

**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:** Аветисян Давид

**Группа:** НПМбд-01-20

**Ст. билет №:** 1032201709

Москва

2021 г.

## Цель работы:

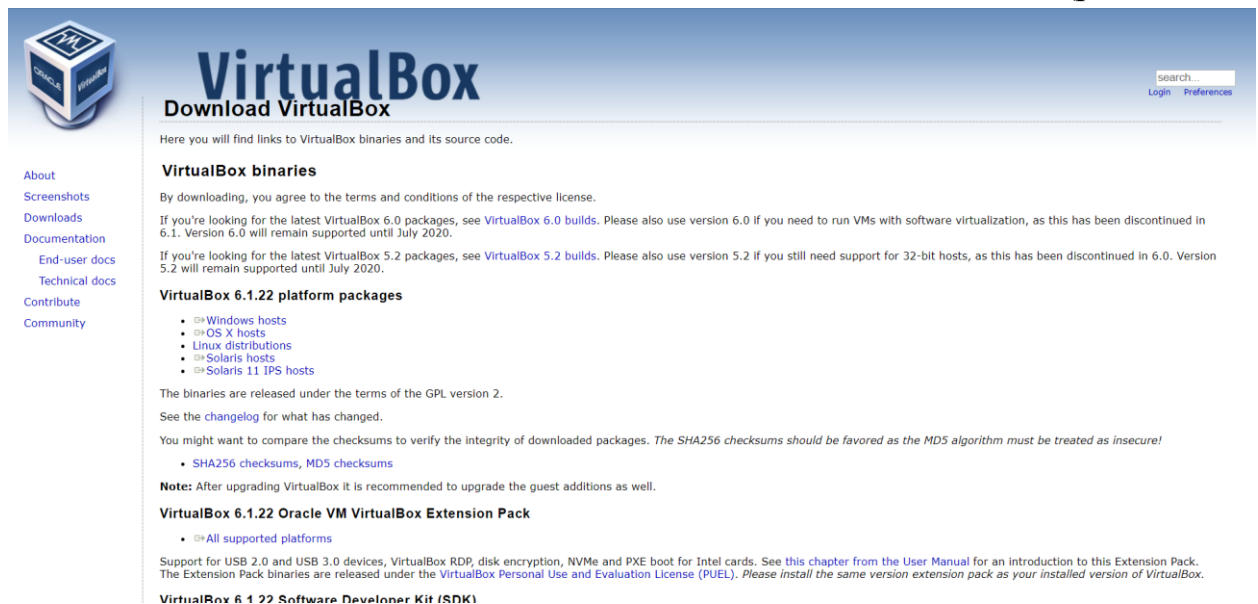
Целью данной работы является приобретение практических навыков установки операционной системы на виртуальную машину, настройки минимально необходимых для дальнейшей работы сервисов.

## Ход работы:

Скачиваем и устанавливаем VirtualBox, которая необходима для запуска виртуальных машин (рис. 1, рис. 2) (скачать можно на сайте <https://www.virtualbox.org>).



(рис. 1)



(рис. 2)

Устанавливаем скачанный файл (рис. 3)

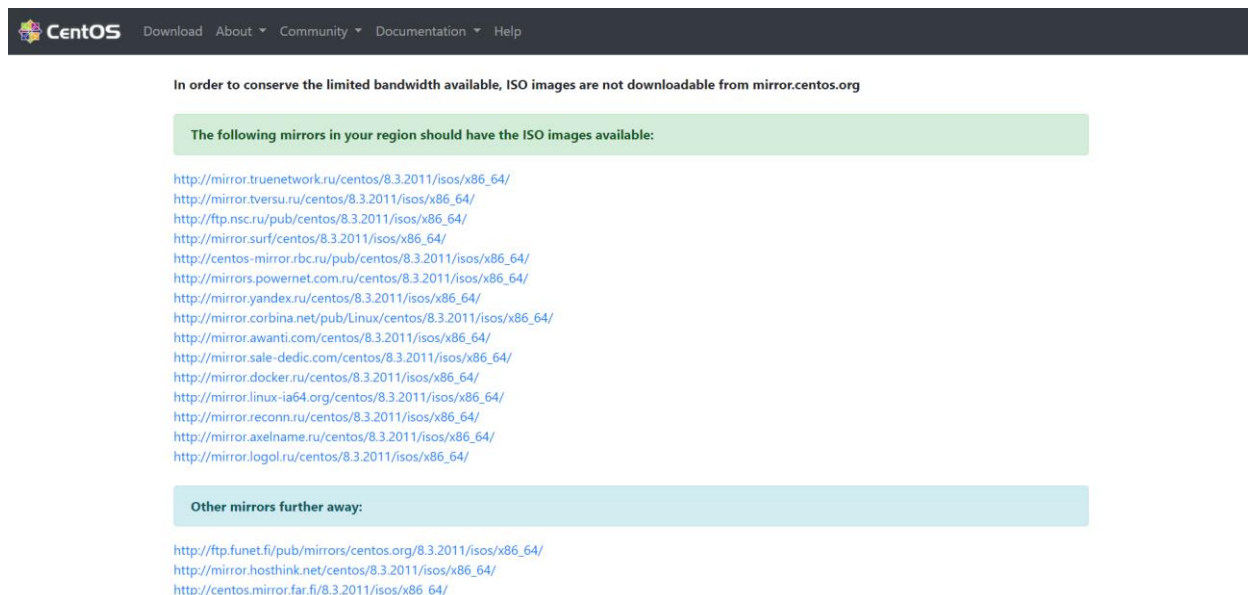


(рис. 3)

Также скачиваем дистрибутив Linux CentOS-8 (рис. 4, рис. 5) (можно скачать на сайте <https://wiki.centos.org>).

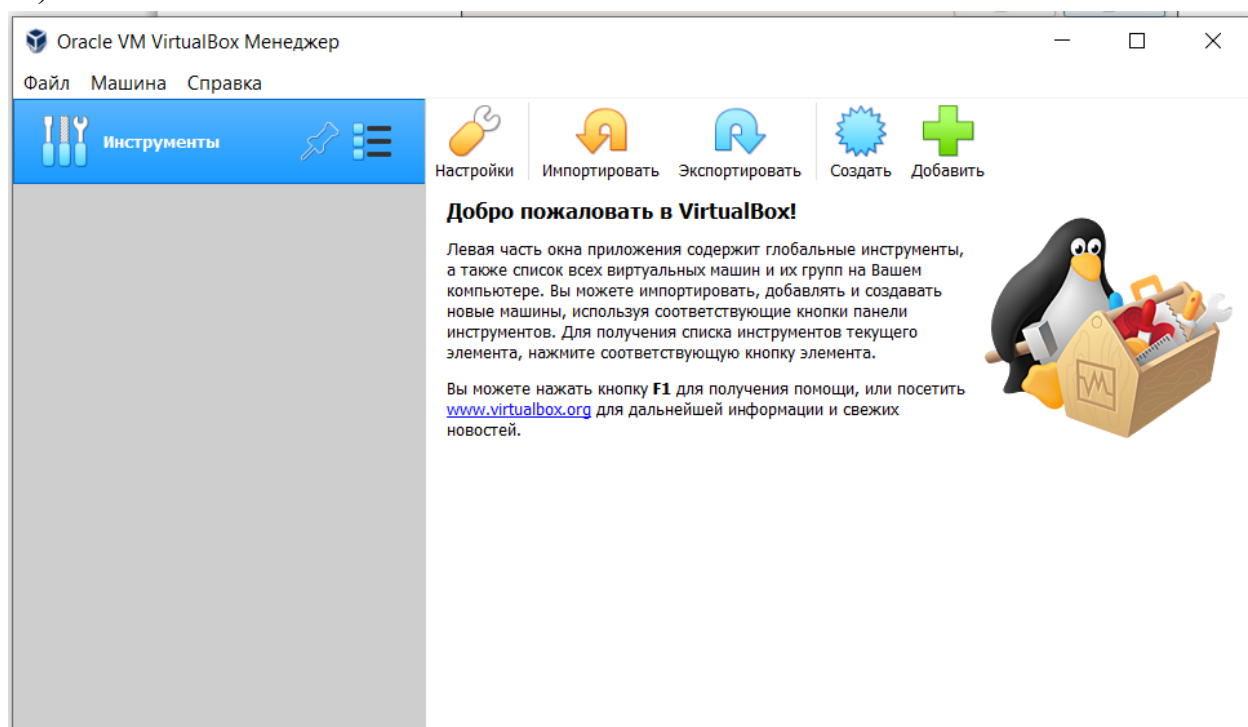
CentOS Linux Version	Minor release	CD and DVD ISO Images	Packages	Release Email	Release Notes	End-Of-Life
8-Stream	N/A	DVD and Netinstall images (including checksums) are available on <a href="#">mirrors</a>	<a href="#">RPMs</a>	<a href="#">CentOS Stream</a>	<a href="#">CentOS Stream</a>	N/A
8	3 (2011)	DVD and Netinstall x86_64 images are available on <a href="#">mirrors</a> ( <a href="#">checksums</a> ).	<a href="#">RPMs</a>	<a href="#">CentOS</a>	<a href="#">CentOS RHEL</a>	31 December 2021**

(рис. 4)



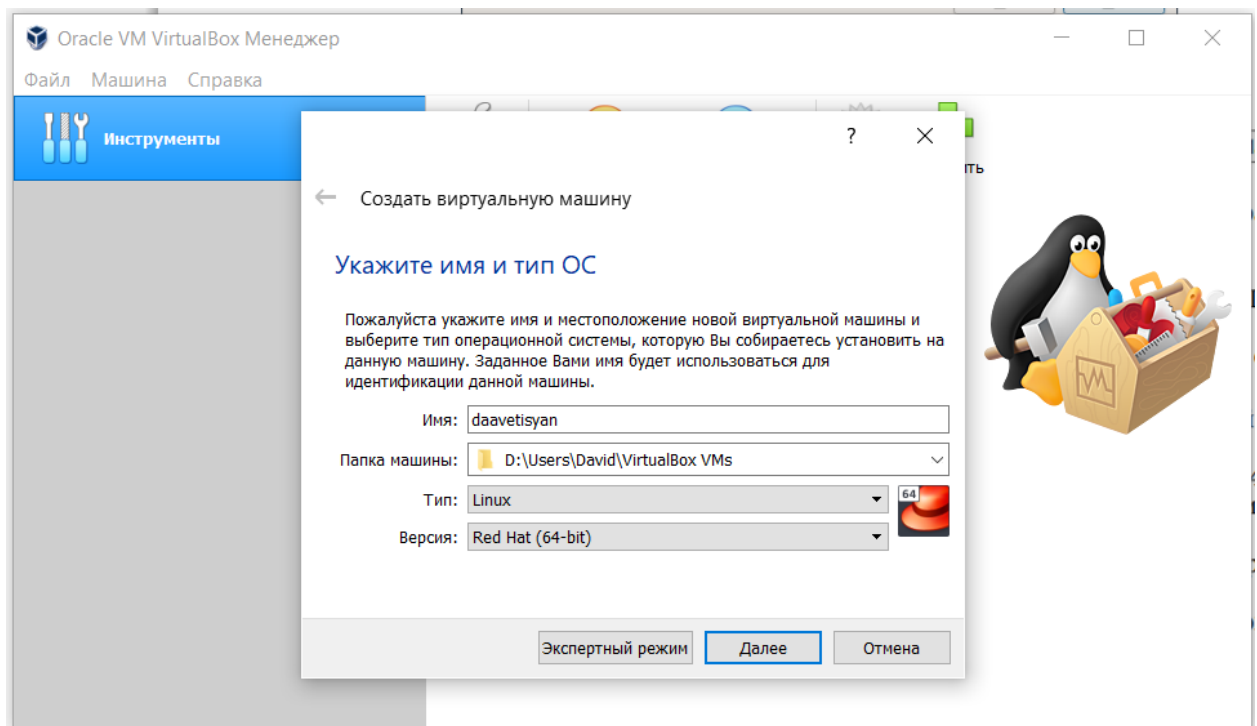
(рис. 5)

Запускаем виртуальную машину и проверяем месторасположения каталога для виртуальных машин. Затем переходим к созданию новой виртуальной машины. Для этого в VirtualBox мы выбираем Машина – Создать (рис. 6).



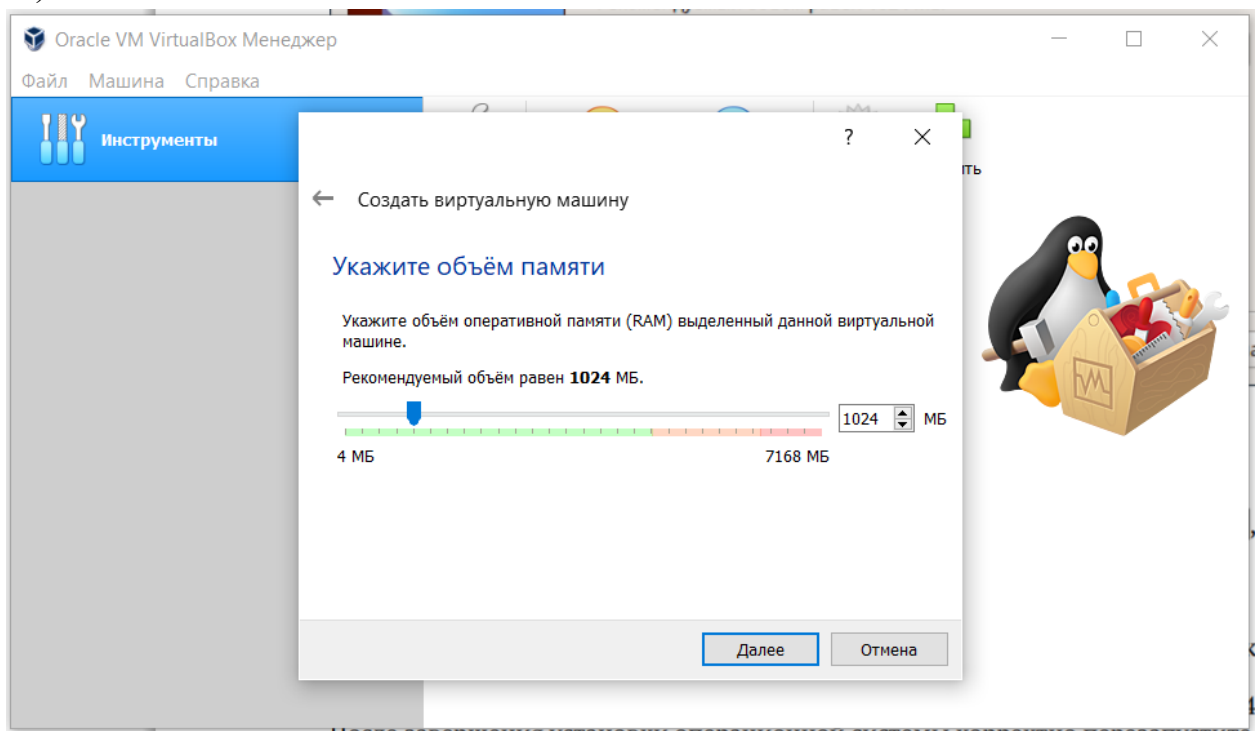
(рис 6)

Указываем имя виртуальной машины, тип операционной системы - Linux, RedHat (рис. 7).



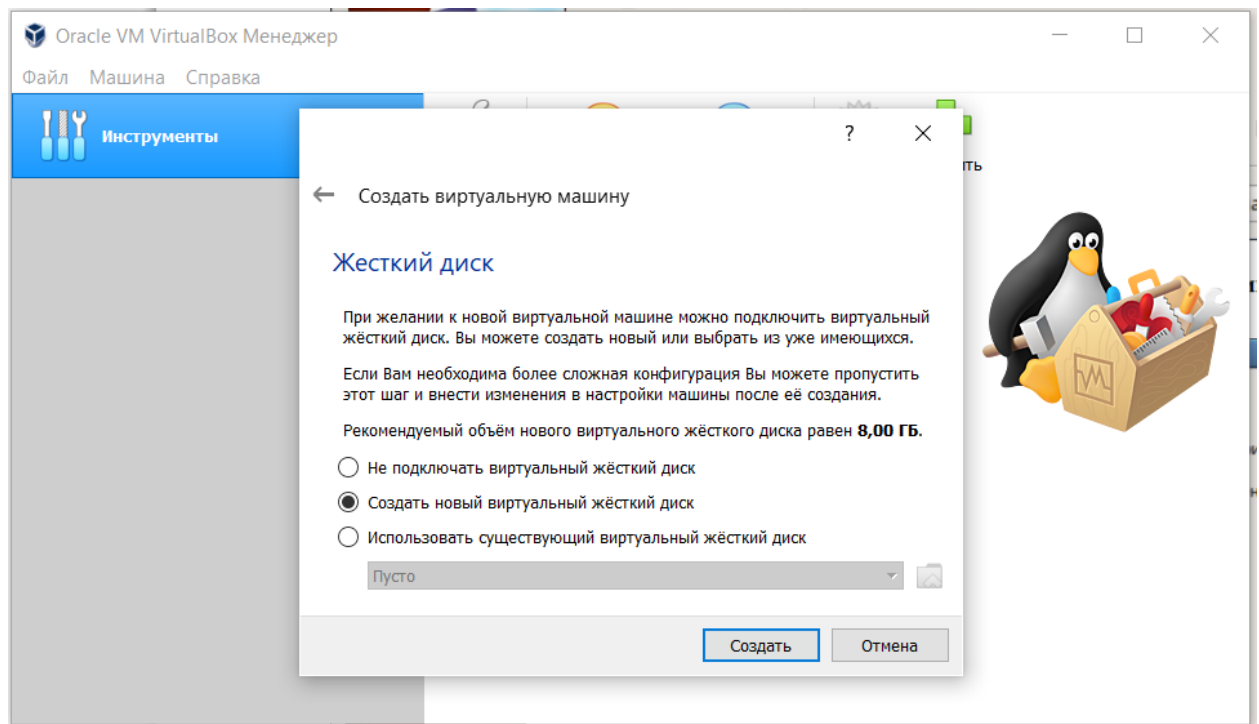
(рис 7)

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

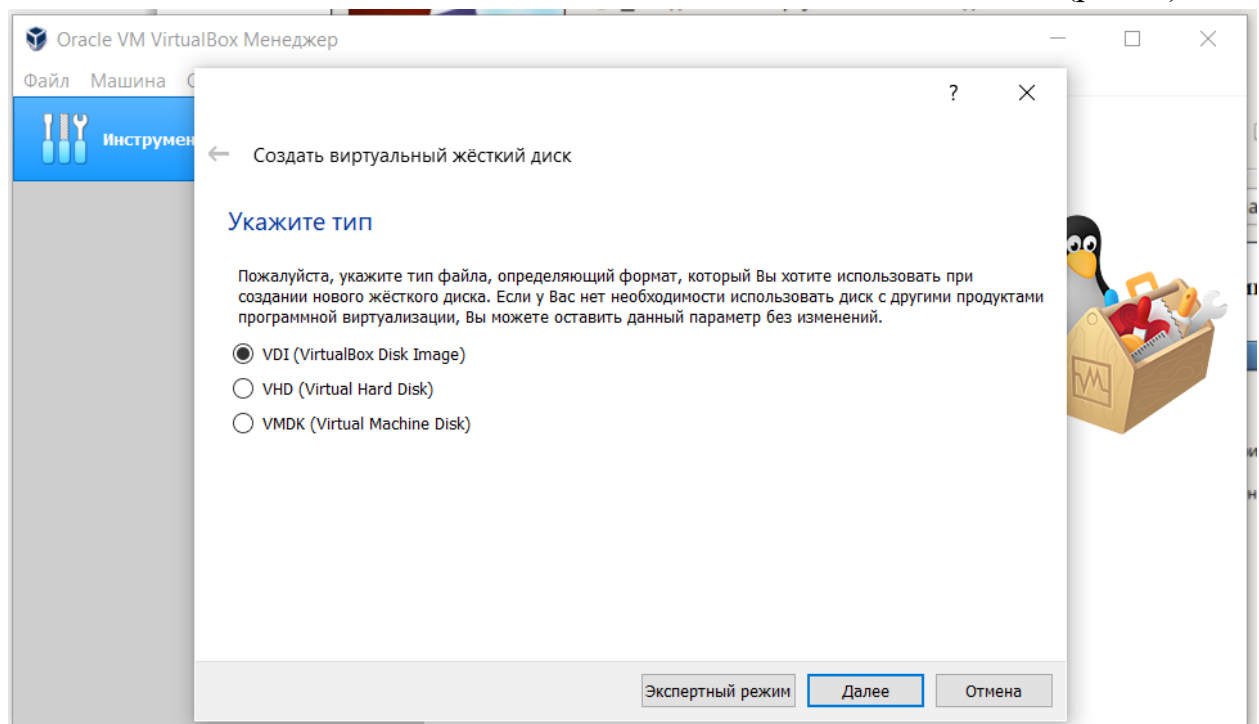


(рис 8)

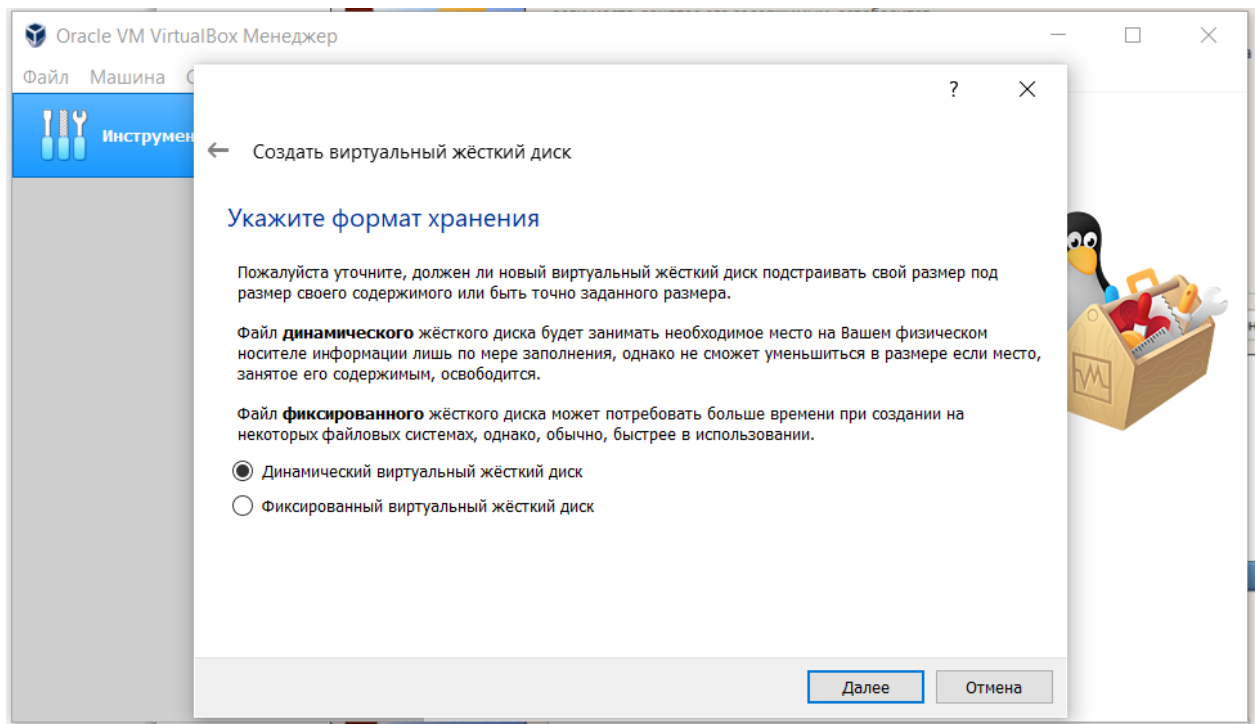
Задаем конфигурацию жесткого диска VDI, динамический виртуальный жесткий диск (рис. 9, рис. 10). Затем задаем размер диска 20 ГБ (но рекомендовано 40 или больше) и его расположение (рис. 11).



(рис. 9)

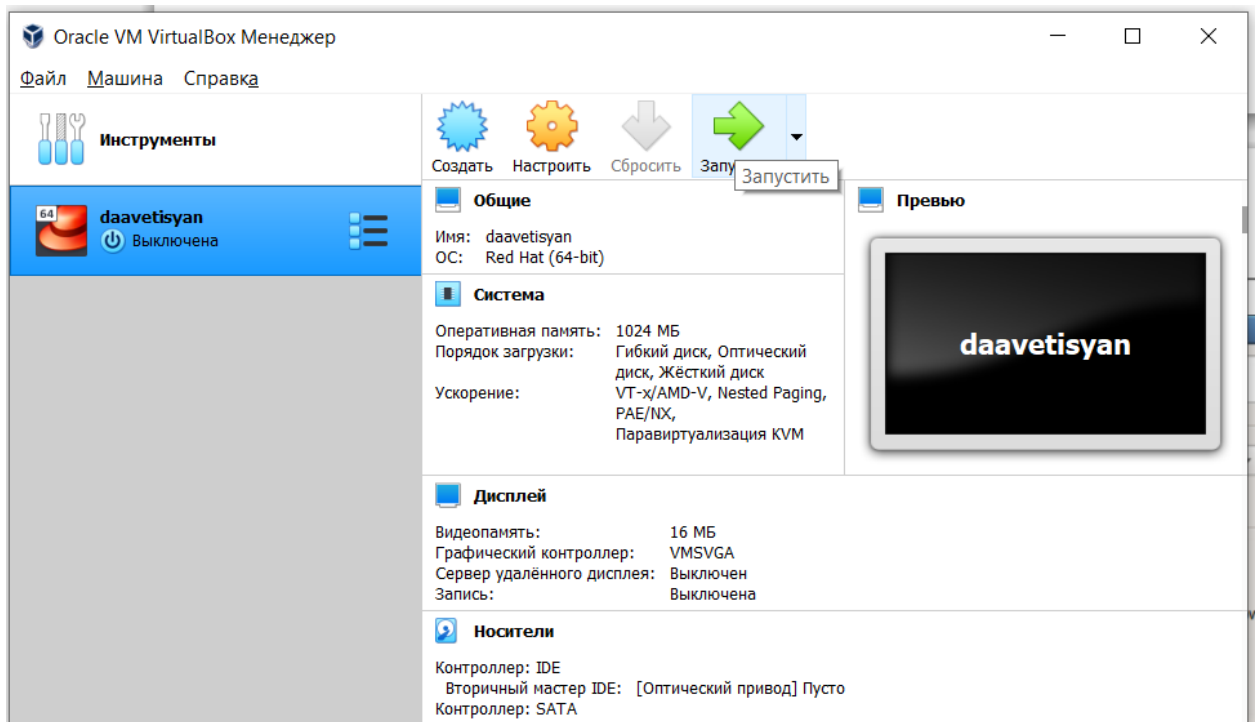


(рис. 10)

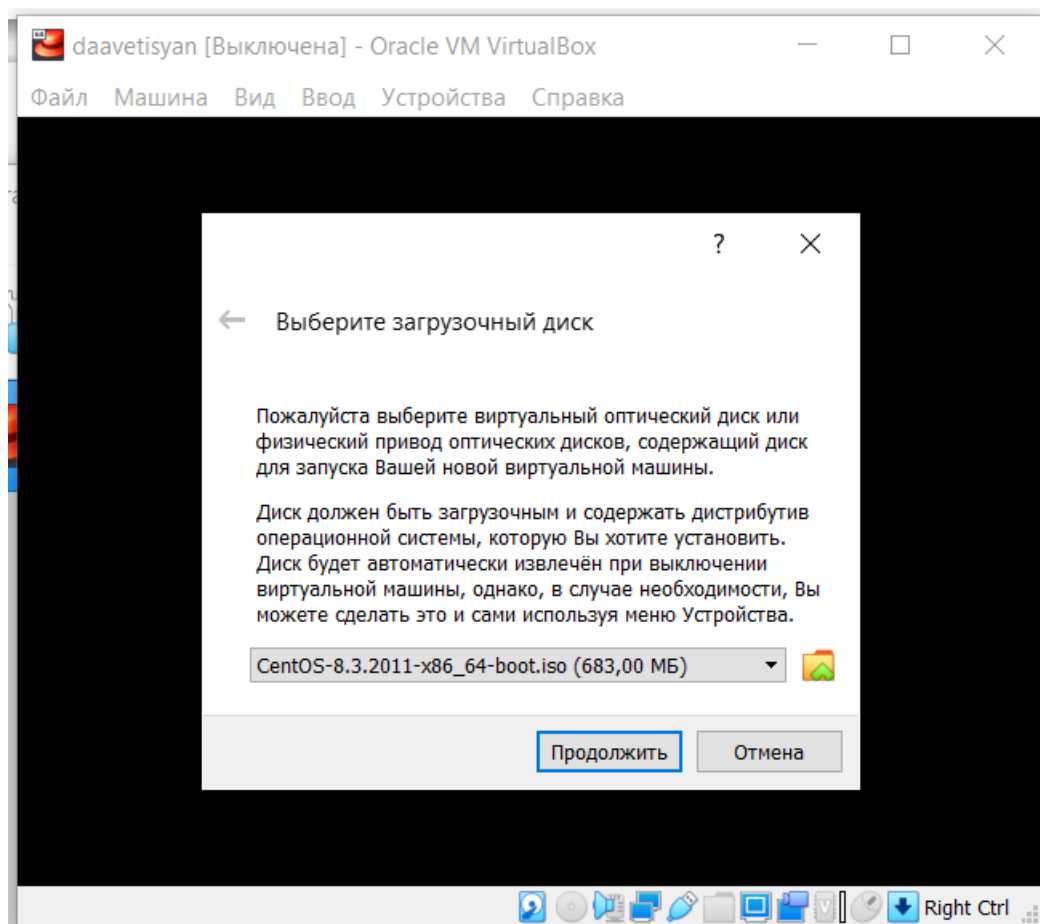


(рис. 11)

Запускаем виртуальную машину (рис. 12). Заходим в Свойства - Носители в виртуальной машине и добавляем новый привод оптических дисков. Выбираем образ, который мы ранее скачали на наш компьютер (рис. 13).



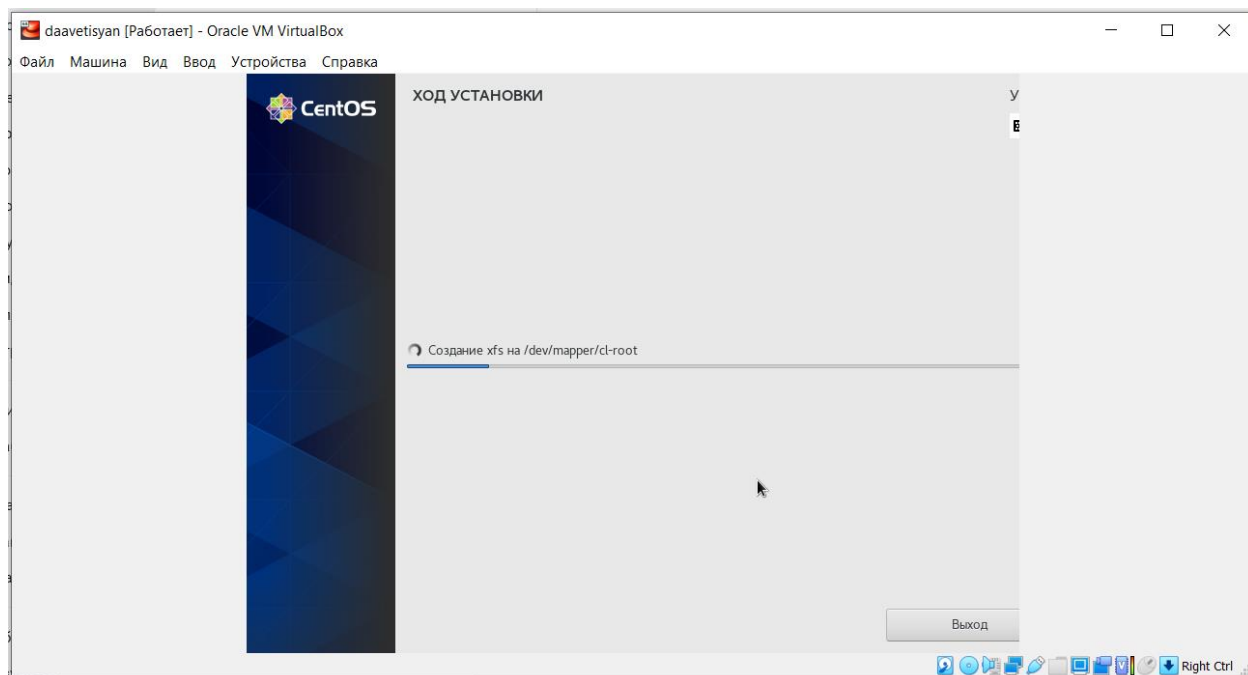
(рис. 12)



(рис. 13)

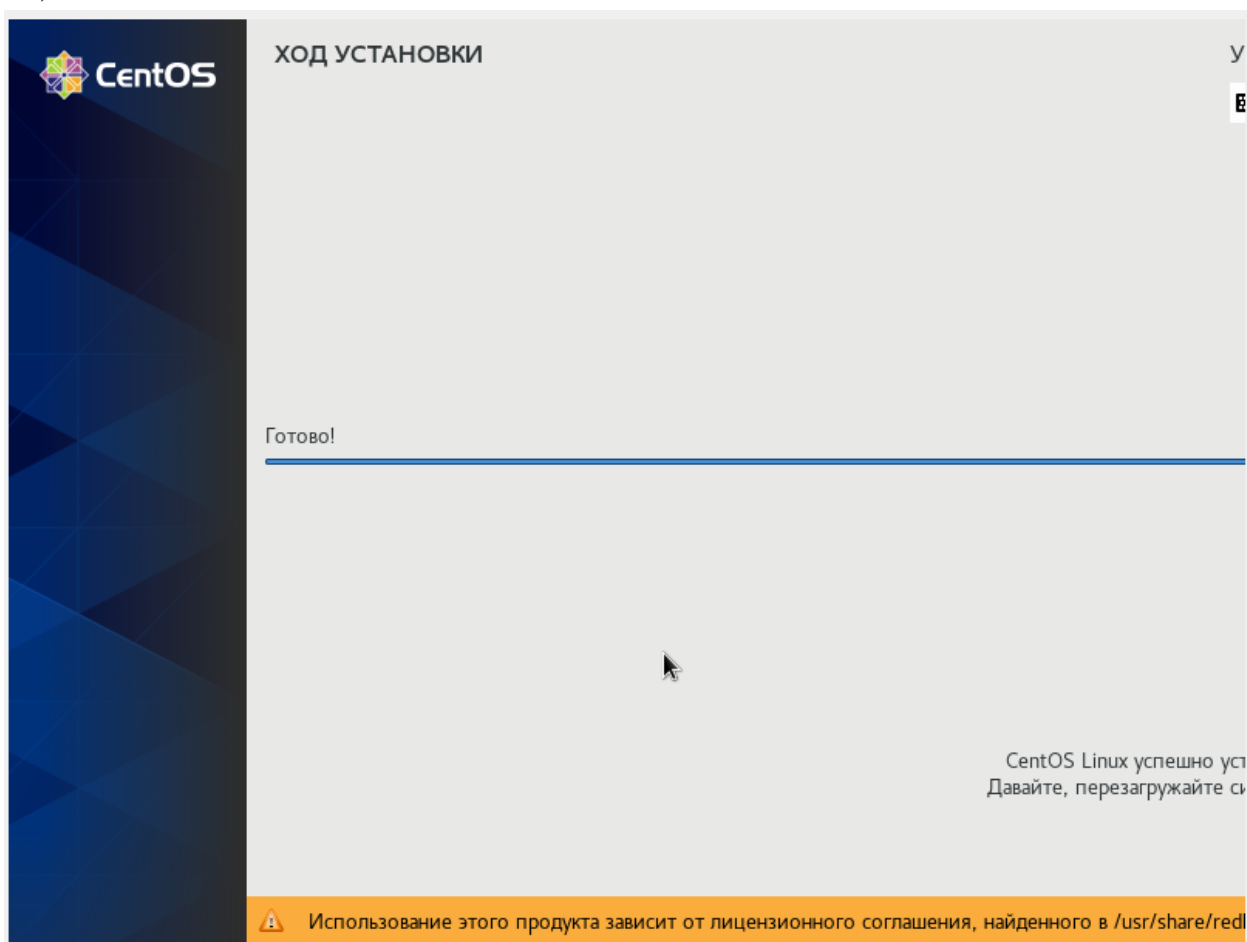
Продолжаем и устанавливаем язык интерфейса. В разделе выбора программ указываем в качестве базового окружения Сервер с GUI, а в качестве дополнения – средства разработки. Место установки оставляем без изменений. Включаем сетевое соединение и в качестве имени узла указываем daavetisyan.localdomain. Нажимаем начать установку и устанавливаем пароль для root и пользователя с правами администратора (рис. 14).



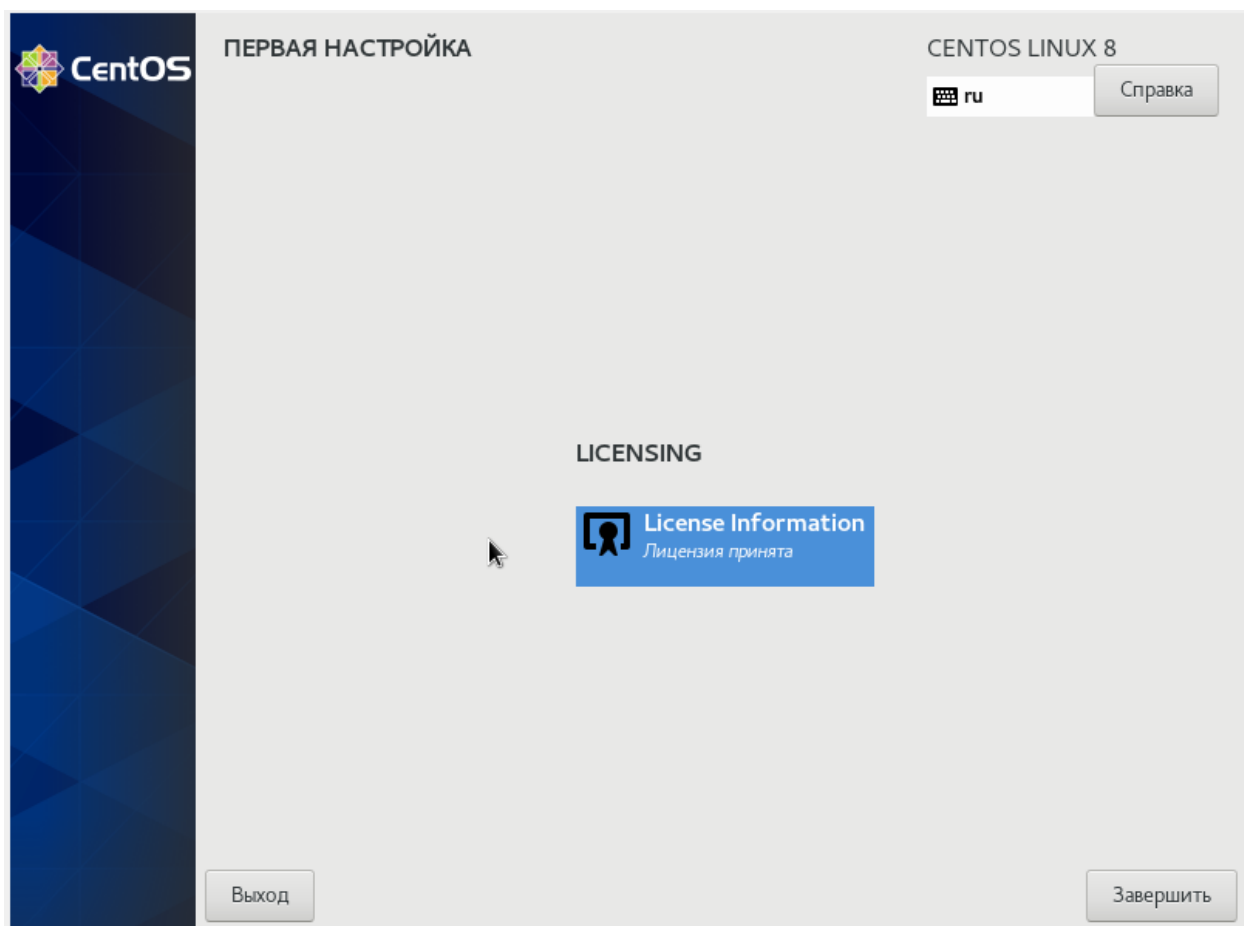


(рис. 14)

После завершения установки операционной системы корректно перезагружаем виртуальную машину и принимаем условия лицензии (рис. 15, рис. 16).

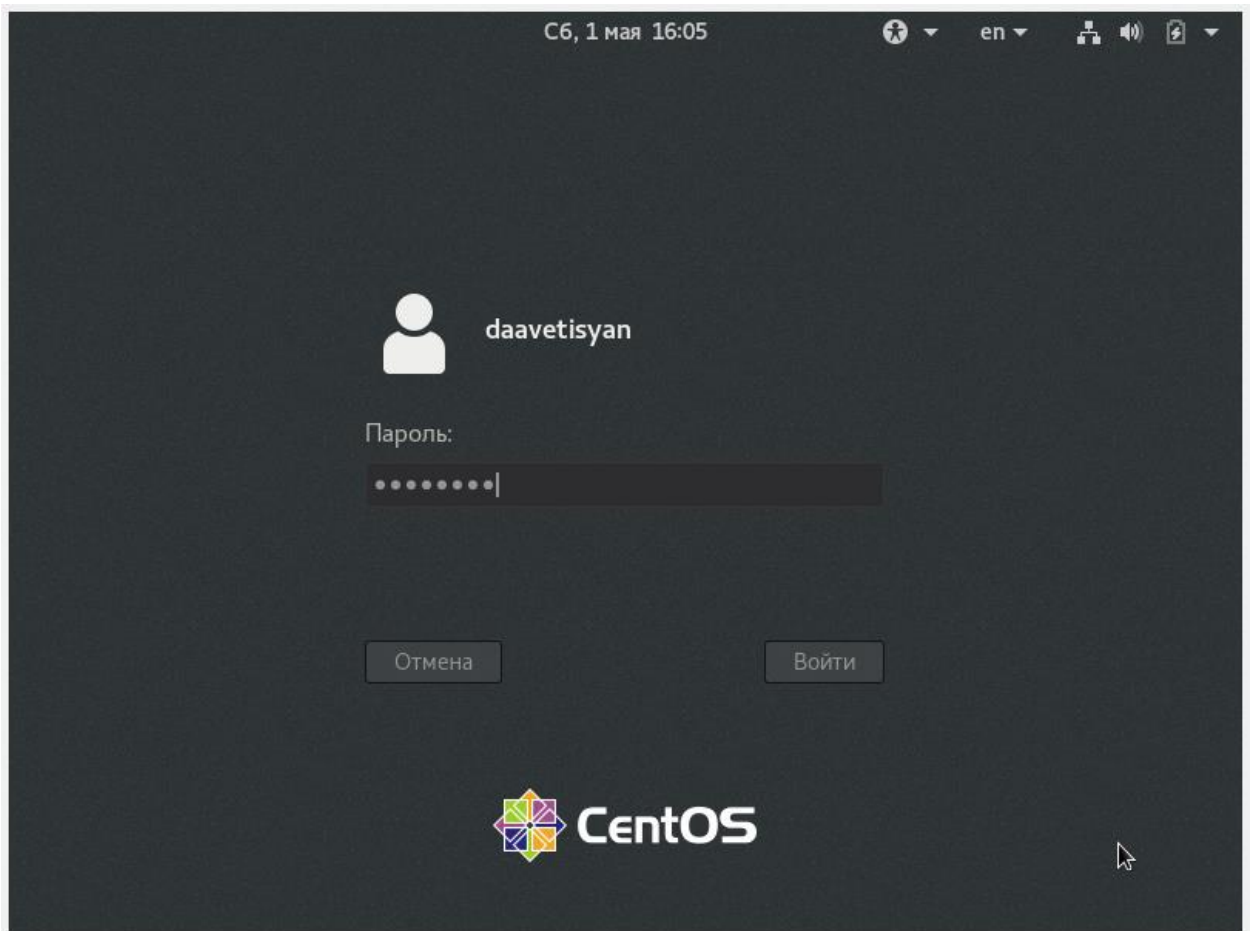


(рис. 15)

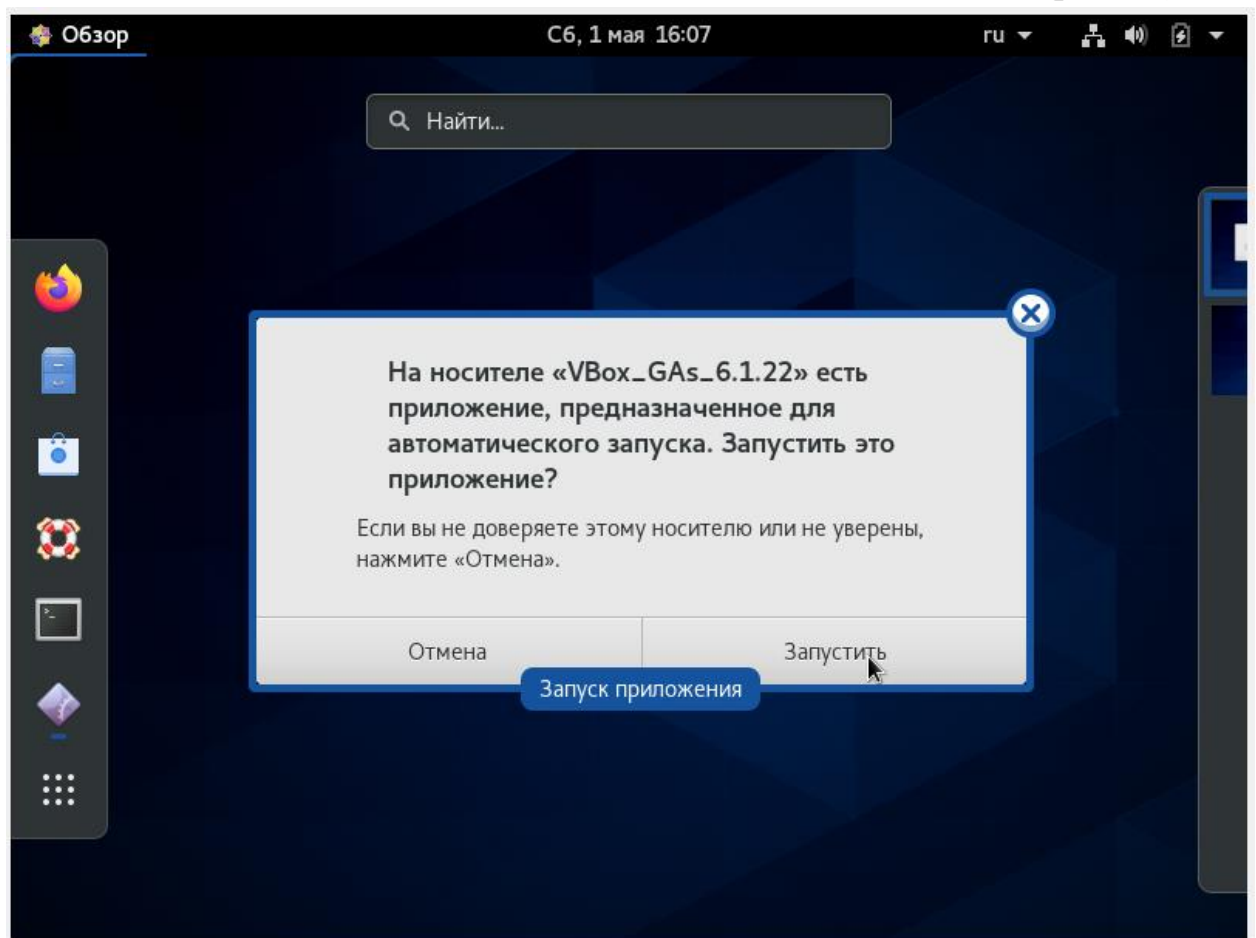


(рис. 16)

Входим под заданной при установке учетной записью (рис. 17). В меню устройства виртуальной машины подключаем образ диска дополнительной гостевой ОС (рис. 18).

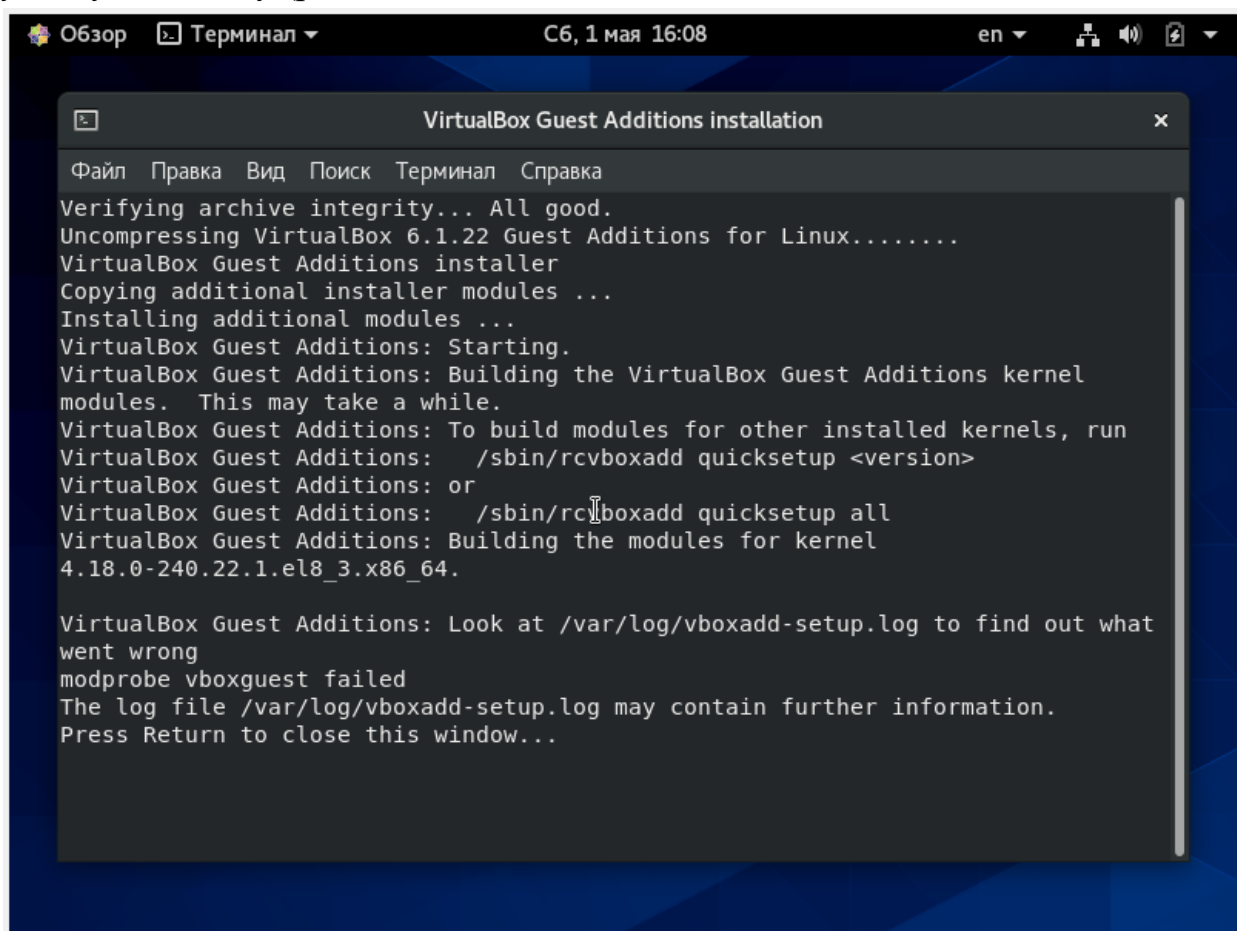


(рис. 17)



(рис. 18)

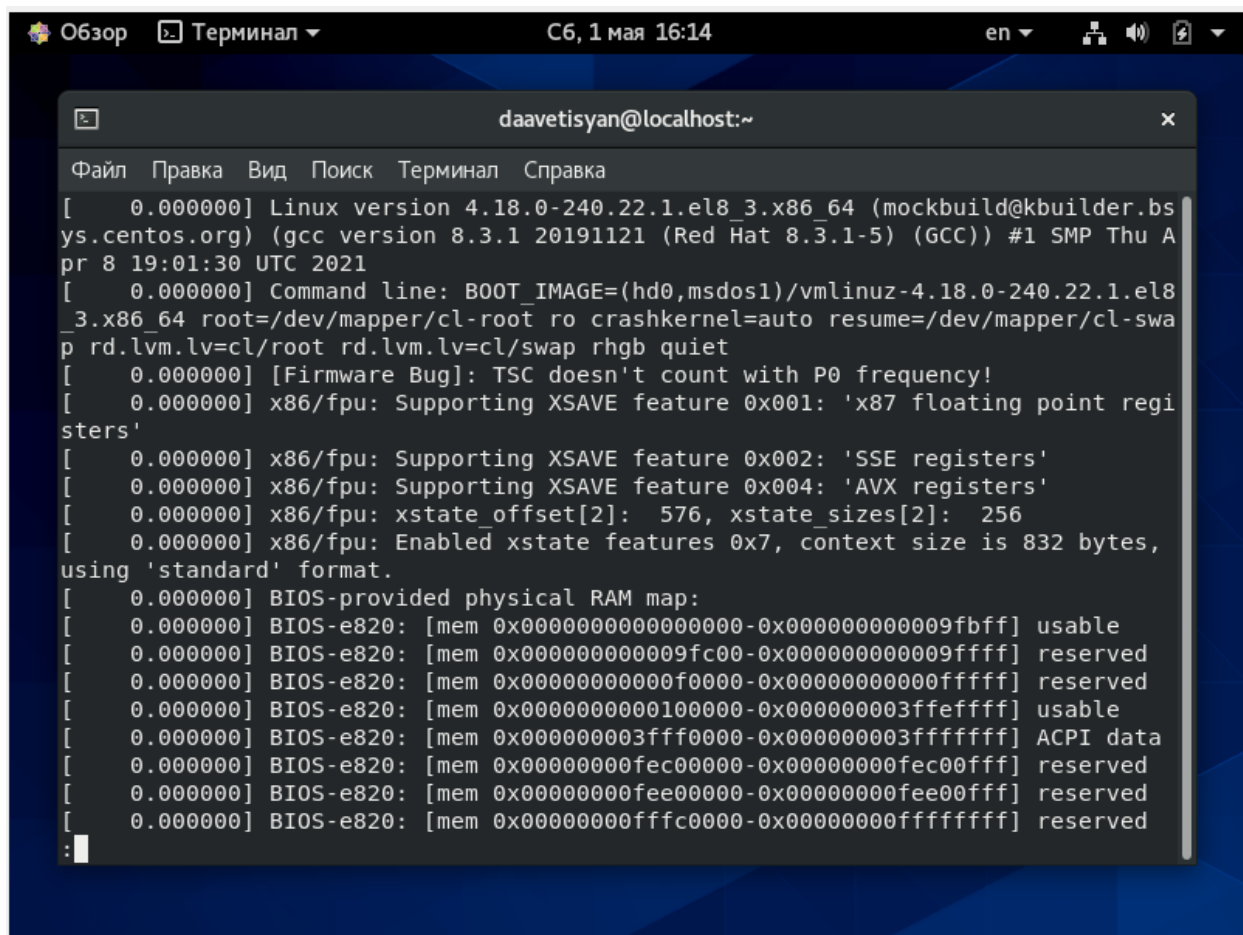
После загрузки дополнений нажимаем Enter и корректно перезагружаем виртуальную машину (рис. 19)



(рис. 19)

### Домашняя работа:

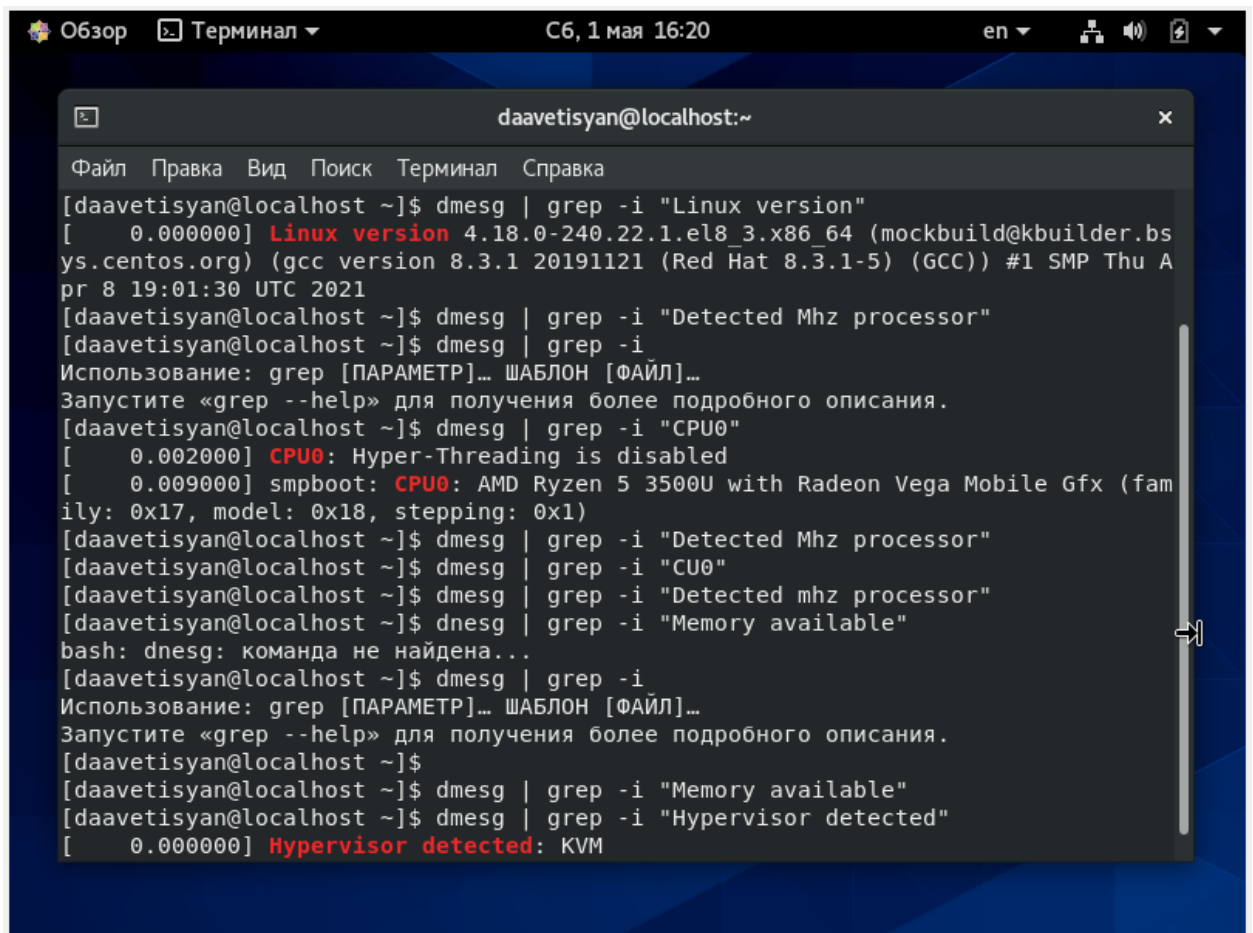
Загружаем графическое окружения и открываем консоль. Анализируем последовательность загрузки системы, используя команду «dmesg» и введя пароль. Просмотрим вывод этой команды, выполнив команду «dmesg | less» (рис. 20).



(рис. 20)

Используем команду «`dmesg | grep -i "то, что ищем"`», чтобы найти необходимую информацию (рис. 21):

- 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"`».



```
Обзор Терминал C6, 1 мая 16:20 en
daavetisyan@localhost:~
Файл Правка Вид Поиск Терминал Справка
[daavetisyan@localhost ~]$ dmesg | grep -i "Linux version"
[    0.000000] Linux version 4.18.0-240.22.1.el8_3.x86_64 (mockbuild@kbuilder.bsys.centos.org) (gcc version 8.3.1 20191121 (Red Hat 8.3.1-5) (GCC)) #1 SMP Thu Apr 8 19:01:30 UTC 2021
[daavetisyan@localhost ~]$ dmesg | grep -i "Detected Mhz processor"
[daavetisyan@localhost ~]$ dmesg | grep -i
Использование: grep [ПАРАМЕТР]... ШАБЛОН [ФАЙЛ]...
Запустите «grep --help» для получения более подробного описания.
[daavetisyan@localhost ~]$ dmesg | grep -i "CPU0"
[    0.002000] CPU0: Hyper-Threading is disabled
[    0.009000] smpboot: CPU0: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx (family: 0x17, model: 0x18, stepping: 0x1)
[daavetisyan@localhost ~]$ dmesg | grep -i "Detected Mhz processor"
[daavetisyan@localhost ~]$ dmesg | grep -i "CPU0"
[daavetisyan@localhost ~]$ dmesg | grep -i "Detected mhz processor"
[daavetisyan@localhost ~]$ dmesg | grep -i "Memory available"
bash: dmesg: команда не найдена...
[daavetisyan@localhost ~]$ dmesg | grep -i
Использование: grep [ПАРАМЕТР]... ШАБЛОН [ФАЙЛ]...
Запустите «grep --help» для получения более подробного описания.
[daavetisyan@localhost ~]$
[daavetisyan@localhost ~]$ dmesg | grep -i "Memory available"
[daavetisyan@localhost ~]$ dmesg | grep -i "Hypervisor detected"
[    0.000000] Hypervisor detected: KVM
```

(рис. 21)

### Контрольные вопросы:

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`. 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 ( $8 \cdot 2^{60}$  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 ( $16 \cdot 2^{40}$  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 пэббайта ( $32 \cdot 2^{50}$  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 теббайт ( $4 \cdot 2^{40}$  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 больший размер файловой системы (1 эксбайт). 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 делать 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 `psigrep`. 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 данной лабораторной работы я изучил, как установить операционную систему на виртуальную машину и настроить минимально необходимые для дальнейшей работы сервисы.