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:

Андреев Владислав

Владимирович

Группа:

НПМбд-01-20

МОСКВА

2020 г.

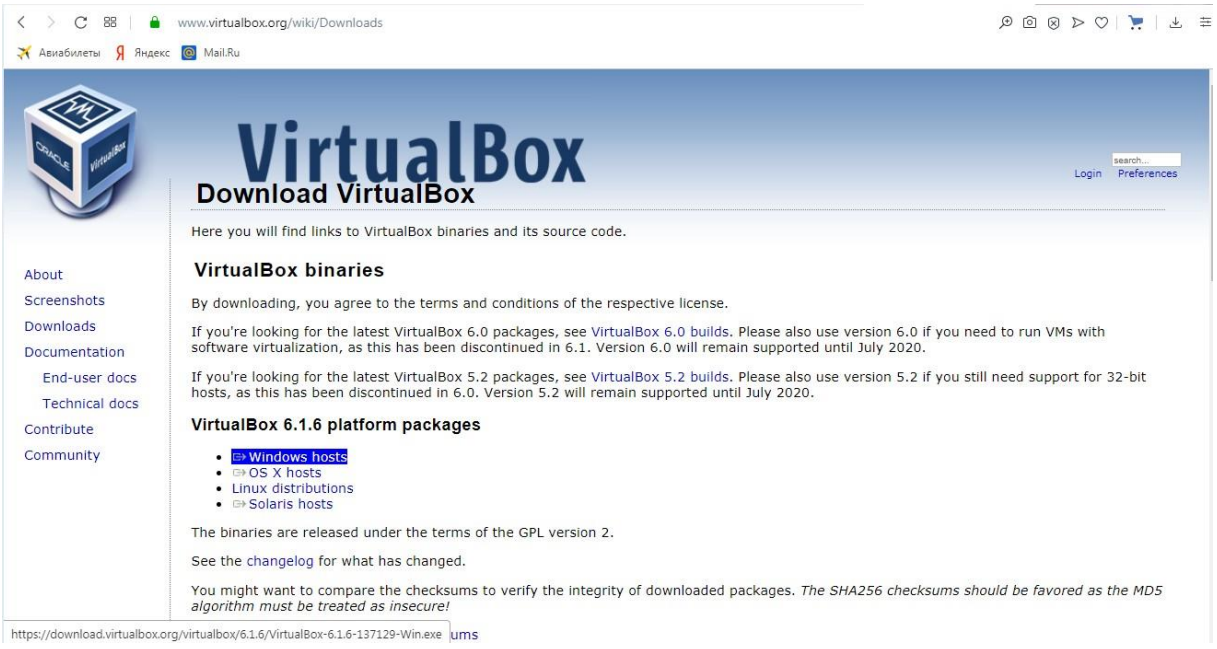
1. **Цель работы:**

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

2. **Ход работы:**

Для начала скачаем VirtualBox, необходимую для запуска виртуальных машин. Скачать можно на официальном сайте: https://www.virtualbox.org *(Рисунок 1)*. Необходимо выбрать версию своей операционной системы *(Рисунок 2)*.

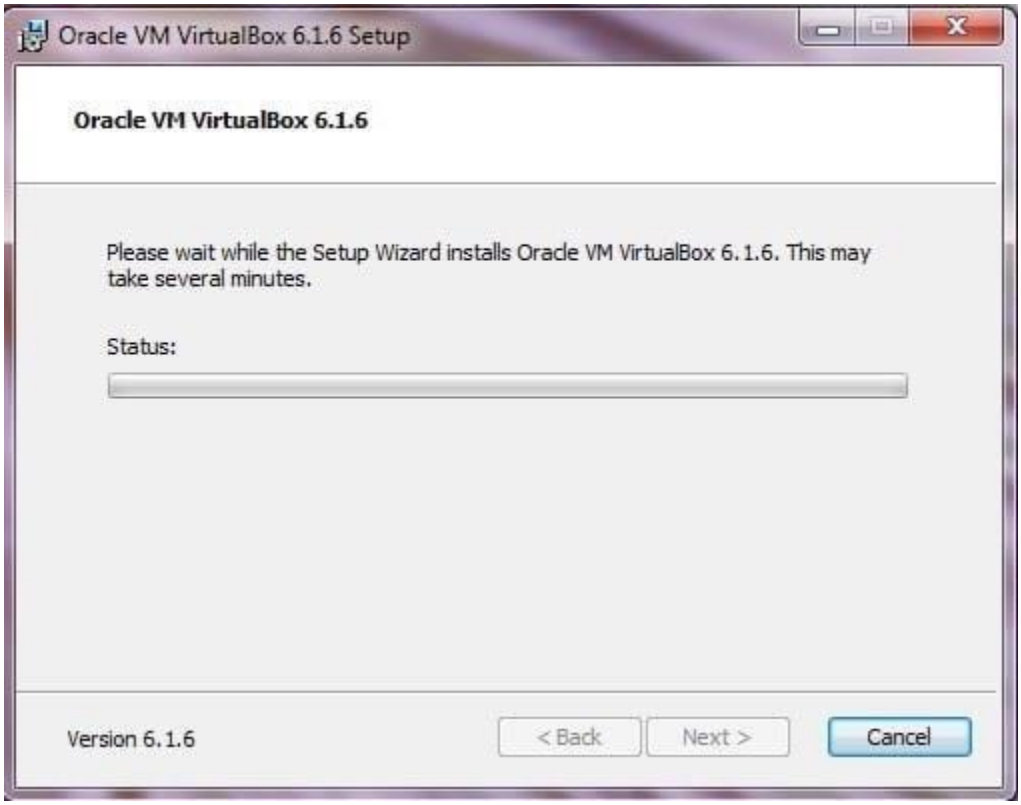
(Рисунок 1)



(Рисунок 2)

Далее выполняем установку скачанного файла *(Рисунки 3, 4)*.

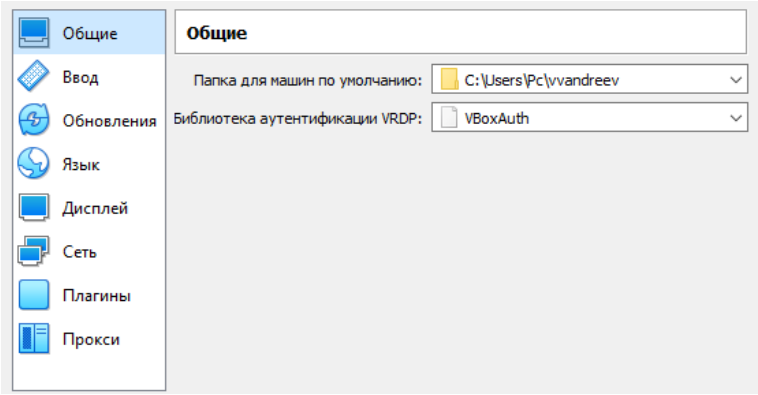
(Рисунок 3)



(Рисунок 4)

Далее создаём на рабочем столе папку, в которой будет храниться наша

виртуальная машина. Имя папки – имя пользователя (логин студента в дисплейном классе). В данном случае «vvandreev». Проверяем в свойствах VirtualBox месторасположение папки для виртуальных машин. Для этого открываем VirtualBox, далее «Файл» → «Свойства» → вкладка «Общие» и в поле «Папка для машин по умолчанию» указываем путь к папке, созданной ранее *(Рисунок 5)*.

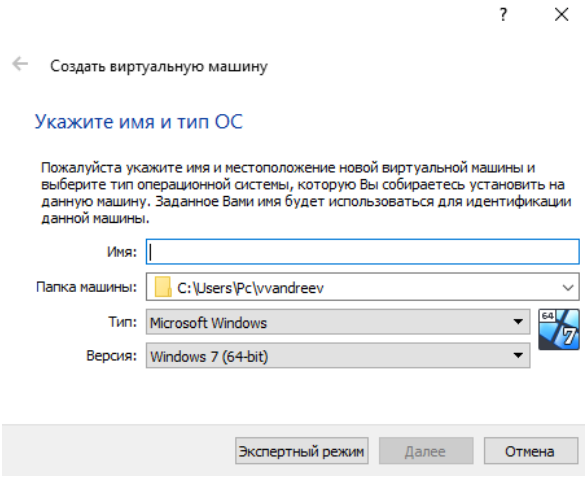


(Рисунок 5)

Переходим к созданию виртуальной машины. Для этого нажимаем «Машина» → «Создать» *(Рисунок 6)*.

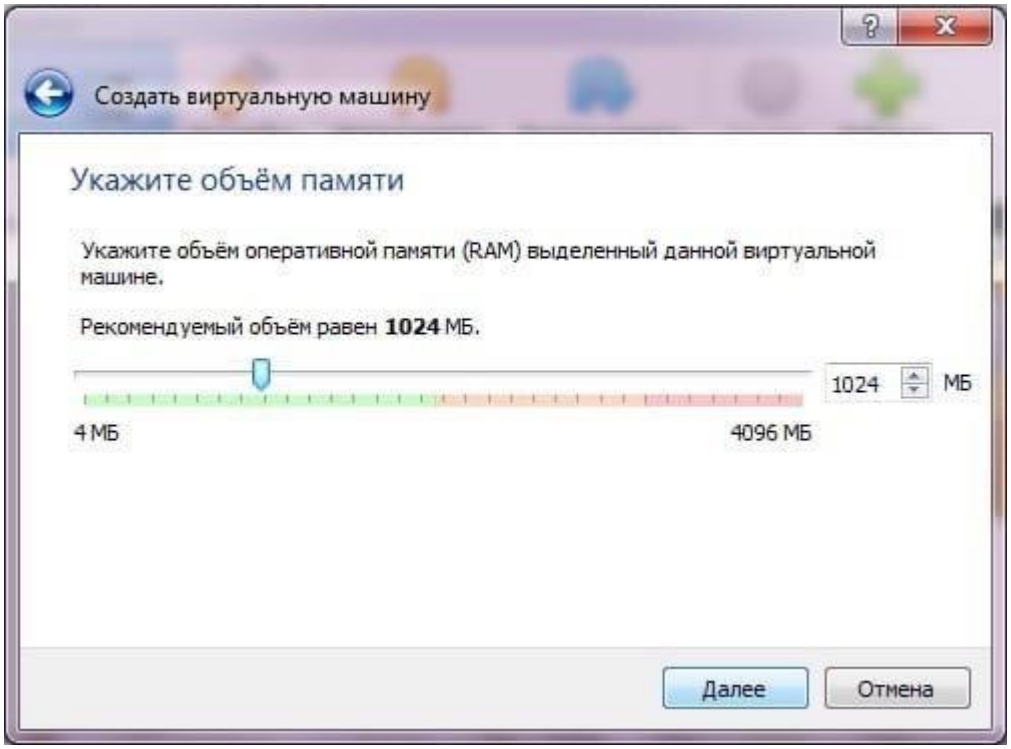
(Рисунок 6)

Указываем имя виртуальной машины (логин в дисплейном классе, «vvandreev») и тип операционной системы – Linux, RedHat (64-bit, т.к. на компьютере установлен 64 битный процессор) *(Рисунок 7)*.



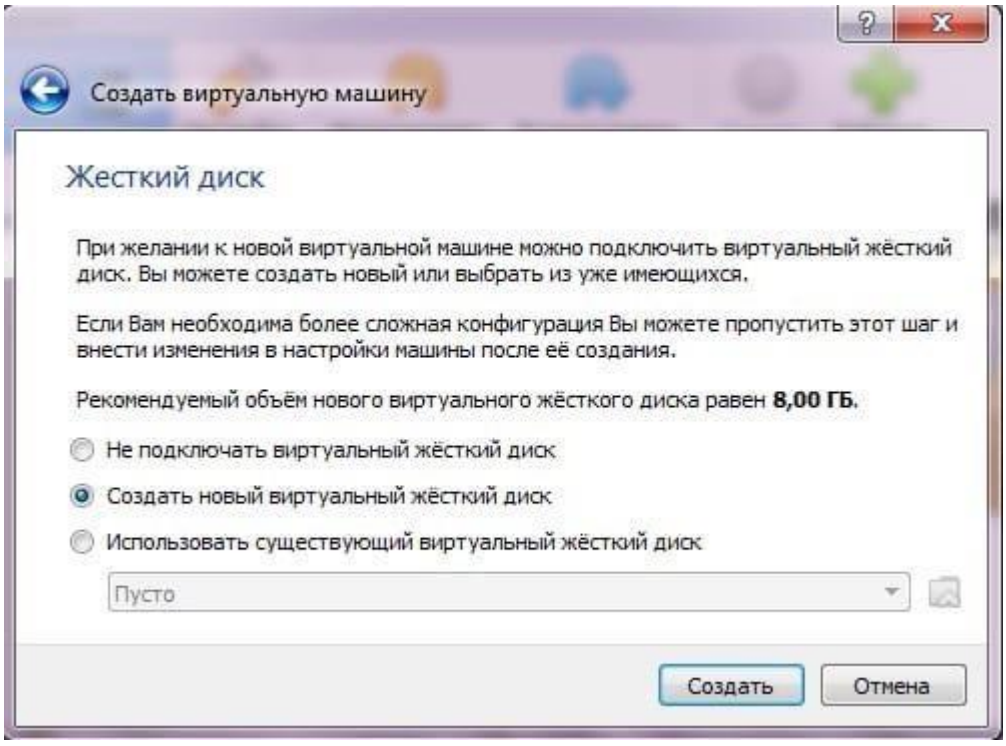
(Рисунок 7)

Указываем размер основной памяти виртуальной машины – 1024 МБ *(Рисунок 8)*.



(Рисунок 8)

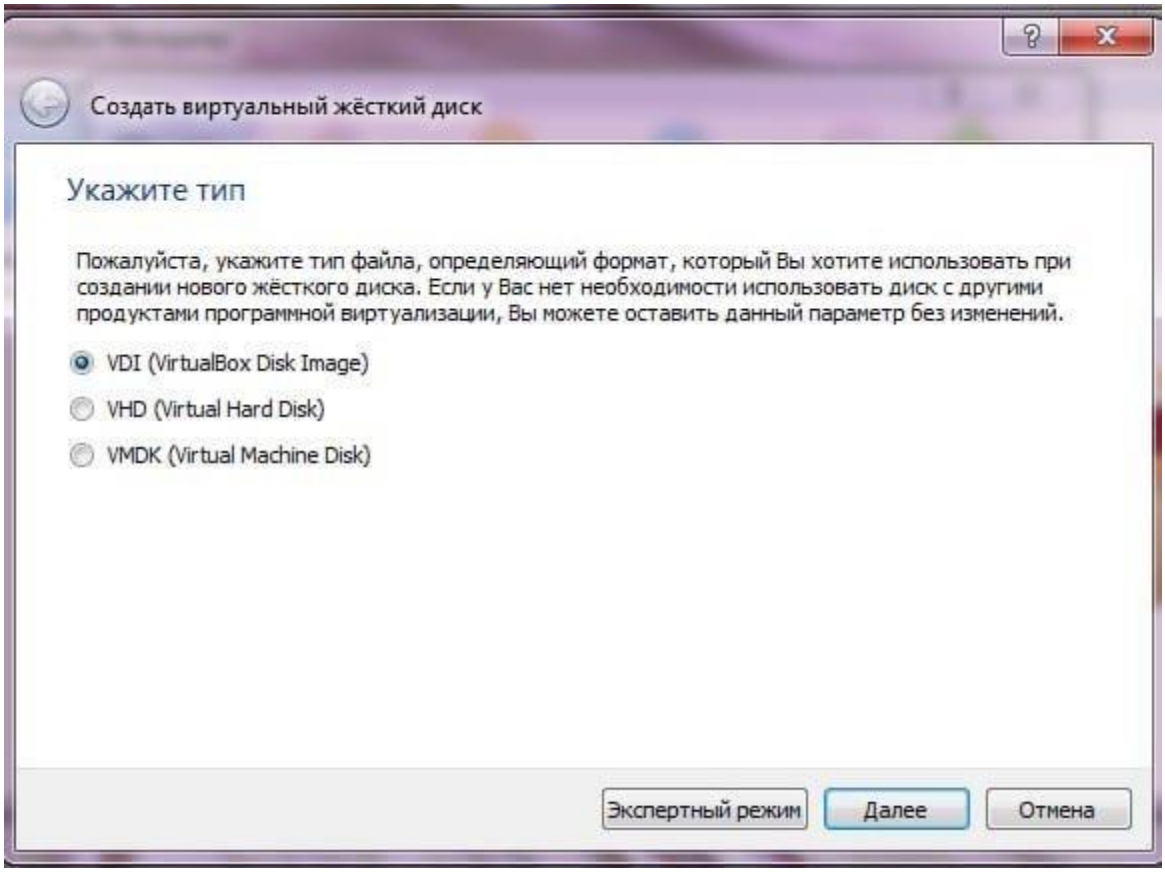
Создаём новый виртуальный жёсткий диск *(Рисунок 9)*.



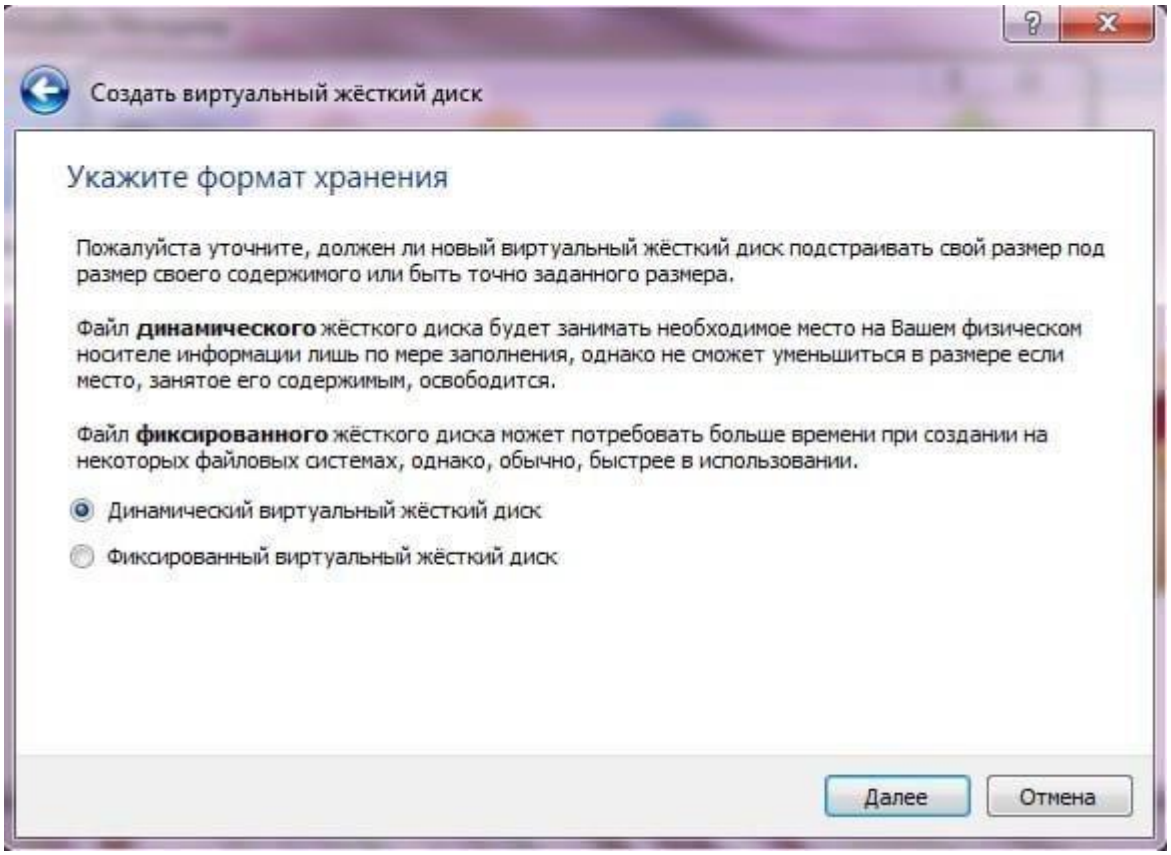
(Рисунок 9)

Задаём конфигурацию жёсткого диска – VDI (BirtualBox Disk Image),

динамический виртуальный жёсткий диск *(Рисунки 10, 11)*.

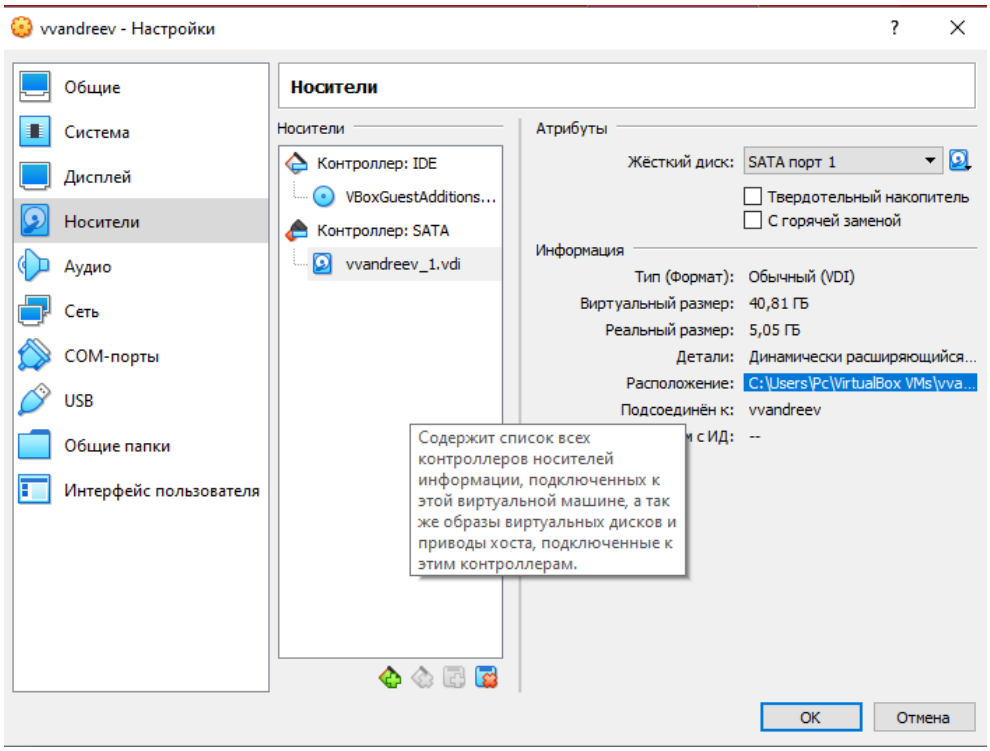


(Рисунок 10)



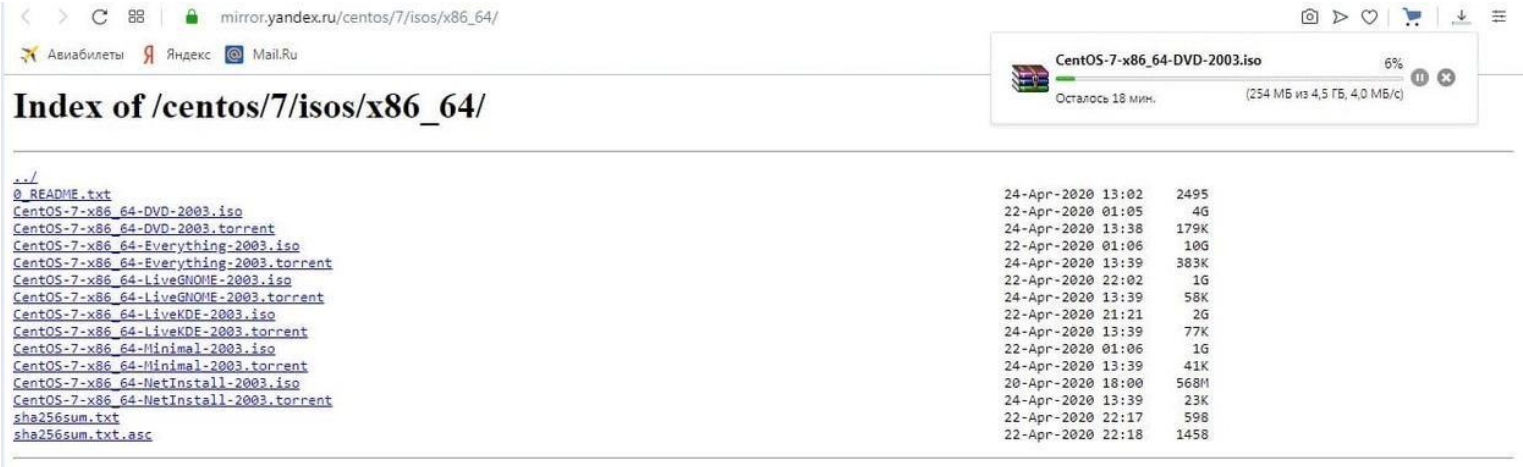
(Рисунок 11)

Задаём расположение и размер диска. В данном случае: «C:\Users\Pc\VirtualBoxVms\vvandreev\vvandreev.vdi»;20 ГБ, но рекомендуется 40 ГБ (или больше) *(Рисунок 12).*



(Рисунок 12)

Далее нам необходимо скачать образ операционной системы. В данном случае – это «CentOS-7-x86_64-DVD-2003.iso». Скачать можно на сайте: https://mirror.yandex.ru/centos/7/isos/x86_64/ *(Рисунок 13).*

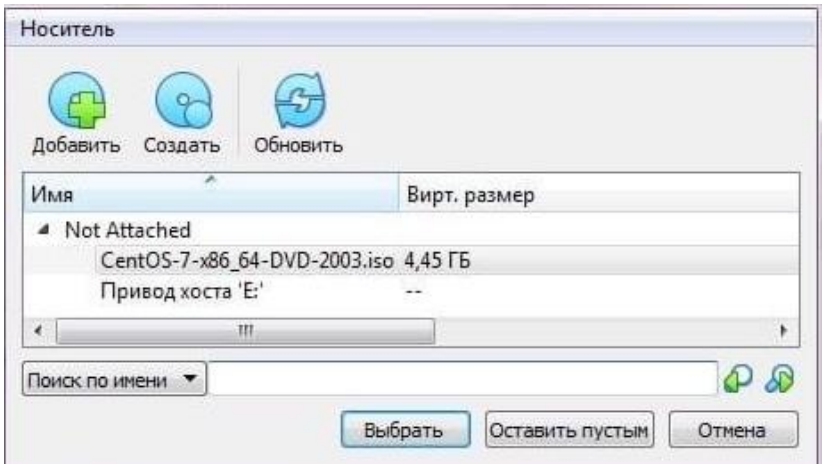


(Рисунок 13)

Теперь в VirtualBox для нашей виртуальной машины выбираем «Свойства» →

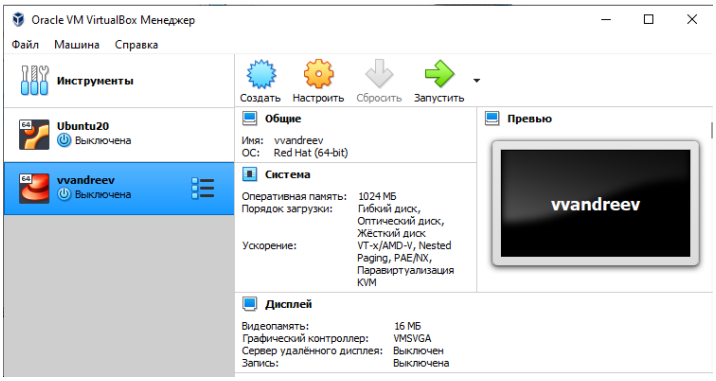
«Носители». Добавляем новый привод оптических дисков и выбираем образ «CentOS-7-x86_64-DVD-2003.iso» *(Рисунок 14)*.

(Рисунок 14)



(Рисунок 15)

После этого необходимо запустить виртуальную машину и продолжить настройку *(Рисунок 16)*.

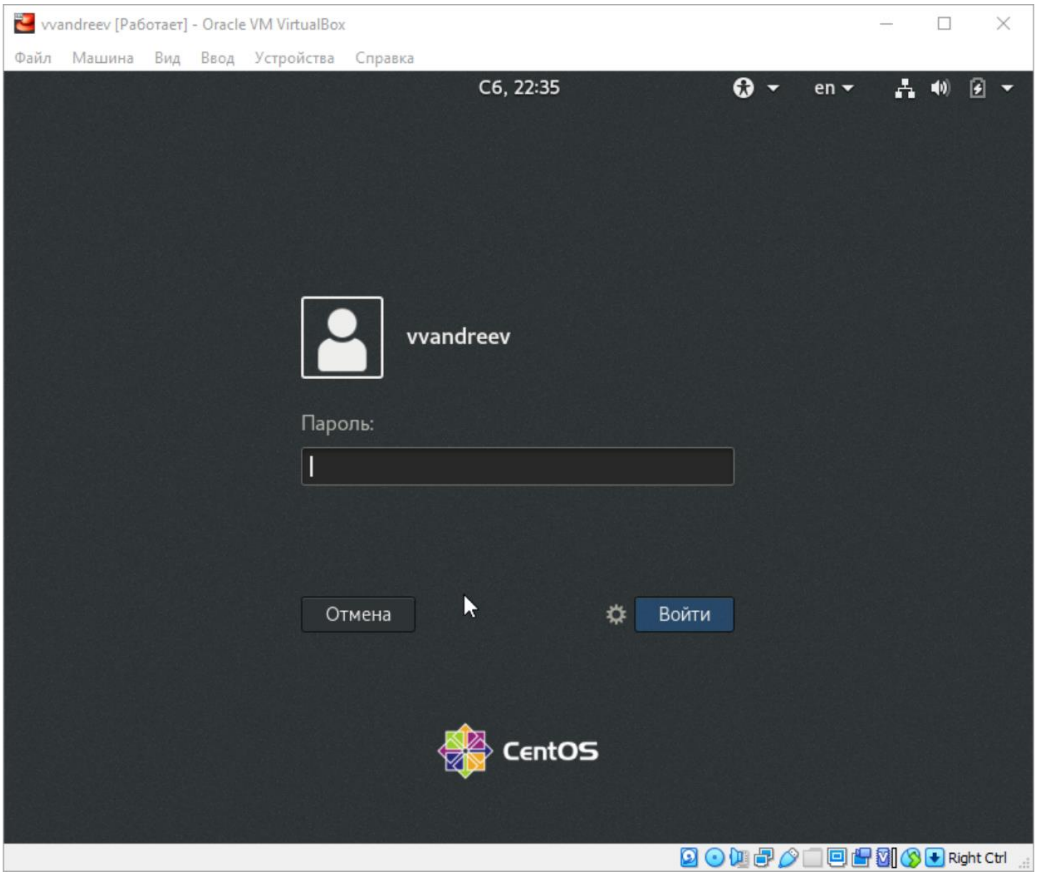


(Рисунок 16)

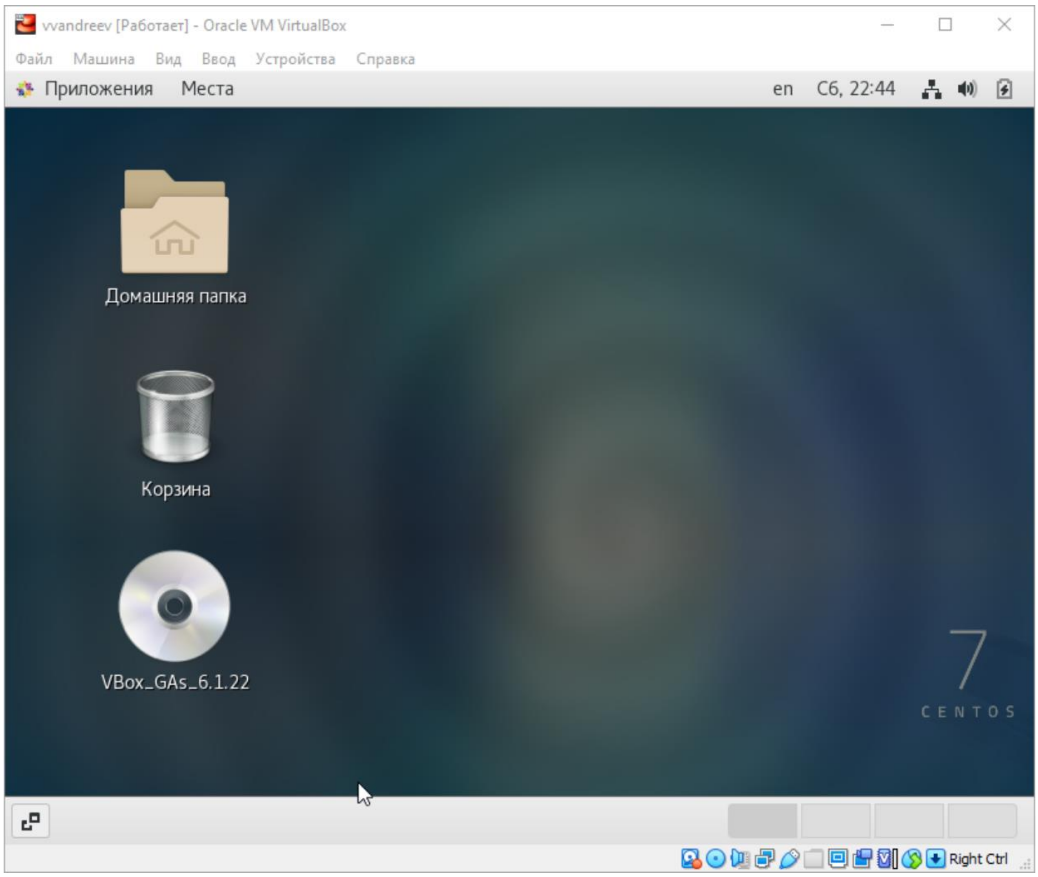
Затем настаиваем систему выбираем все базовые настройки:

- Выбираем язык
- Диск куда установить систему
- Какие файлы установить
- Путь к сетевому домену
- Задаем пароль для root
- Создаем админа
- Перезагружаем
- Заходим в систему

Загружаем

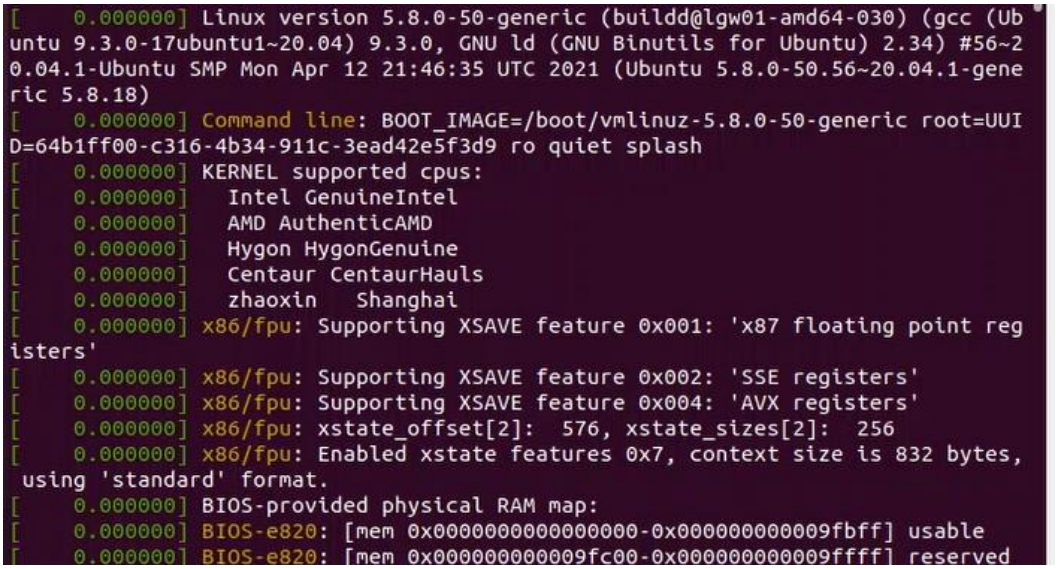


Затем подключаем образ диска дополнительной, гостевой ОС: кликаем на устройства, установить гостевую ОС



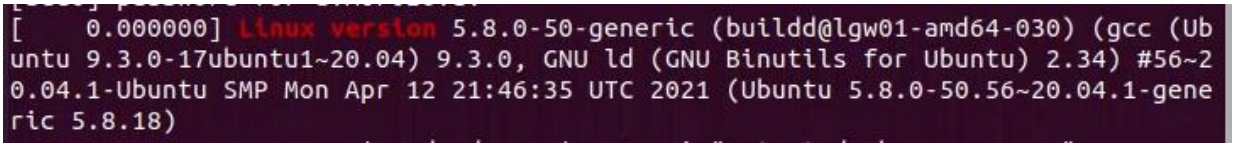
3)Домашнее задание:

Загружаем графическое окружение и открываем консоль. Анализируем последовательность загрузки системы

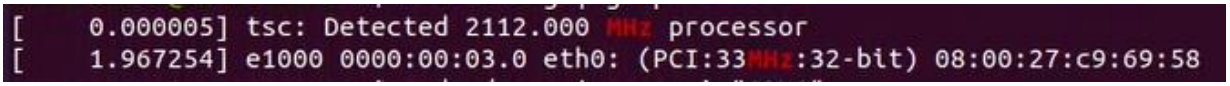


Далее используем команду «dmesg | grep -i "то, что ищем"», чтобы найти необходимую информацию:

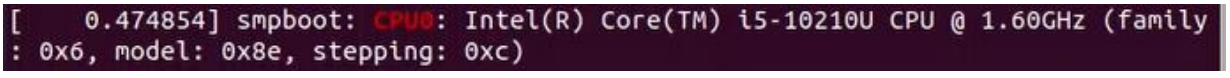
1) Версия ядра Linux: команда «dmesg | grep -i "Linux version"». В данном случае версия операционной системы – 5.8.0-50-generic



2) Частота процессора: команда «dmesg | grep –i "MHz"». Частота процессора составляет 2112 МГц



3) Модель процессора: команда «dmesg | grep –i "CPU0"». Из рисунка видно, что модель моего процессора – Intel(R) Core(TM) i5- 10210U CPU @ 1.60GHz



4) Объем доступной оперативной памяти: команда «dmesg | grep –i "Memory"». Из рисунка видно, что объем доступной оперативной памяти составляет 4193848 Кбайт ОЗУ.

```
[    0.092850] check: Scanning 1 areas for low memory corruption
[    0.096803] Early memory node ranges
[    0.206576] PM: hibernation: Registered nosave memory: [mem 0x00000000-0x000
00fff]
[    0.206578] PM: hibernation: Registered nosave memory: [mem 0x0009f000-0x000
9ffff]
[    0.206578] PM: hibernation: Registered nosave memory: [mem 0x000a0000-0x000
effff]
[    0.206578] PM: hibernation: Registered nosave memory: [mem 0x000f0000-0x000
fffff]
[    0.206579] PM: hibernation: Registered nosave memory: [mem 0xdfff0000-0xdff
fffff]
[    0.295165] Memory: 3971216K/4193848K available (14339K kernel code, 2537K
wdata, 5452K rodata, 2644K init, 4916K bss, 222632K reserved, 0K cma-reserved)
[    0.363789] Freeing SMP alternatives memory: 40K
[    0.477199] x86/mm: Memory block size: 128MB
```

5) Тип обнаруженного гипервизора: команда «dmesg | grep –i "Hypervisor detected"». Тип данного гипервизора – KVM

```
[    0.000000] Hypervisor detected: KVM
```

6-7) Тип файловой системы корневого раздела и последовательность монтирования файловых систем: команда «dmesg | grep –i "Mount"». Тип файловой системы корневого раздела – EXT4.

```
[    0.355848] Mount-cache hash table entries: 8192 (order: 4, 65536 bytes, lin
ear)
[    0.355856] Mountpoint-cache hash table entries: 8192 (order: 4, 65536 bytes
, linear)
[    2.568298] EXT4-fs (sda5): mounted filesystem with ordered data mode. Opts:
 (null)
[    3.354457] systemd[1]: Set up automount Arbitrary Executable File Formats F
ile System Automount Point.
[    3.356646] systemd[1]: Mounting Huge Pages File System...
[    3.357681] systemd[1]: Mounting POSIX Message Queue File System...
[    3.358812] systemd[1]: Mounting Kernel Debug File System...
```

1) Учетная запись пользователя – это необходимая для системы информация о пользователе, хранящаяся в специальных файлах. Информация используется Linux для аутентификации пользователя и назначения ему прав доступа. Аутентификация – системная процедура, позволяющая Linux определить, какой именно пользователь осуществляет вход. Вся информация о пользователе обычно хранится в файлах /etc/passwd и /etc/group. 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 [команда]. Например, команда «man ls» выведет справку о команде «ls».

• Для перемещения по файловой системе: cd [путь]. Например, команда «cd newdir» осуществляет переход в каталог newdir

• Для просмотра содержимого каталога: ls [опции] [путь]. Например, команда «ls –a ~/newdir» отобразит имена скрытых файлов в каталоге newdir • Для определения объёма каталога: du [опция] [путь]. Например, команда «du –k ~/newdir» выведет размер каталога newdir в килобайтах

• Для создания / удаления каталогов / файлов: mkdir [опции] [путь] / rmdir [опции] [путь] / rm [опции] [путь]. 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 [опции]. Например, команда «history 5» покажет список последних 5 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*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*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 рамер тома 32 пэбибайта (32*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 (Stephen Tweedie) 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*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.

• ext4 – попытка создать 64-х битную ext3 способную поддерживать больший 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 пригодится 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 Дерева», 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 посредствам 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 дамп 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 pkill – 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 выполнения работы я на практике приобрел навыки установки операционной системы на виртуальную машину, настройки необходимых для дальнейшей работы сервисов, поиска информации об установленной ОС и ее комплектующих, используя консоль