - загрузочный, VDI (VirtualBox DiskImage), динамический виртуальный диск (рис.5; рис.6)

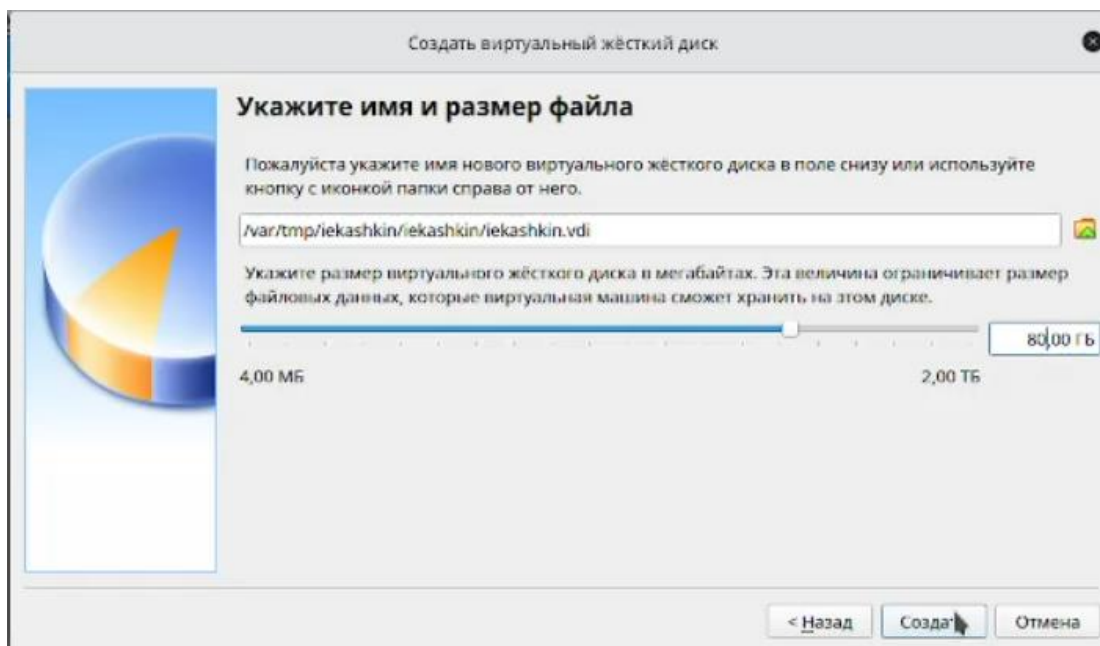


(Рис.5)



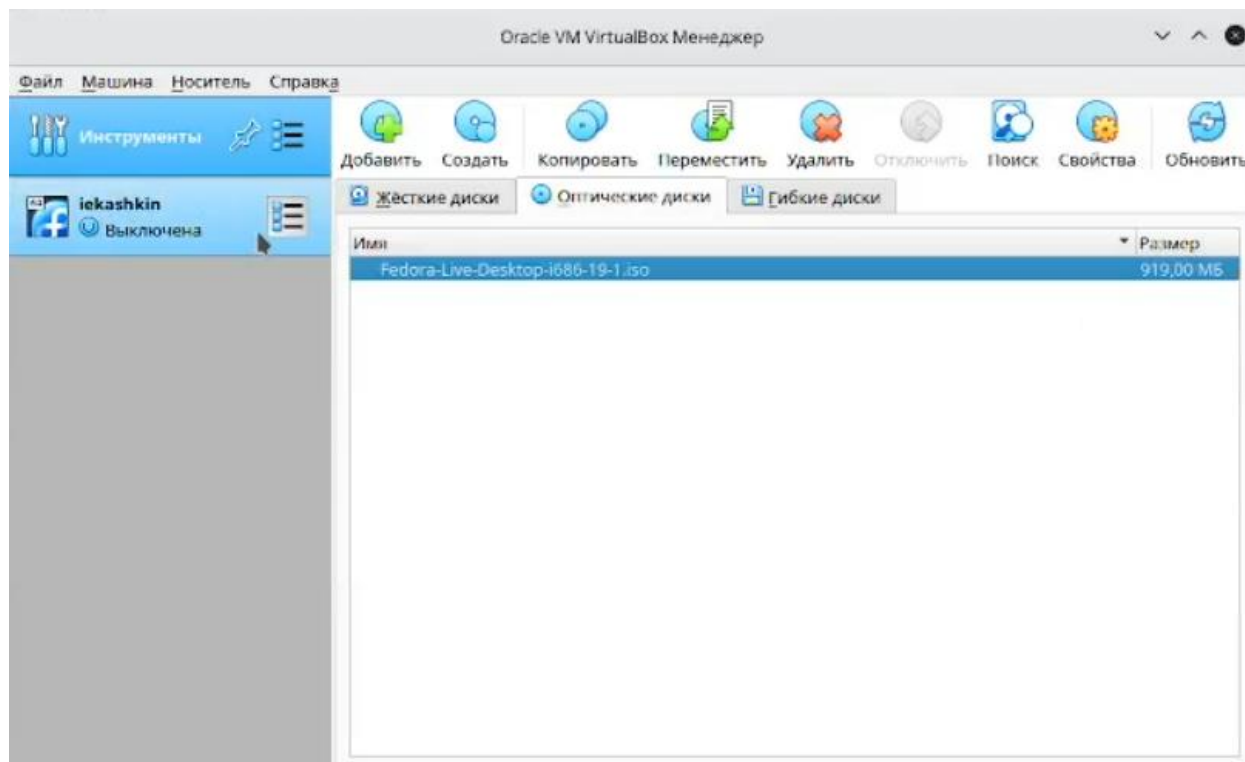
(Рис.6)

Задайте размер диска - 80 ГБ, его расположение - в данном случае /var/tmp/имя_пользователя/fedora.vdi (рис.7).

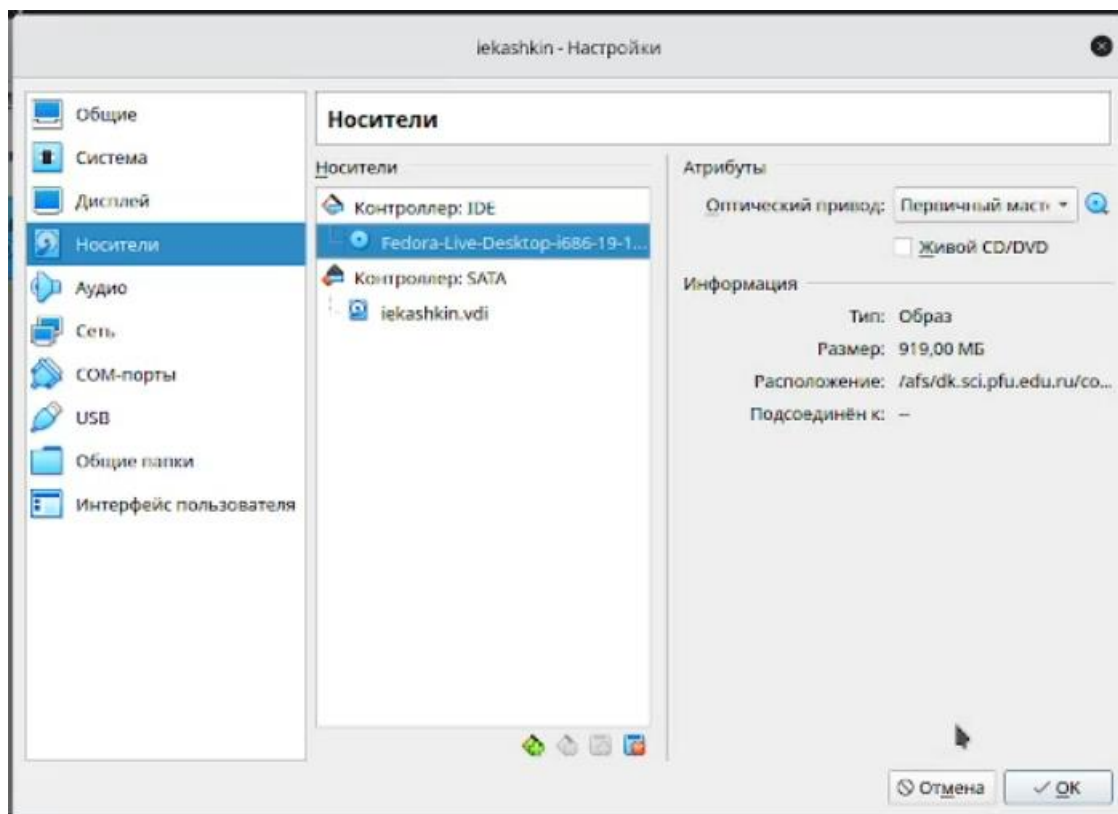


(Рис.7)

Выберем в VirtualBox «Свойства – Носители» Вашей виртуальной машины. Добавим новый привод оптических дисков и выберем образ «afs - dk.sci.pfu.edu.ru-common-files-iso-Fedora-Live-Desktop-i686-19-1.iso» (рис.8, рис.9)

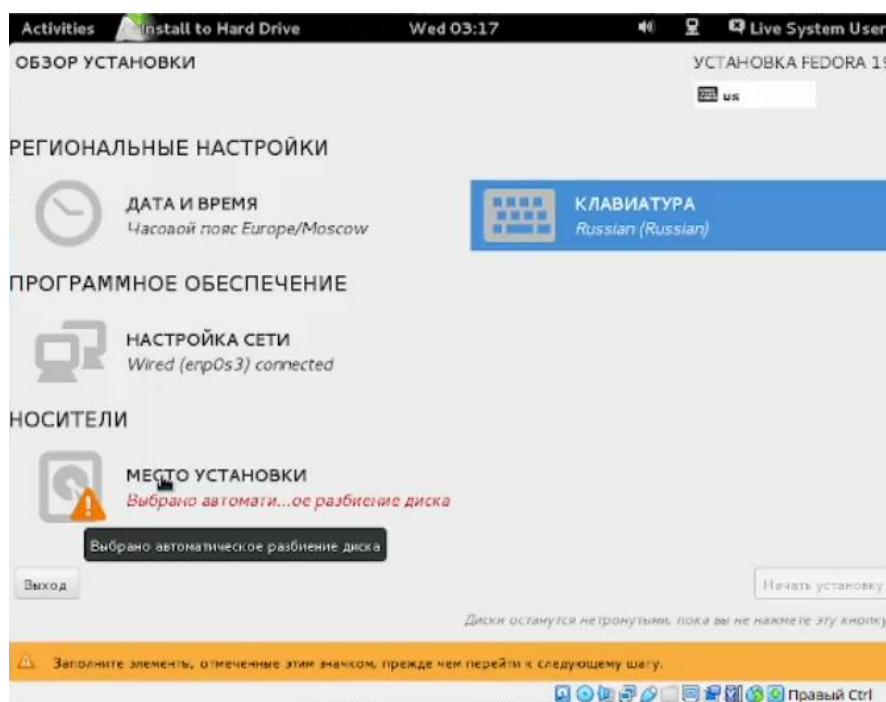


(Рис.8)



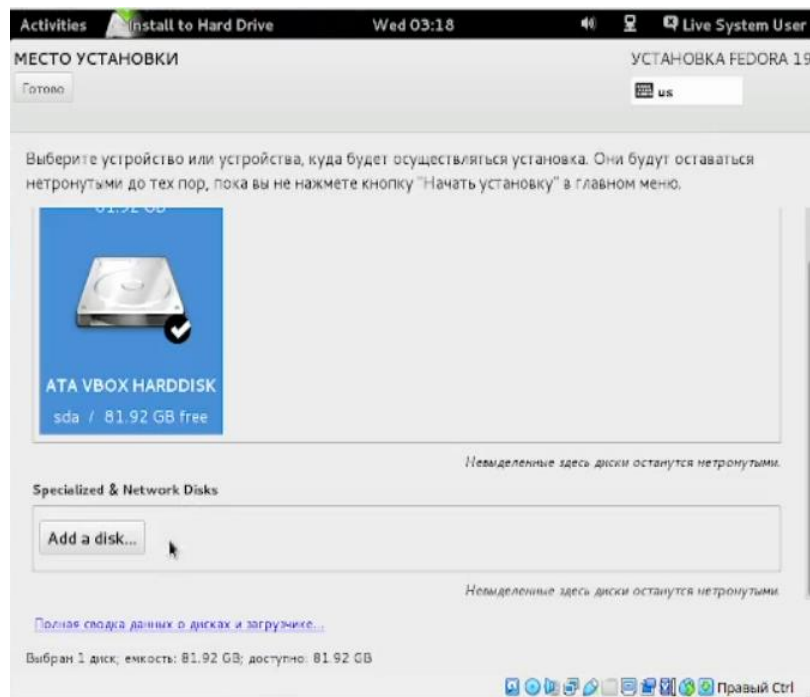
(Рис.9)

Запустим виртуальную машину, выберем язык интерфейса и перейдите к настройкам установки операционной системы (рис. 10).



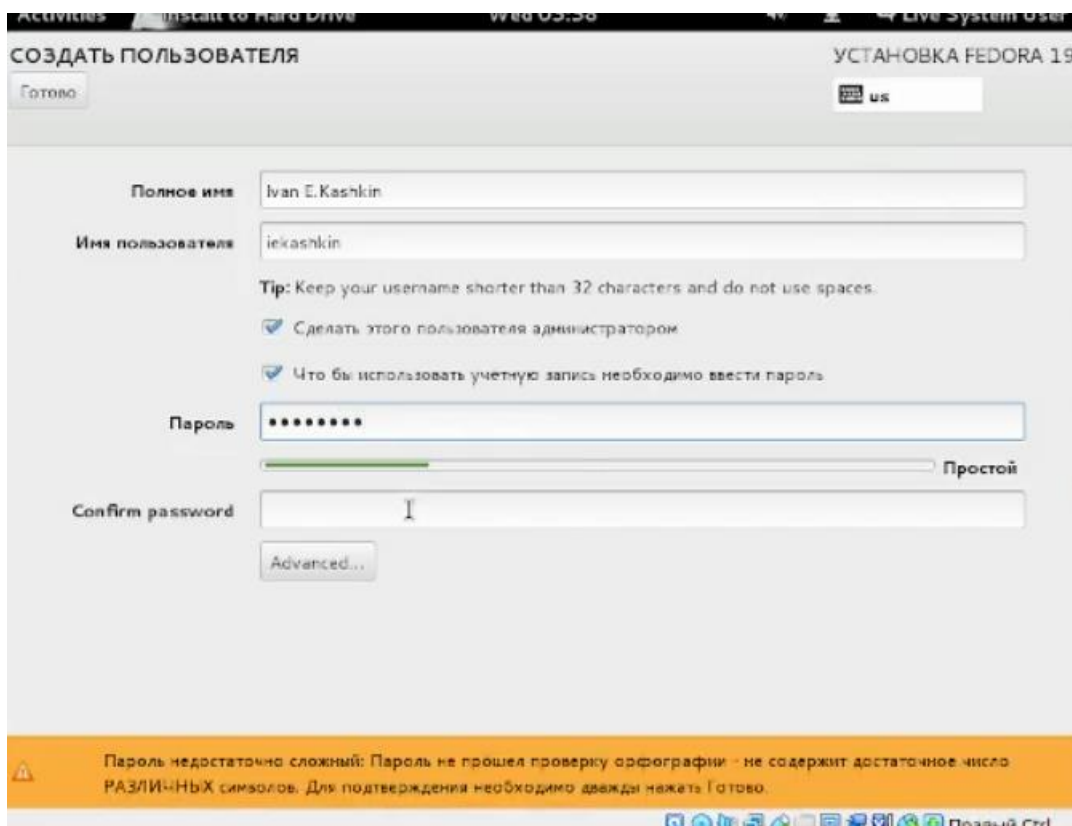
(Рис.10)

Место установки ОС оставьте без изменения (рис. 11).



(Рис.11)

Устанавливаем пароль для пользователя (Рис.12)

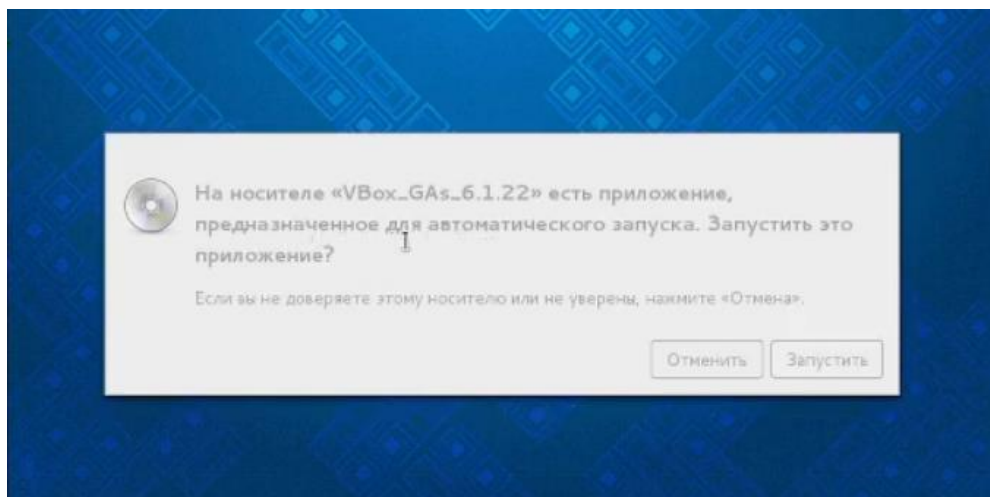


(Рис.12)

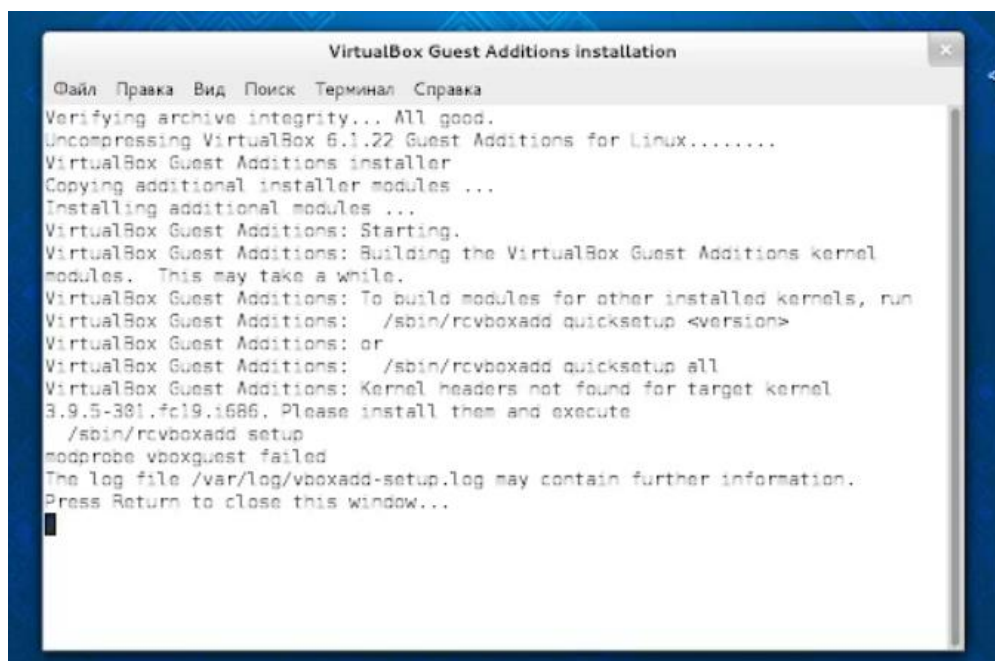
В VirtualBox оптический диск должен отключиться автоматически, но у меня это не произошло, поэтому необходимо отключить носитель информации с образом, выбрав: «Свойства-Носители-Fedora-Live-Desktop-i686-19-1.iso-Удалить устройство» (Рис.9)

Войдите в ОС под заданной вами при установке учётной записью. В меню «Устройства» виртуальной машины подключите образ диска дополнений гостевой ОС (рис.), для этого понадобился пароль root моей виртуальной ОС.(рис.13)

После загрузки дополнений нажмите Return или Enter (рис.14) и корректно перезагрузите виртуальную машину.



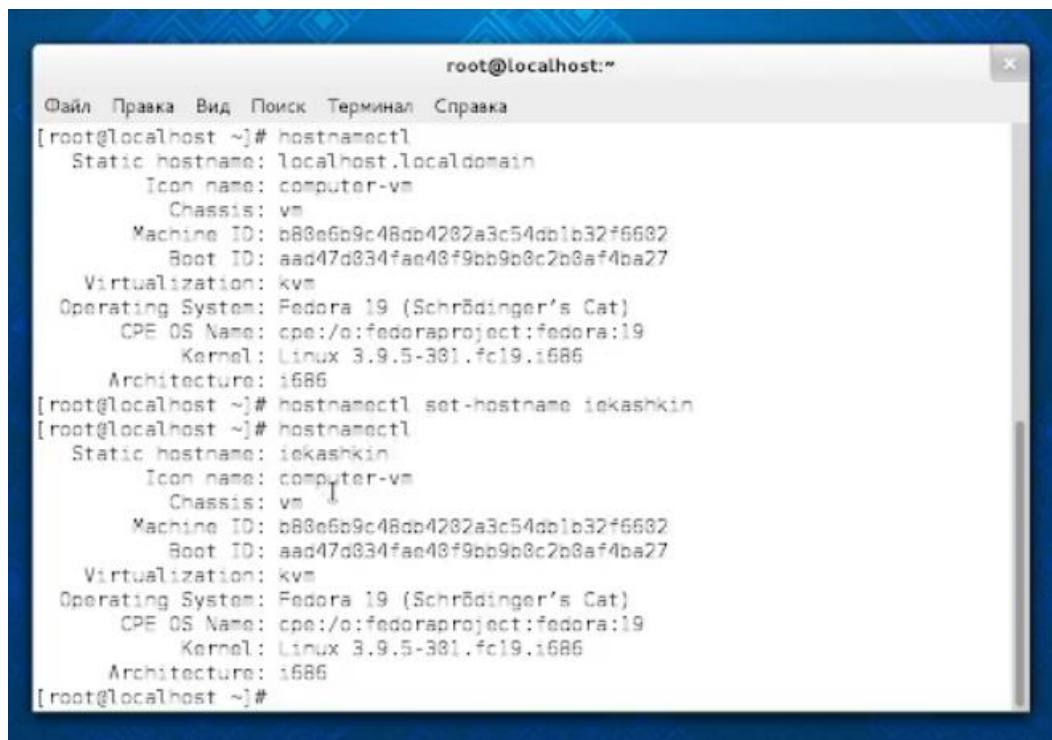
(Рис.13)



(Рис.14)

Устанавливаем имя хоста командой: «hostnamectl-set-hostname-iekashkin»

Проверьте, что имя хоста установлено верно: «hostname» (рис.15)



```
root@localhost:~  
Файл Правка Вид Поиск Терминал Справка  
[root@localhost ~]# hostnamectl  
  Static hostname: localhost.localdomain  
    Icon name: computer-vm  
    Chassis: vm  
  Machine ID: b88e6b9c46db4282a3c54db1b32f6682  
    Boot ID: aad47d834fae48f9bb9b8c2b8af4ba27  
  Virtualization: kvm  
Operating System: Fedora 19 (Schrödinger's Cat)  
  CPE OS Name: cpe:/o:fedoraproject:fedora:19  
    Kernel: Linux 3.9.5-381.fc19.i686  
  Architecture: i686  
[root@localhost ~]# hostnamectl set-hostname iekashkin  
[root@localhost ~]# hostnamectl  
  Static hostname: iekashkin  
    Icon name: computer-vm  
    Chassis: vm  
  Machine ID: b88e6b9c46db4282a3c54db1b32f6682  
    Boot ID: aad47d834fae48f9bb9b8c2b8af4ba27  
  Virtualization: kvm  
Operating System: Fedora 19 (Schrödinger's Cat)  
  CPE OS Name: cpe:/o:fedoraproject:fedora:19  
    Kernel: Linux 3.9.5-381.fc19.i686  
  Architecture: i686  
[root@localhost ~]#
```

(Рис.15)

Домашнее задание:

Дождемся загрузки графического окружения и откройте терминал. В окне терминала проанализируйте последовательность загрузки системы, выполнив команду dmesg.

Используем команду «dmesg | grep -i "то, что ищем"», чтобы найти необходимую информацию (рис.16, рис.17):

1. Версия ядра Linux: команда «dmesg | grep -i "Linux version"»
2. Частота процессора: команда «dmesg | grep -i "MHz"»
3. Модель процессора: команда «dmesg | grep -i "CPU0"»
4. Объем доступной оперативной памяти: команда «dmesg | grep -i "Memory"»
5. Тип обнаруженного гипервизора: команда «dmesg | grep -i "Hypervisor detected"»
6. Тип файловой системы корневого раздела и последовательность монтирования файловых систем: команда «dmesg | grep -i "Mount"».

```
root@localhost:~  
Файл Правка Вид Поиск Терминал Справка  
[ 137.864728] SELinux: initialized (dev fusectl, type fusectl), uses genfs_cont  
exts  
[ 137.148885] SELinux: initialized (dev fuse, type fuse), uses genfs_contexts  
[ 158.555688] ISO 9660 Extensions: Microsoft Joliet Level 3  
[ 158.631285] ISO 9660 Extensions: RRIP_1991A  
[ 158.631353] SELinux: initialized (dev sr0, type iso9660), uses genfs_contexts  
[ 184.468919] TCP: lp registered  
[ 258.637813] SELinux: 2848 avtab hash slots, 188852 rules.  
[ 258.677885] SELinux: 2848 avtab hash slots, 188852 rules.  
[ 259.947791] SELinux: 8 users, 82 roles, 4468 types, 252 booleans, 1 sensors, 1824  
cats  
[ 259.947796] SELinux: 83 classes, 188852 rules  
[ 264.273693] LVM: Logical Volume autoactivation enabled.  
[ 264.273699] LVM: Activation generator successfully completed.  
[ 264.647691] LVM: Logical Volume autoactivation enabled.  
[ 264.647695] LVM: Activation generator successfully completed.  
[ 264.778769] LVM: Logical Volume autoactivation enabled.  
[ 264.778774] LVM: Activation generator successfully completed.  
[ 269.171167] SELinux: 2848 avtab hash slots, 188852 rules.  
[ 269.285381] SELinux: 2848 avtab hash slots, 188852 rules.  
[ 278.114817] SELinux: 8 users, 82 roles, 4468 types, 252 booleans, 1 sensors, 1824  
cats  
[ 278.114822] SELinux: 83 classes, 188852 rules  
[root@localhost ~]#
```

(Рис.16)

```
root@localhost:~  
Файл Правка Вид Поиск Терминал Справка  
[ 137.864728] SELinux: initialized (dev fusectl, type fusectl), uses genfs_cont  
exts  
[ 137.148885] SELinux: initialized (dev fuse, type fuse), uses genfs_contexts  
[ 158.555688] ISO 9660 Extensions: Microsoft Joliet Level 3  
[ 158.631285] ISO 9660 Extensions: RRIP_1991A  
[ 158.631353] SELinux: initialized (dev sr0, type iso9660), uses genfs_contexts  
[ 184.468919] TCP: lp registered  
[ 258.637813] SELinux: 2848 avtab hash slots, 188852 rules.  
[ 258.677885] SELinux: 2848 avtab hash slots, 188852 rules.  
[ 259.947791] SELinux: 8 users, 82 roles, 4468 types, 252 booleans, 1 sensors, 1824  
cats  
[ 259.947796] SELinux: 83 classes, 188852 rules  
[ 264.273693] LVM: Logical Volume autoactivation enabled.  
[ 264.273699] LVM: Activation generator successfully completed.  
[ 264.647691] LVM: Logical Volume autoactivation enabled.  
[ 264.647695] LVM: Activation generator successfully completed.  
[ 264.778769] LVM: Logical Volume autoactivation enabled.  
[ 264.778774] LVM: Activation generator successfully completed.  
[ 269.171167] SELinux: 2848 avtab hash slots, 188852 rules.  
[ 269.285381] SELinux: 2848 avtab hash slots, 188852 rules.  
[ 278.114817] SELinux: 8 users, 82 roles, 4468 types, 252 booleans, 1 sensors, 1824  
cats  
[ 278.114822] SELinux: 83 classes, 188852 rules  
[root@localhost ~]# dmesg gr
```

(Рис.17)

Контрольные вопросы:

1) Учётная запись пользователя содержит:

- Имя пользователя (user name)
- Идентификационный номер пользователя (UID)
- Идентификационный номер группы (GID).
- Пароль (password)

- Полное имя (full name)
- Домашний каталог (home directory)
- Начальную оболочку (login shell)

2) Команды терминала:

- Для получения справки по команде: `man`. Например, команда «`man ls`» выведет справку о команде «`ls`».
- Для перемещения по файловой системе: `cd`. Например, команда «`cd newdir`» осуществляет переход в каталог `newdir`.
- Для просмотра содержимого каталога: `ls`. Например, команда «`ls -a ~/newdir`» отобразит имена скрытых файлов в каталоге `newdir`.
- Для определения объёма каталога: `du`. Например, команда «`du -k ~/newdir`» выведет размер каталога `newdir` в килобайтах.
- Для создания / удаления каталогов / файлов: `mkdir` / `rmdir` / `rm`. Например, команда «`mkdir -p ~/newdir1/newdir2`» создаст иерархическую цепочку подкаталогов, создав каталоги `newdir1` и `newdir2`; команда «`rmdir -v ~/newdir`» удалит каталог `newdir`; команда «`rm -r ~/newdir`» так же удалит каталог `newdir`.
- Для задания определённых прав на файл / каталог: `chmod` [опции] [путь]. Например, команда «`chmod g+r ~/text.txt`» даст группе право на чтение файла `text.txt`.
- Для просмотра истории команд: `history`. Например, команда «`history 7`» покажет список последних 7 команд.

3) Файловая система имеет два значения: с одной стороны – это архитектура хранения битов на жестком диске, с другой – это организация каталогов в соответствии с идеологией Unix.

Файловая система (англ. «file system») – это архитектура хранения данных в системе, хранение данных в оперативной памяти и доступа к конфигурации ядра. Файловая система устанавливает физическую и логическую структуру файлов, правила их создания и управления ими. В физическом смысле файловая система

Linux представляет собой пространство раздела диска, разбитое на блоки фиксированного размера. Их размер кратен размеру сектора: 1024, 2048, 4096 или 8120 байт.

Существует несколько типов файловых систем:

- XFS – начало разработки 1993 год, фирма Silicon Graphics, в мае 2000 года предстала в GNU GPL, для пользователей большинства Linux систем стала доступна в 2001-2002 гг. Отличительная черта системы – прекрасная поддержка больших файлов и файловых томов, 8 эксбибайт ($8 \cdot 260$ байт) для 64-х битных систем.

- ReiserFS (Reiser3) – одна из первых журналируемых файловых систем под Linux, разработана Namesys, доступна с 2001 г. Максимальный объём тома для этой системы равен 16 тебибайт ($16 \cdot 240$ байт).

- JFS (Journaled File System) – файловая система, детище IBM, явившееся миру в далёком 1990 году для ОС AIX (Advanced Interactive eXecutive). В виде первого стабильного релиза, для пользователей Linux, система стала доступна в 2001 году. Из плюсов системы – хорошая масштабируемость. Из минусов – не особо активная поддержка на протяжении всего жизненного цикла. Максимальный размер тома 32 пэбибайта ($32 \cdot 250$ байт).

- ext (extended filesystem) – появилась в апреле 1992 года, это была первая файловая система, изготовленная специально под нужды Linux ОС. Разработана Remy Card с целью преодолеть ограничения файловой системы Minix.

- ext2 (second extended file system) – была разработана Remy Card в 1993 году. Не журналируемая файловая система, это был основной её недостаток, который исправит ext3.

- ext3 (third extended filesystem) – по сути расширение исконной для Linux ext2, способное к журналированию. Разработана Стивеном Твиди в 1999 году, включена в основное ядро Linux в ноябре 2001 года. На фоне других своих сослуживцев обладает более скромным размером пространства, до 4 тебибайт ($4 \cdot 240$ байт) для 32-х разрядных систем. На данный момент является наиболее стабильной и поддерживаемой файловой системой в среде Linux.

- Reiser4— первая попытка создать файловую систему нового поколения для Linux. Впервые представленная в 2004 году, система включает в себя такие передовые технологии как транзакции, задержка выделения пространства, а так же встроенная возможность кодирования и сжатия данных. Ханс Рейзер (Hans Reiser) — главный разработчик системы.

-xt4 — попытка создать 64-х битную ext3 способную поддерживать большой размер файловой системы (1 эксбибайт). Позже добавились возможности — непрерывные области дискового пространства, задержка выделения пространства, онлайн дефрагментация и прочие. Обеспечивается прямая совместимость с системой ext3 и ограниченная обратная совместимость при недоступной способности к непрерывным областям дискового пространства.

- Btrfs (B-tree FS или Butter FS)— проект изначально начатый компанией Oracle, впоследствии поддержанный большинством Linux систем. Ключевыми особенностями данной файловой системы являются технологии: copy-on-write, позволяющая сделать снимки областей диска (снапшоты), которые могут пригодиться для последующего восстановления; контроль за целостностью данных и метаданных (с повышенной гарантией целостности); сжатие данных; оптимизированный режим для накопителей SSD (задаётся при монтировании) и прочие. Немаловажным фактором является возможность перехода с ext3 на Btrfs. С августа 2008 года данная система выпускается под GNU GPL.

- Tux2 — известная, но так и не анонсированная публично файловая система. Создатель Дэниэл Филипс (Daniel Phillips). Система базируется на алгоритме «Фазового Древа», который как и журналирование защищает файловую систему от сбоев. Организована как надстройка на ext2.

- Tux3 — система создана на основе FUSE (Filesystem in Userspace), специального модуля для создания файловых систем на Unix платформах. Данный проект ставит перед собой цель избавиться от привычного журналирования, взамен предлагая версионное восстановление (состояние в определённый промежуток времени). Преимуществом используемой в данном случае версионной системы, является

способ описания изменений, где для каждого файла создаётся изменённая копия, а не переписывается текущая версия.

- Xiafs—задумка и разработка данной файловой системы принадлежат Frank Xia, основана на файловой системе MINIX. В настоящее время считается устаревшей и практически не используется. Наряду с ext2 разрабатывалась, как замена системе ext. В декабре 1993 года система была добавлена в стандартное ядро Linux. И хотя система обладала большей стабильностью и занимала меньше дискового пространства под контрольные структуры —она оказалась слабее ext2, ведущую роль сыграли ограничения максимальных размеров файла и раздела, а так же способность к дальнейшему расширению.

- ZFS (Zettabyte File System)—изначально созданная в Sun Microsystems файловая система, для небезызвестной операционной системы Solaris в 2005 году. Отличительные особенности —отсутствие фрагментации данных как таковой, возможности по управлению снапшотами (snapshots), пулами хранения (storage pools), варьируемый размер блоков, 64-х разрядный механизм контрольных сумм, а так же способность адресовать 128 бит информации. В Linux системах может использоваться посредством FUSE.

4) Команда «findmnt» или «findmnt--all» будет отображать все подмонтированные файловые системы или искать файловую систему.

5) Основные сигналы (каждый сигнал имеет свой номер), которые используются для завершения процесса:

- SIGINT —самый безобидный сигнал завершения, означает Interrupt. Он отправляется процессу, запущенному из терминала с помощью сочетания клавиш Ctrl+C. Процесс правильно завершает все свои действия и возвращает управление;

- SIGQUIT —это еще один сигнал, который отправляется с помощью сочетания клавиш, программе, запущенной в терминале. Он сообщает ей что нужно завершиться и программа может выполнить корректное завершение или проигнорировать сигнал. В отличие от

предыдущего, она генерирует дампы памяти. Сочетание клавиш Ctrl+;/

- **SIGHUP** –сообщает процессу, что соединение с управляющим терминалом разорвано, отправляется, в основном, системой при разрыве соединения с интернетом;

- **SIGTERM** –немедленно завершает процесс, но обрабатывается программой, поэтому позволяет ей завершить дочерние процессы и освободить все ресурсы;

- **SIGKILL** –тоже немедленно завершает процесс, но, в отличие от предыдущего варианта, он не передается самому процессу, а обрабатывается ядром. Поэтому ресурсы и дочерние процессы остаются запущенными. Также для передачи сигналов процессам в Linux используется утилита **kill**, её синтаксис: **kill [-сигнал][pid_процесса](PID–уникальный идентификатор процесса)**. Сигнал представляет собой один из выше перечисленных сигналов для завершения процесса. Перед тем, как выполнить остановку процесса, нужно определить его PID. Для этого используют команды **ps** и **grep**. Команда **ps** предназначена для вывода списка активных процессов в системе и информации о них. Команда **grep** запускается одновременно с **ps** (в канале) и будет выполнять поиск по результатам команды **ps**. Утилита **killall** –это оболочка для **kill**, она ведет себя точно так же, и имеет тот же синтаксис, только в качестве идентификатора процесса ей нужно передать его имя.

killall работает аналогично двум предыдущим утилитам. Она тоже принимает имя процесса в качестве параметра и ищет его PID в директории **/proc**. Но эта утилита обнаружит все процессы с таким именем и завершит их.

Вывод:

В ходе данной лабораторной работы я приобрел практические навыки установки операционной системы на виртуальную машину и настройки минимально необходимых для дальнейшей работы сервисов.