Цель работы: приобрести навыки установки операционной системы Linux.

Задачи:

1. Закрепить знания о работе с программой VirtualBox.
2. Создать виртуальную машину исходя из предоставленной информации о минимальных аппаратных требованиях, предлагаемых к установке и изучению операционной системы Linux (ОС).
3. Установить ОС Linux на виртуальный компьютер. Разобрать процесс установки ОС на этапы.

Этапы выполнения:

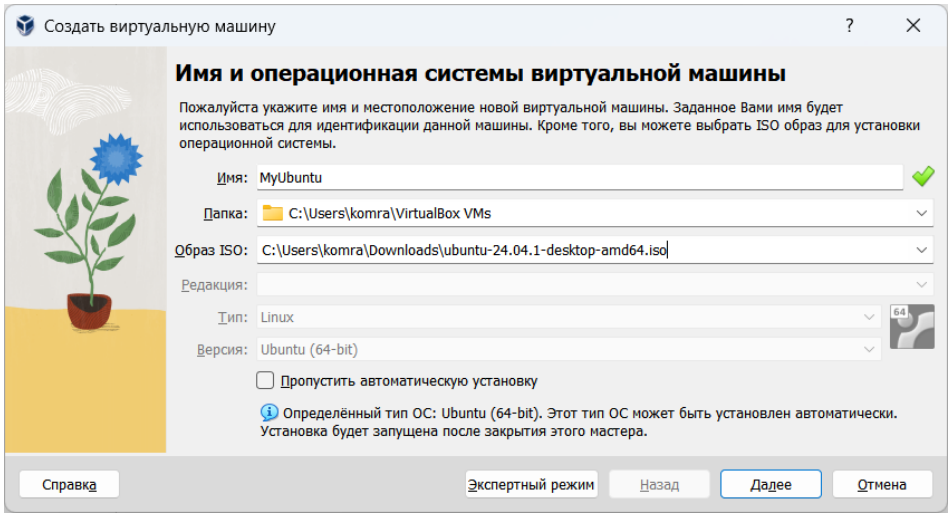

Рисунок 1 Создание операционной системы Ubuntu на виртуальную машину

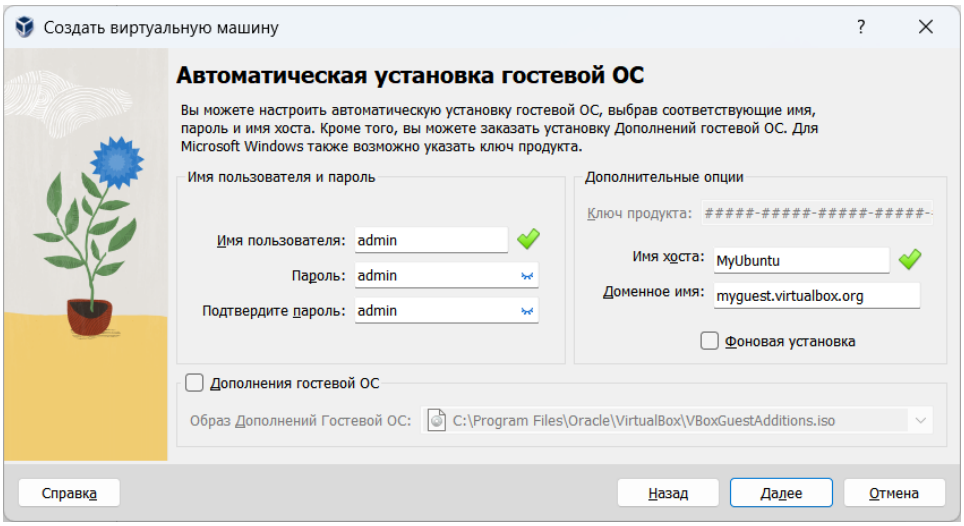

Рисунок 2 Конфигурация операционной системы

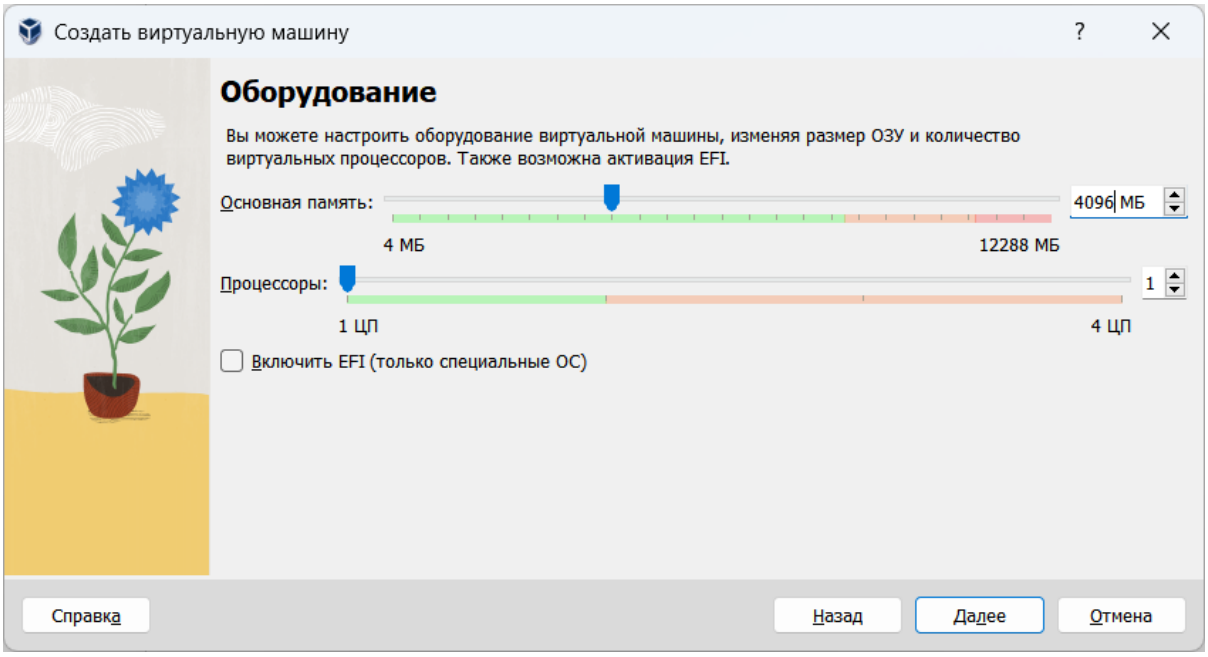

Рисунок 4 Выбор количества ОЗУ и количества ядер процессора для работы ОС

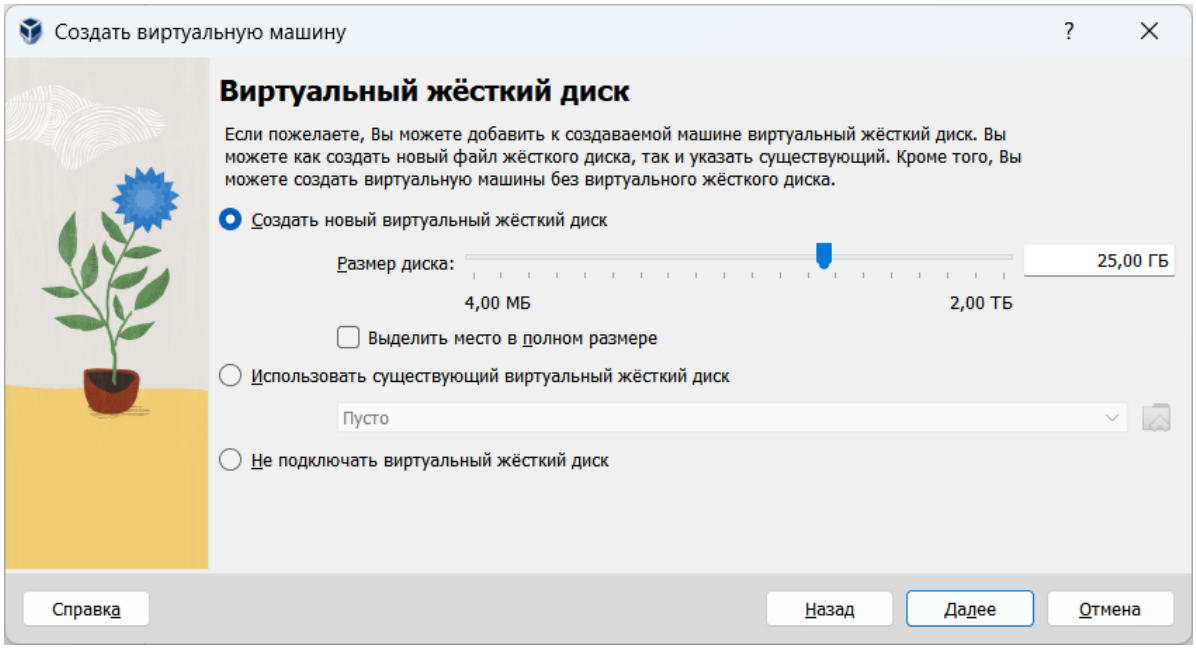

Рисунок 5 Выделение пространства для ОС

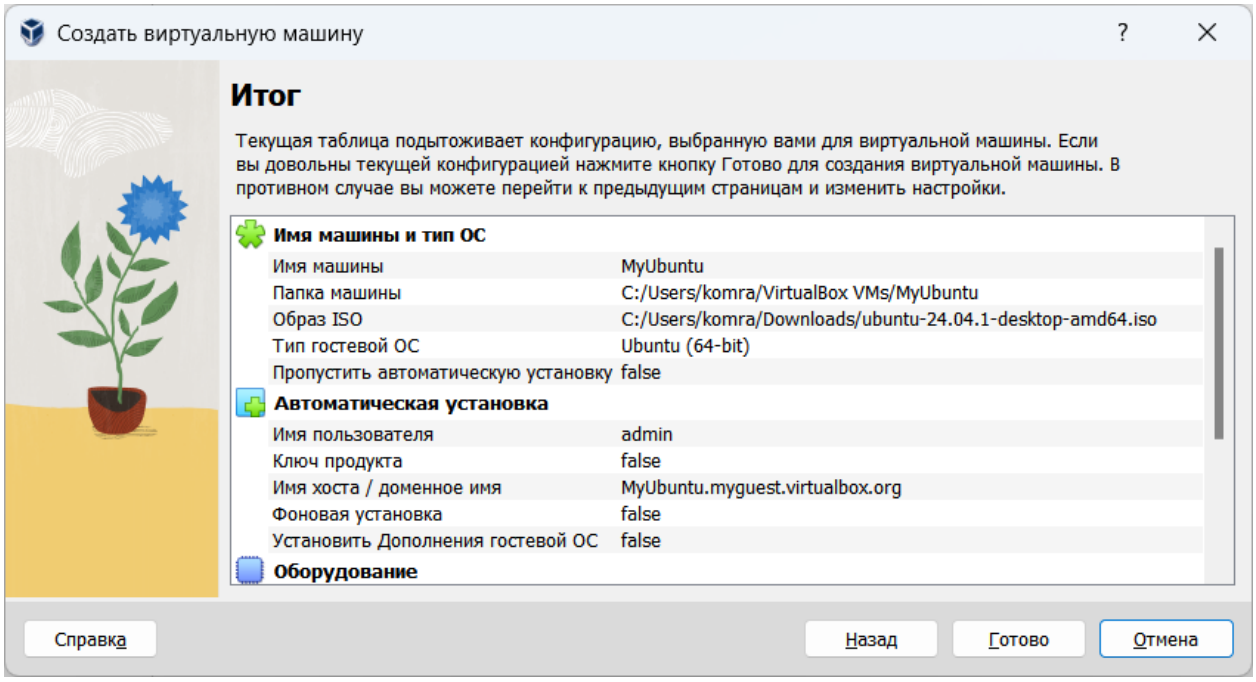

Рисунок 6 Окно перед установкой ОС

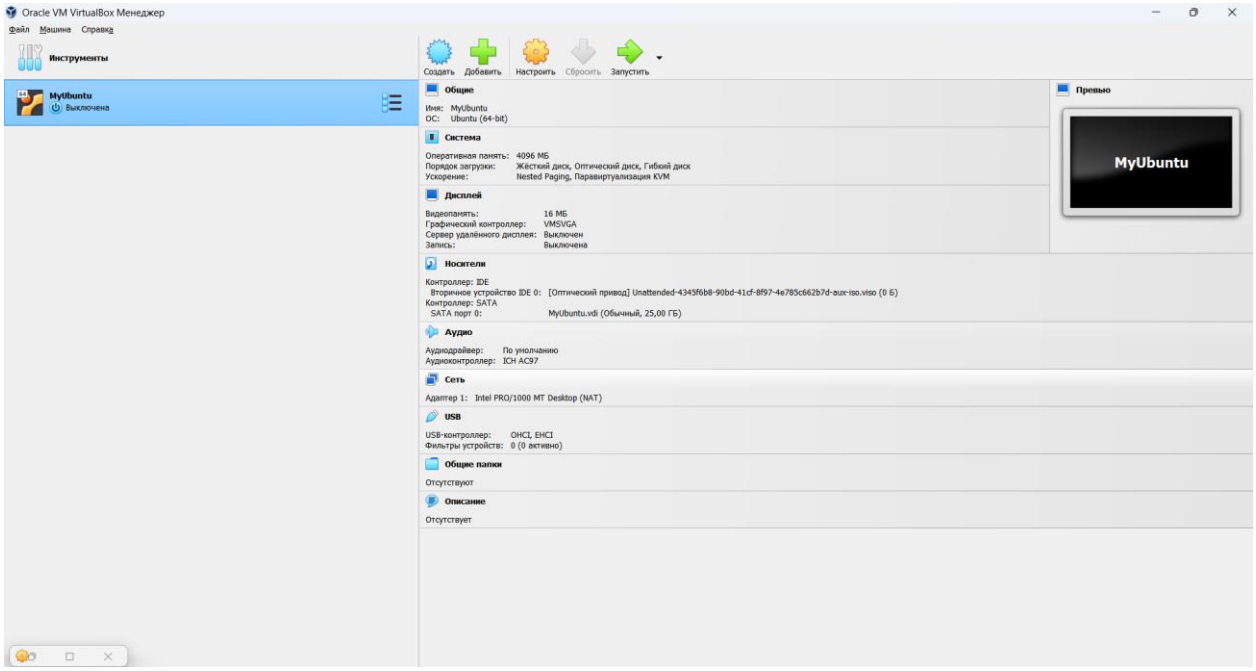

Рисунок 7 Созданная виртуальная машина

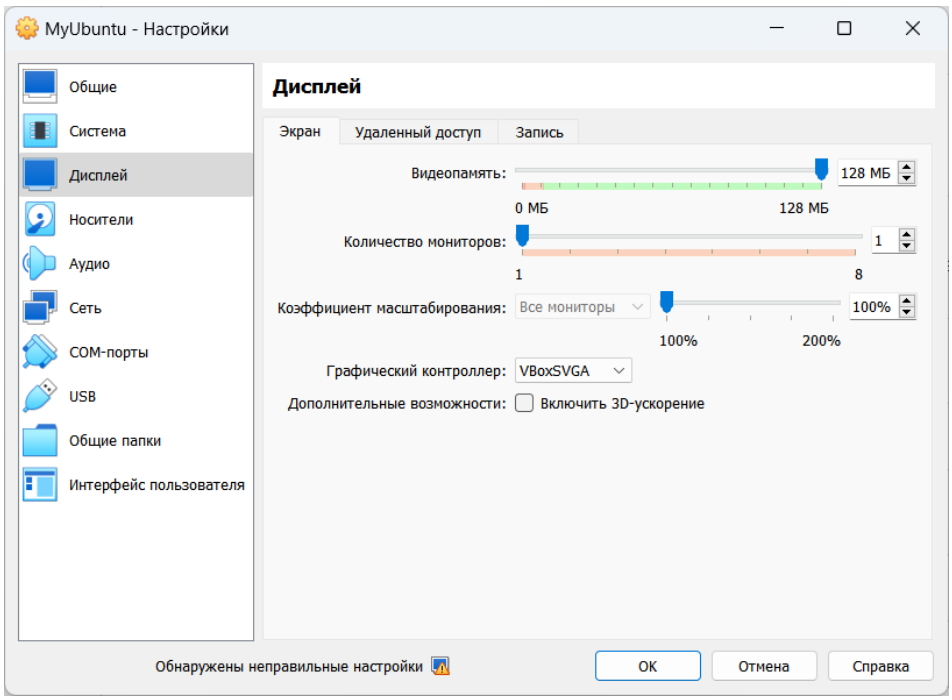

Рисунок 8 Корректировка количества выделяемой видеопамяти и типа графического контроллера

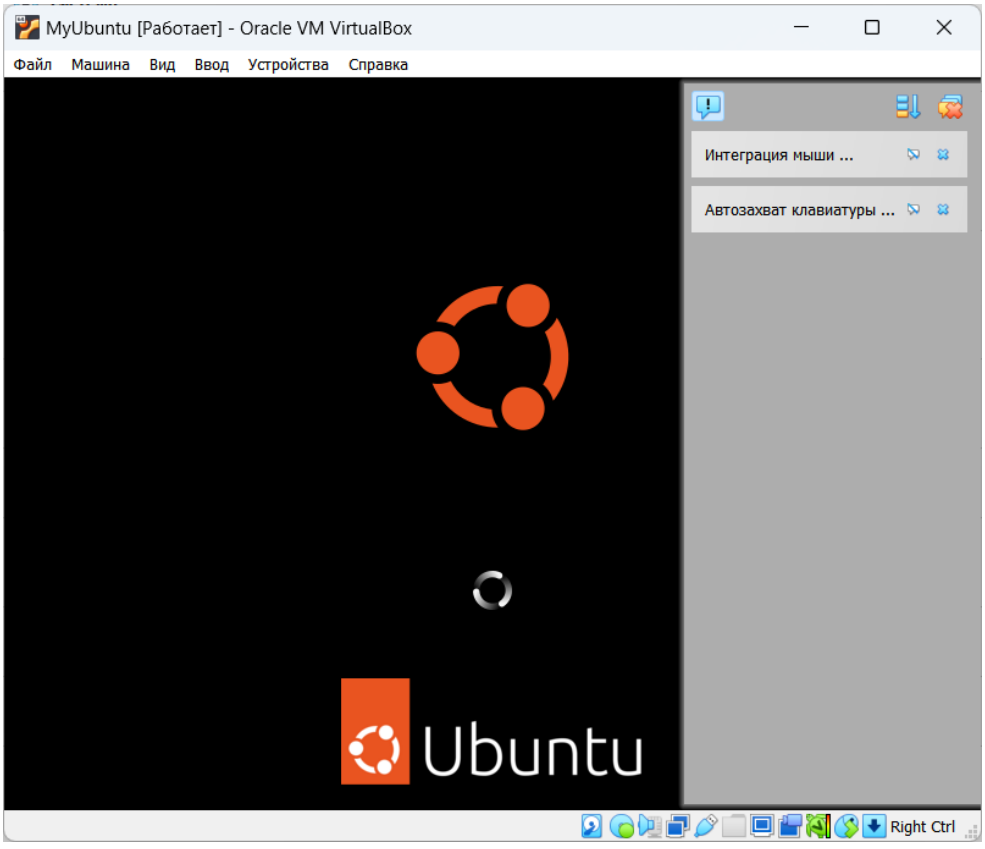

Рисунок 9 Установка Ubuntu

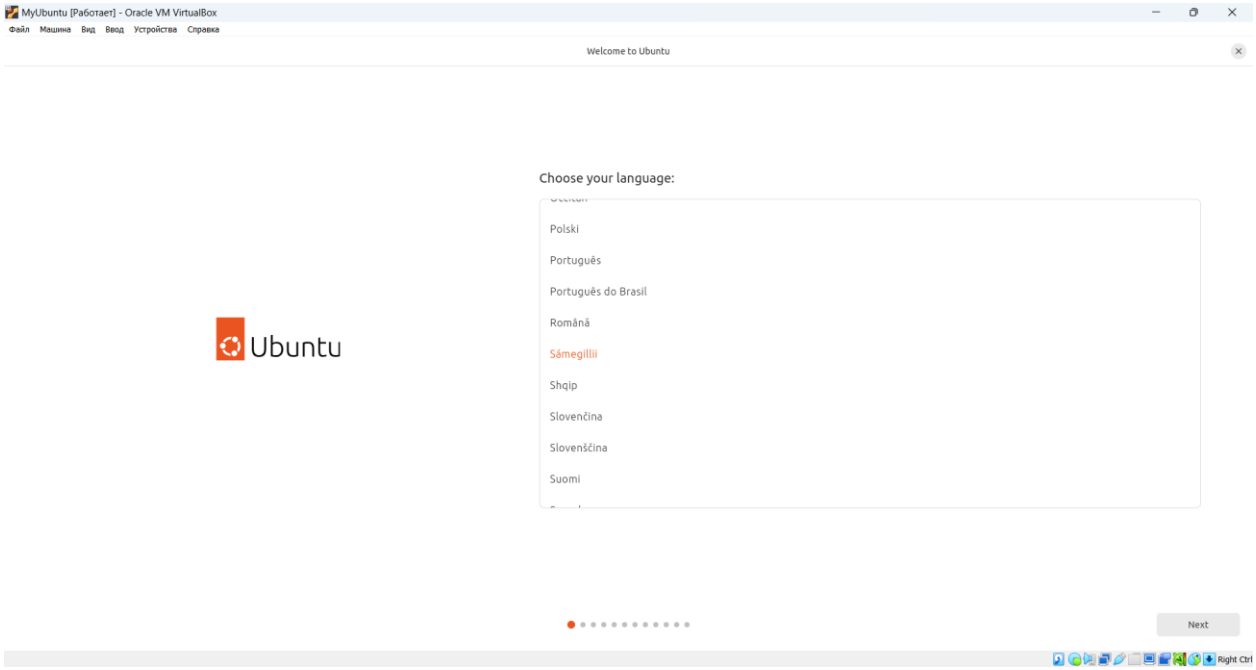

Рисунок 10 Выбор системного языка

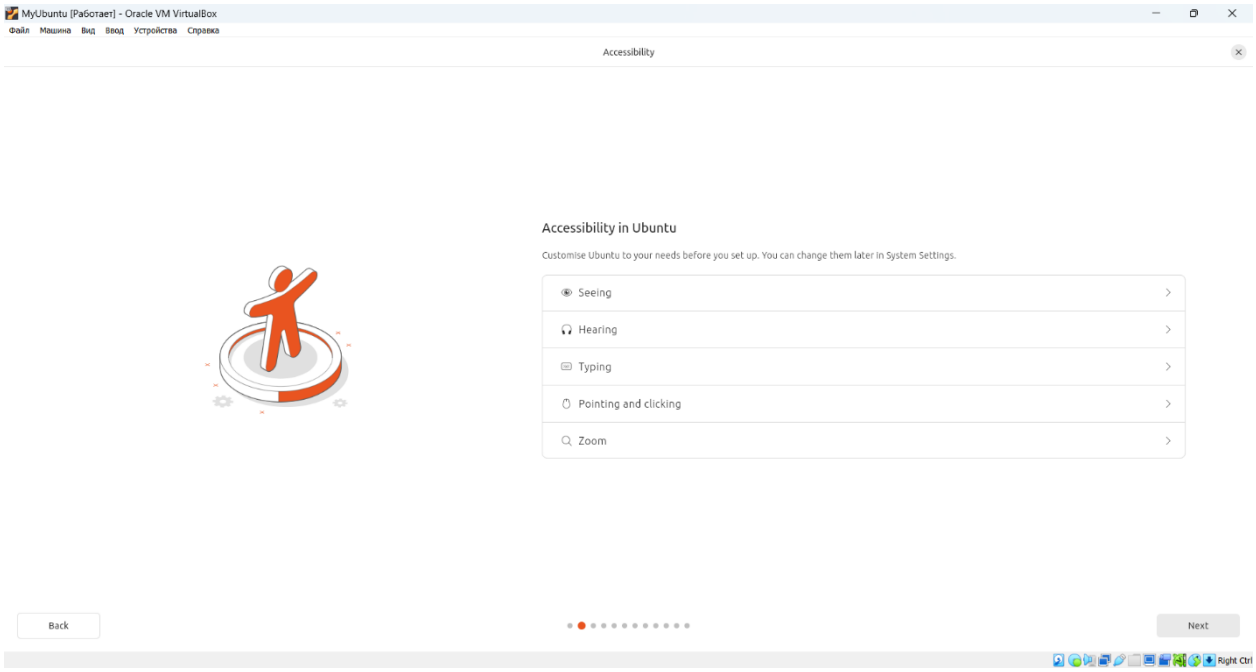

Рисунок 11 Выбор специальных возиожностей

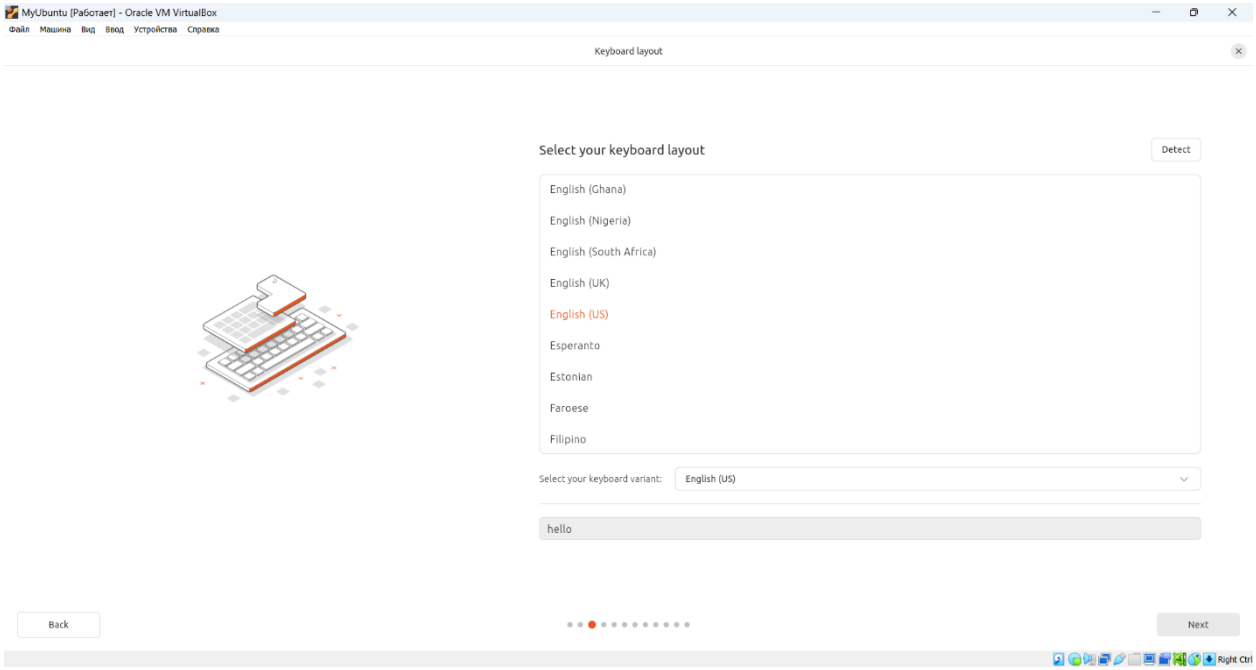

Рисунок 12 Выбор системной раскладки языка

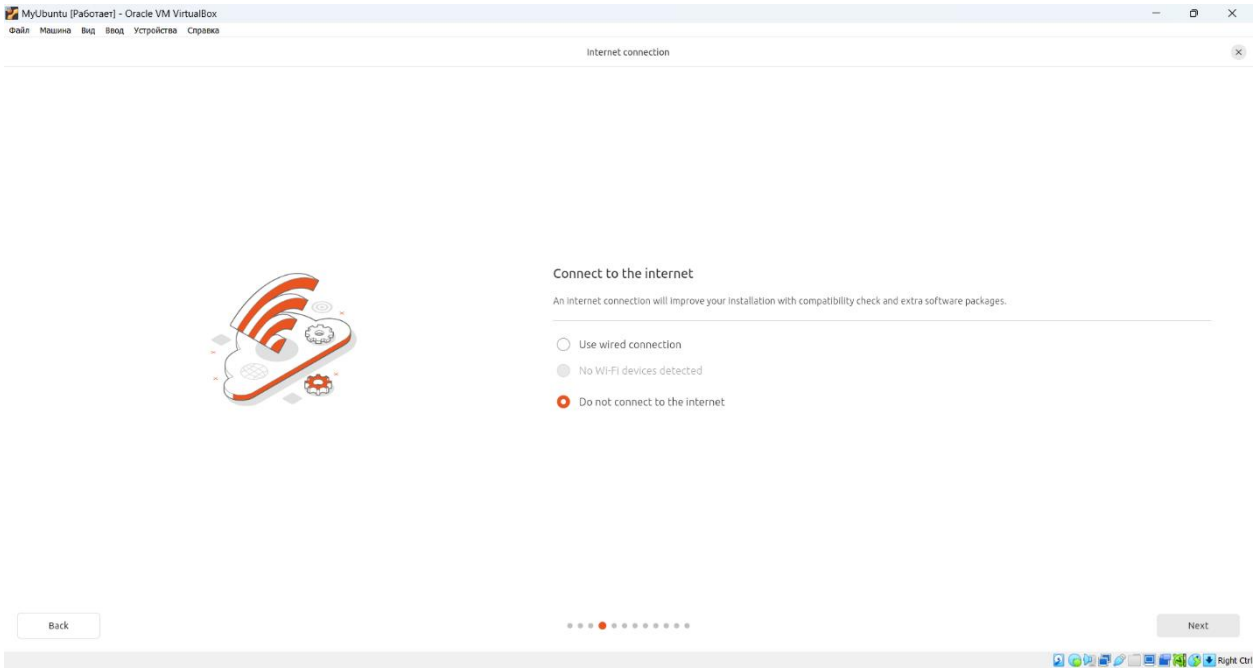

Рисунок 13 Подключение к интернету

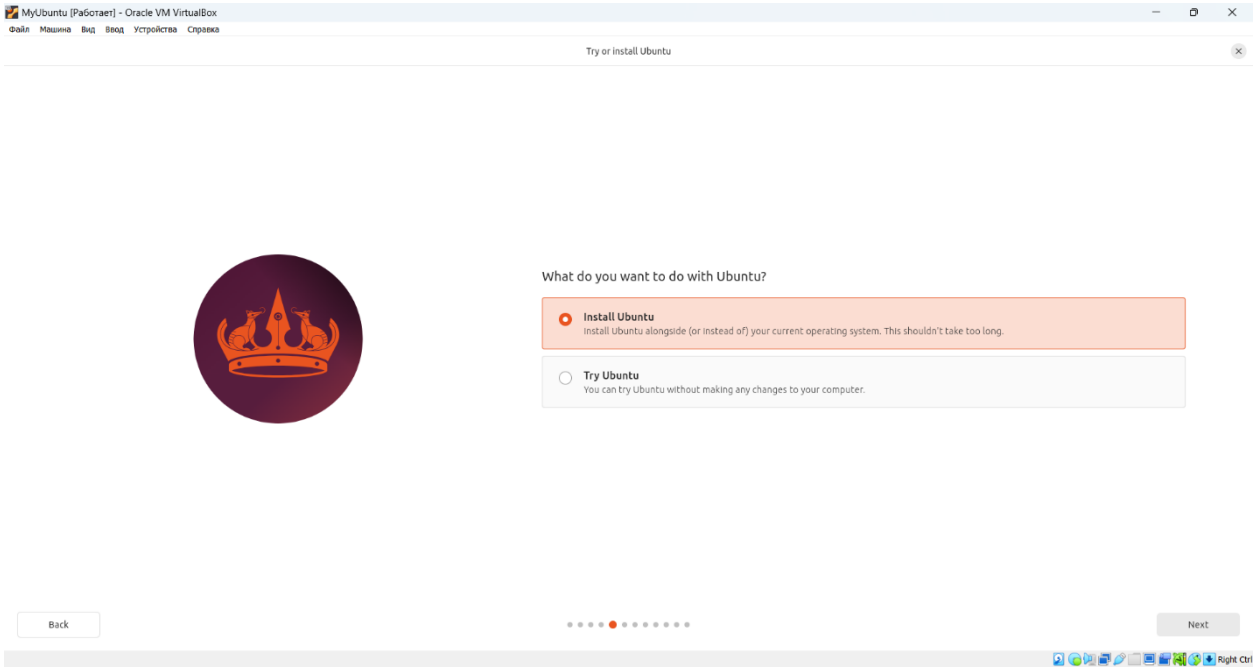

Рисунок 14 Выбор типа установки

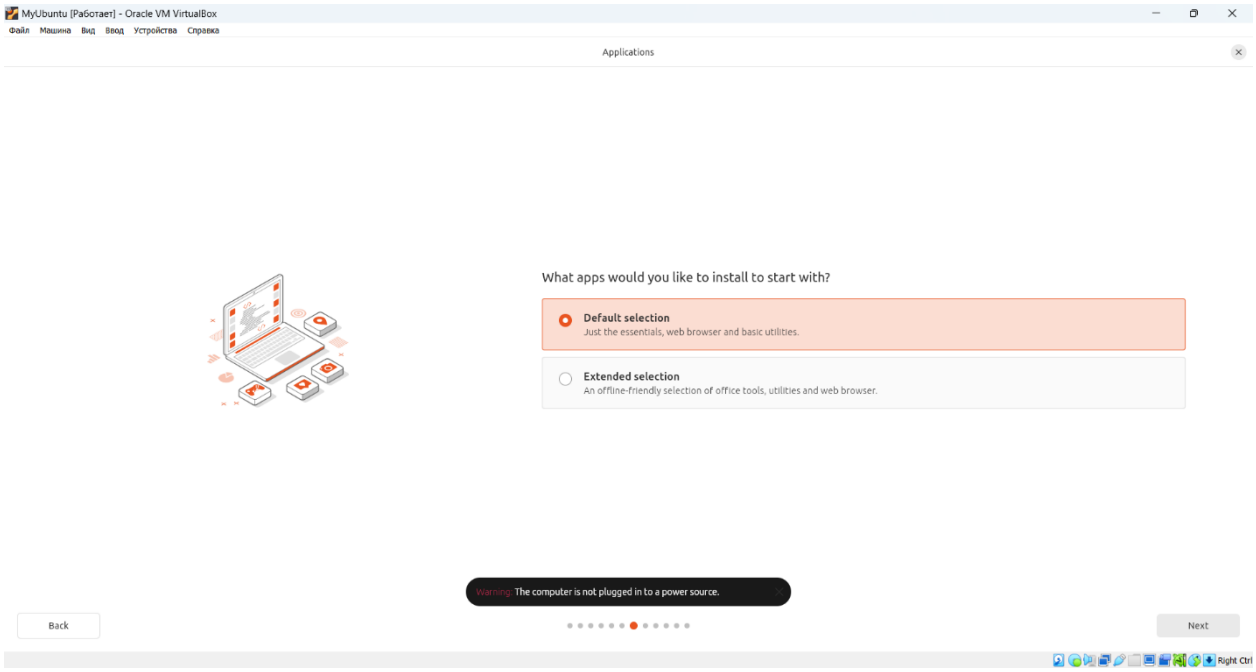

Рисунок 15 Выбор готового ПО в системе

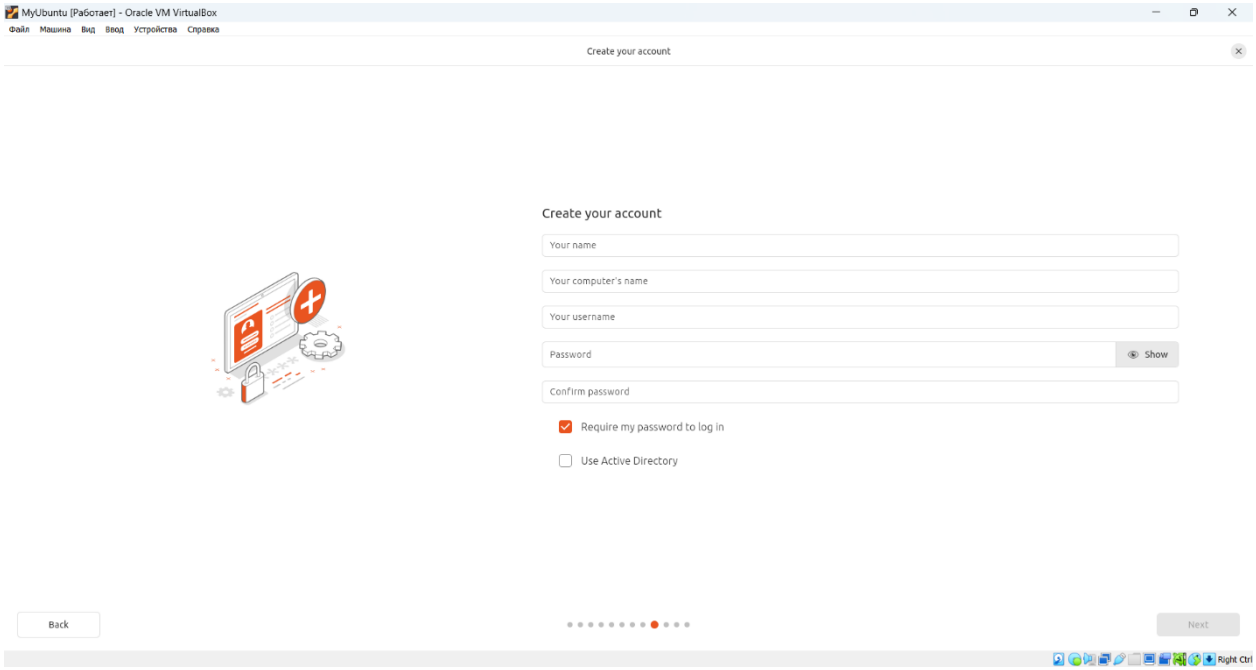

Рисунок 16 Конфигурация пользовательских данных

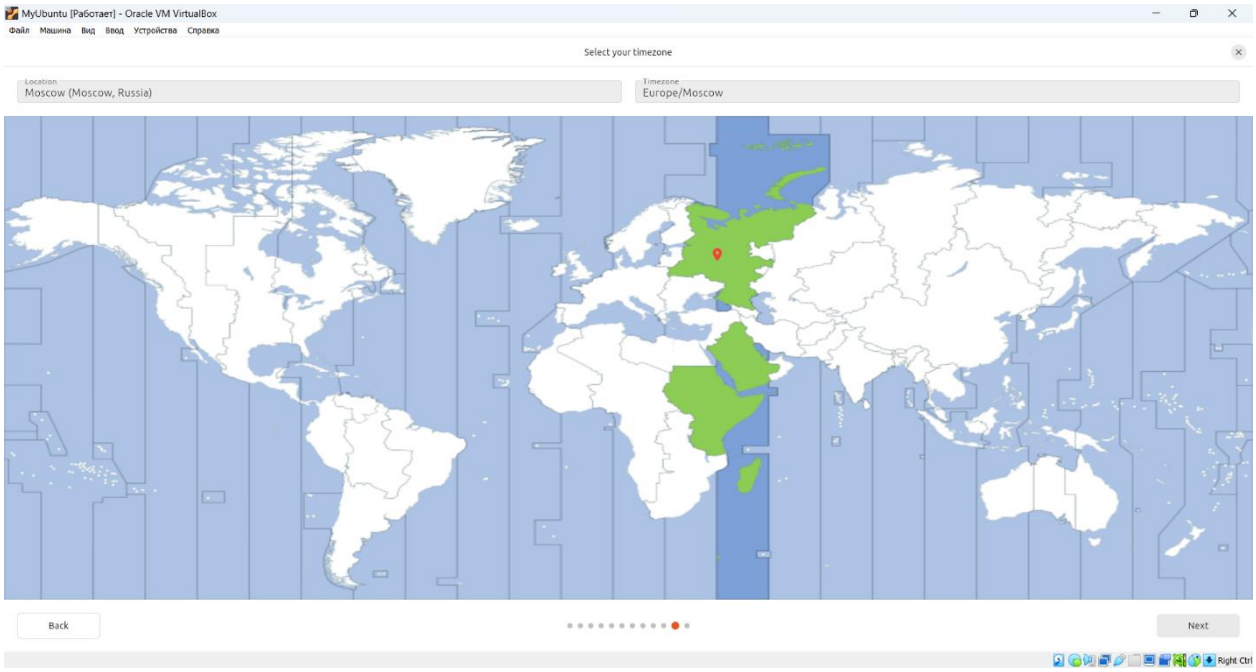

Рисунок 17 Выбор часового пояса

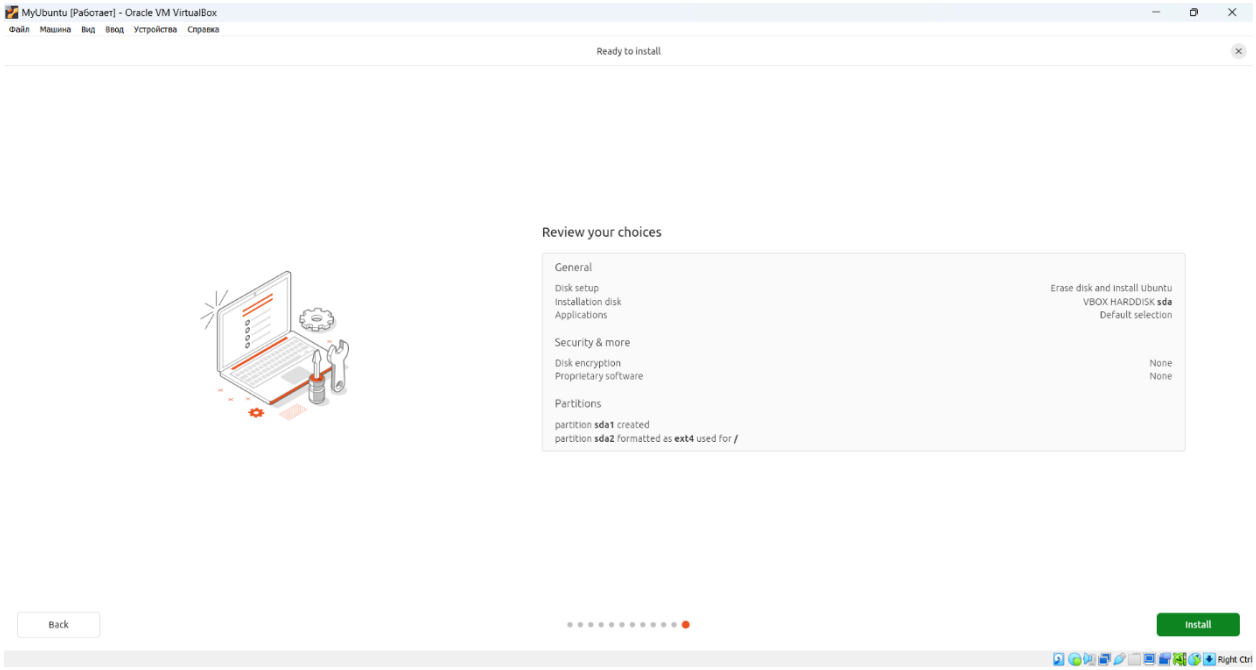

Рисунок 18 Окно для установки

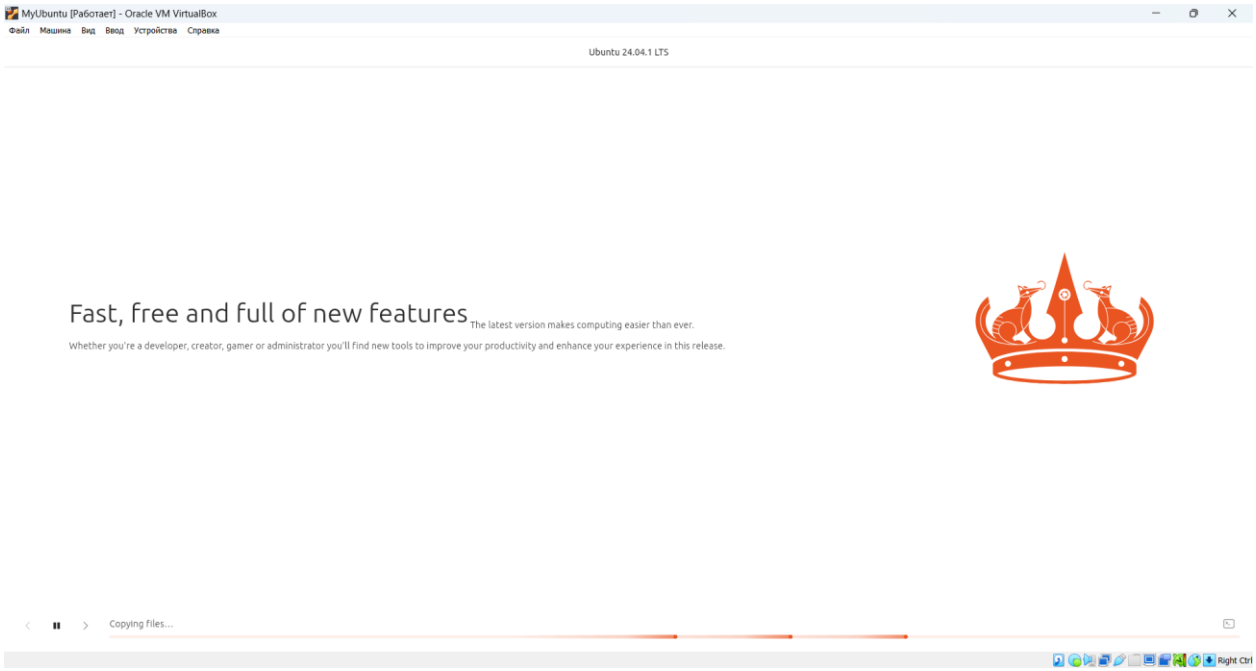

Рисунок 19 Установка Ubuntu

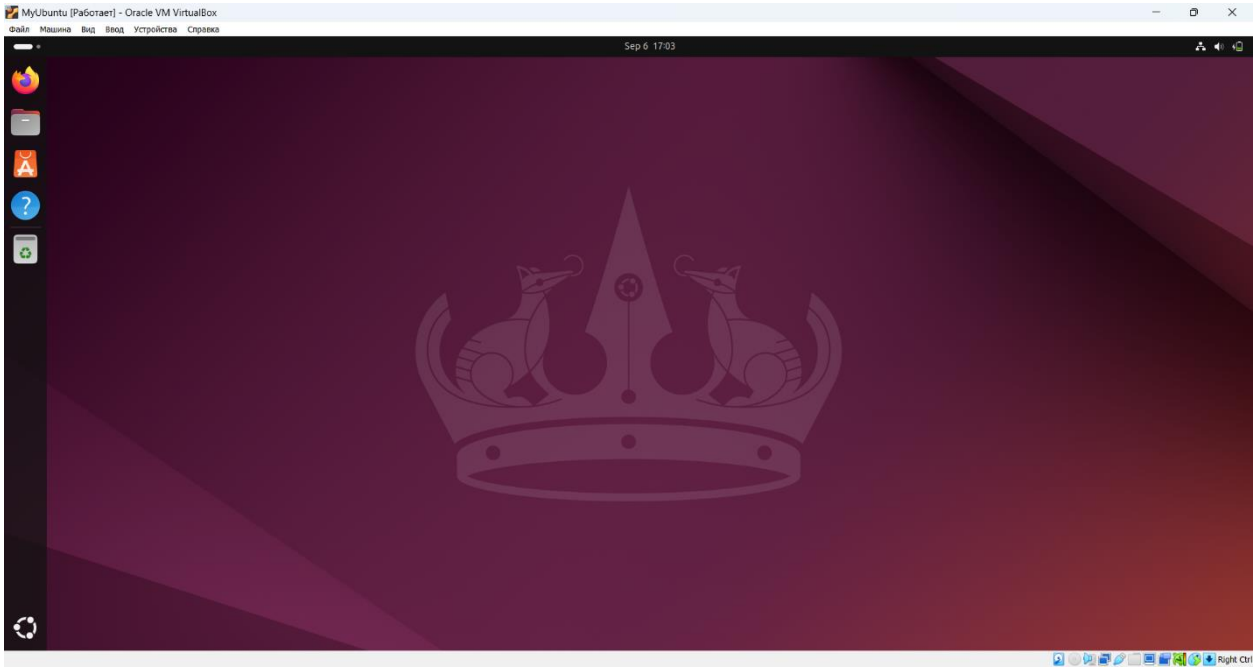

Рисунок 20 Система Ubuntu

Ответы на контрольные вопросы:

1) Укажите, как реализуется модель Linux-системы.

Linux-системы реализуются на модульных принципах, стандартах и соглашениях, заложенных в Unix в течение 1970-х и 1980-х годов. Такая система использует монолитное ядро, которое управляет процессами, сетевыми функциями, периферией и доступом к файловой системе. Драйверы устройств либо интегрированы непосредственно в ядро, либо добавлены в виде модулей, загружаемых во время работы системы. Отдельные программы, взаимодействуя с ядром, обеспечивают функции системы более высокого уровня. Например, пользовательские компоненты GNU являются важной частью большинства Линукс-систем, включающей в себя наиболее распространённые реализации библиотеки языка Си, популярных оболочек операционной системы, и многих других общих инструментов Unix, которые выполняют многие основные задачи операционной системы. Графический интерфейс пользователя (или GUI) в большинстве систем Linux построен на основе X Window System. В Linux-системах пользователи работают через интерфейс командной строки (CLI), 8 графический интерфейс пользователя (GUI), или, в случае встраиваемых систем, через элементы управления соответствующих аппаратных средств. Настольные системы, как правило, имеют графический пользовательский интерфейс, в котором командная строка доступна через окно эмулятора терминала или в отдельной виртуальной консоли. Большинство низкоуровневых компонентов Линукс, включая пользовательские компоненты GNU, используют исключительно

командную строку. Командная строка особенно хорошо подходит для автоматизации повторяющихся или отложенных задач, а также предоставляет очень простой механизм межпроцессного взаимодействия. Программа графического эмулятора терминала часто используется для доступа к командной строке с рабочего стола Linux. Дистрибутивы, специально разработанные для серверов, могут использовать командную строку в качестве единственного интерфейса. На настольных системах наибольшей популярностью пользуются пользовательские интерфейсы, основанные на таких средах рабочего стола как KDE Plasma Desktop, GNOME и Xfce, хотя также существует целый ряд других пользовательских интерфейсов. Самые популярные пользовательские интерфейсы основаны на X Window System, которая предоставляет прозрачность сети и позволяет графическим приложениям, работающим на одном компьютере, отображаться на другом компьютере, на котором пользователь может взаимодействовать с ними.

2) Раскройте способ нумерации версий ядра Linux. Укажите, за что отвечает каждая часть.

Номер версии ядра Linux в настоящее время содержит четыре числа, следуя недавнему изменению в долго используемой до этого политике схемы версий, основанной на трёх числах. Для иллюстрации допустим, что номер версии составлен таким образом: A.B.C[.D] (например 2.2.1, 2.4.13 или 2.6.12.3). Число A обозначает версию ядра. Оно изменяется менее часто и только тогда, когда вносятся значительные изменения в код и концепцию ядра. Оно изменялось дважды в истории ядра: в 1994 (версия 1.0) и в 1996 (версия 2.0). Число B обозначает старшую версию ревизии ядра. Чётные числа обозначают стабильные ревизии, то есть те, которые предназначены для промышленного использования, такие как 1.2, 2.4 или 2.6. Нечётные числа обозначают ревизии для разработчиков, такие как 1.1 или 2.5. Они предназначены для тестирования новых улучшений и драйверов до тех пор, пока они не станут достаточно стабильными для того, чтобы быть включёнными в стабильный выпуск. Число C обозначает младшую версию ревизии ядра. В старой трёхчисловой схеме нумерации, оно изменялось тогда, когда в ядро включались заплатки связанные с безопасностью, исправления ошибок, новые улучшения или драйверы. С новой политикой нумерации, однако, оно изменяется только тогда, когда вносятся новые драйверы или улучшения; небольшие исправления поддерживаются числом D. Число D впервые появилось после случая, когда в коде ядра версии 2.6.8 была обнаружена грубая, требующая незамедлительного исправления ошибка, связанная с NFS. Однако, было недостаточно других изменений, для того чтобы это послужило причиной для выпуска новой младшей ревизии (которой должна была стать 2.6.9). Поэтому была выпущена версия 2.6.8.1 с

единственным исправлением в виде исправления для этой ошибки. С ядра 2.6.11, эта нумерация была адаптирована в качестве новой официальной политики версий. Исправления ошибок и заплатки безопасности теперь обозначаются с помощью четвёртого числа, тогда как большие изменения выполняются в изменениях младшей версии ревизии ядра (число С)

3) Перечислите и опишите основные этапы процесса загрузки Linux.

При загрузке компьютера происходит последовательная передача управления от системной прошивки компьютера (BIOS или UEFI) к загрузчику, а от него — к ядру. Затем ядро запускает планировщик (для реализации многозадачности) и выполняет программу init (которая настраивает пользовательское окружение и позволяет осуществлять взаимодействие с пользователем и вход в систему), после чего ядро переходит в состояние бездействия до тех пор, пока не получит внешний вызов. Основные этапы загрузки: 1. Системная прошивка компьютера выполняет первичную проверку и инициализацию аппаратного обеспечения. 2. В случае BIOS прошивка загружает в оперативную память и выполняет загрузочный код с одного из разделов заданного загрузочного устройства, который содержит фазу 1 загрузчика Linux. Фаза 1 загружает фазу 2 (значительный по размеру код загрузчика). Некоторые загрузчики могут использовать для этого промежуточный этап (под названием фаза 1,5), поскольку современные диски большого объёма могут некорректно считываться без дальнейшего кода. В случае UEFI запускается загрузчик загруженный со служебного раздела (EFS), который выбирается согласно настройкам приоритета загрузки определенного в энергонезависимой памяти компьютера. При этом возможна загрузка не только специализированного загрузчика, но можно загрузить и непосредственно ядро Linux (для этого ядро должно быть собрано с опцией EFI_STUB). 3. Загрузчик зачастую предлагает пользователю меню с доступными вариантами загрузки. После выбора или после заданного тайм-аута загрузчик загружает ядро. 4. Загруженное ядро распаковывается в памяти, настраивает системные функции, такие как работа необходимого оборудования и управление страницами памяти, после чего делает вызов start_kernel(). 5. После этого start_kernel() выполняет основную настройку системы (прерывания, остальные функции управления памятью, инициализацию устройств, драйверов и т. д.), а потом порождает процесс бездействия, диспетчер и отдельно от них — процесс init (выполняющийся в пользовательском пространстве). 6. Планировщик начинает более эффективно управлять системой, в то время как ядро переходит к бездействию. 7. Процесс init выполняет необходимые сценарии, которые настраивают все службы и структуры, не относящиеся к уровню ядра, в результате чего будет создано пользовательское окружение, и пользователю будет предоставлен экран входа в систему.

4) Приведите примеры загрузчиков операционных систем.

Linux поддерживает несколько загрузчиков, которые позволяют пользователю выбирать операционную систему для загрузки при запуске компьютера. Некоторые из наиболее популярных загрузчиков операционных систем Linux включают: GRUB (GRand Unified Bootloader): GRUB является одним из самых распространенных загрузчиков для Linux. Он обладает богатыми функциональными возможностями и поддерживает множество операционных систем. GRUB предоставляет интерактивное меню выбора операционной системы и параметров загрузки. LILO (LInux LOader): 10 LILO был популярным загрузчиком для Linux, но сейчас его использование снизилось. Он предоставляет базовые функции загрузки и конфигурируется через файл конфигурации. SYSLINUX: SYSLINUX - это загрузчик, который обычно используется для загрузки Linux с съемных носителей, таких как USB-флешки или CD/DVD. Он хорошо подходит для создания загрузочных носителей. rEFInd: rEFInd - это загрузчик, который обеспечивает поддержку для загрузки Linux на компьютерах с UEFI (Unified Extensible Firmware Interface). Он предоставляет графический интерфейс для выбора операционной системы и имеет множество пользовательских настроек. системный загрузчик U-Boot: U-Boot - это загрузчик, который обычно используется во встроенных системах и на устройствах с архитектурой ARM или PowerPC. Он позволяет загружать и запускать Linux на таких устройствах. Эти загрузчики могут быть настроены и адаптированы под конкретные потребности и аппаратное обеспечение системы. GRUB остается одним из самых распространенных и мощных загрузчиков для большинства дистрибутивов Linux.

5) Опишите основные функции загрузчиков операционной системы.

Загрузчики операционной системы выполняют ряд важных функций при запуске компьютера. Вот основные функции загрузчиков операционной системы: Инициализация аппаратного обеспечения (Hardware Initialization): Загрузчики выполняют начальную инициализацию аппаратного обеспечения, включая процессор, память, дисковые устройства и периферийные устройства. Это позволяет операционной системе корректно взаимодействовать с аппаратным обеспечением. Выбор операционной системы (OS Selection): Загрузчики предоставляют пользователю или автоматически выбранный выбор операционной системы, которую следует загрузить. Это может включать в себя выбор между разными версиями операционных систем, разными ядрами Linux или разными дистрибутивами. Загрузка ядра операционной системы (Kernel Loading): Загрузчики загружают исполняемое ядро операционной системы в память. Ядро - это основная часть операционной системы, которая управляет аппаратным

обеспечением и предоставляет интерфейс для работы с приложениями. Инициализация и загрузка дополнительных модулей (Init and Module Loading): После загрузки ядра загрузчики могут инициализировать и загрузить дополнительные модули или драйверы, которые могут быть необходимы для поддержки специфического аппаратного обеспечения или функций. Инициализация окружения (Environment Initialization): 11 Загрузчики могут настроить окружение для ядра и операционной системы, включая параметры загрузки, переменные окружения и другие конфигурационные настройки. Загрузка и выполнение инструкций начальной загрузки (Boot Instructions Execution): Загрузчики могут выполнять дополнительные инструкции начальной загрузки, которые могут включать в себя задачи, такие как проверка целостности файловой системы, решение проблем с загрузкой или настройку параметров ядра. Завершение работы загрузчика (Bootloader Exit): По завершении своей работы загрузчики передают управление ядру операционной системы, которое затем начинает загрузку и инициализацию операционной системы. Управление загрузкой операционной системы (Boot Management): Загрузчики могут предоставлять возможности управления процессом загрузки, такие как выбор ядра, восстановление системы или изменение параметров загрузки. Загрузчики играют критическую роль в процессе загрузки операционной системы, обеспечивая корректное взаимодействие между аппаратным обеспечением и программным обеспечением операционной системы. Разные операционные системы могут использовать разные загрузчики, но их основные функции остаются схожими.

6) Раскройте сущность фазы загрузчика при загрузке через BIOS. 1.

Загрузчик 1-й фазы считывается BIOS из MBR (главной загрузочной записи). Он загружает оставшуюся часть загрузчика (2-ю фазу). Если вторая фаза находится на большом диске, иногда загружается промежуточная фаза 1,5, которая содержит дополнительный код, позволяющий считывать цилиндры с номерами более 1024 (диски LBA). Загрузчик фазы 1,5 хранится (если это необходимо) в MBR или в загрузочном разделе. 3. Выполняется вторая фаза загрузчика и отображает меню запуска GRUB. Оно также позволяет выбрать среду выполнения и просмотреть параметры системы. 4. Когда операционная система выбрана, она загружается и ей передаётся управление. GRUB поддерживает и прямой, и цепной способ загрузки, а также LBA, ext2, и «истинно командно-ориентированную, дооперационную среду на машинах x86». Он имеет три интерфейса: меню выбора, редактор настроек и командную консоль.

7) Раскройте сущность фазы загрузчика при загрузке через UEFI. 1.

Загруженный со служебного раздела EFS GRUB (специальная версия бинарного файла, который умеет загружать UEFI) содержит в себе все необходимые компоненты для доступа к файловой системе /boot где находятся конфигурация и дополнительные файлы загрузчика. 2. Отображается меню загрузчика и отображает меню запуска GRUB. Оно также позволяет выбрать среду выполнения и просмотреть параметры системы. 3. Когда операционная система выбрана, она загружается и ей передаётся управление.

8) Укажите основные возможности загрузчика операционной системы GRUB.

GRUB (GRand Unified Bootloader) - это один из наиболее популярных загрузчиков операционной системы для систем Linux и других UNIX-подобных операционных систем. Он обладает множеством возможностей, которые обеспечивают гибкость и контроль над процессом загрузки. Вот основные возможности GRUB: Поддержка множества операционных систем: GRUB позволяет загружать несколько операционных систем на одном компьютере и предоставляет пользователю выбор операционной системы при загрузке. Интерактивное меню выбора: GRUB предоставляет пользовательский интерфейс в виде интерактивного меню, где пользователь может выбрать операционную систему и дополнительные параметры загрузки. Поддержка многих файловых систем: GRUB поддерживает множество файловых систем, включая ext2, ext3, ext4, Btrfs, NTFS, FAT, ZFS и другие. Это позволяет загружать операционные системы с разных разделов и дисков. Поддержка разных архитектур: GRUB поддерживает различные архитектуры, включая x86, x86-64 (AMD64), ARM, PowerPC и другие, что делает его универсальным загрузчиком для разных платформ. Модульная структура: GRUB построен на модульной архитектуре, что позволяет добавлять новые функции и драйверы без изменения основного кода загрузчика. Поддержка защиты паролем: GRUB может быть настроен для защиты паролем определенных операционных систем или параметров загрузки. Параметры загрузки: Пользователь может вручную настраивать параметры загрузки операционной системы, что полезно для решения проблем с загрузкой или настройки системы. Скрипты загрузки: GRUB поддерживает скрипты загрузки, которые могут выполнять различные действия при загрузке, такие как установка переменных окружения, автоматический выбор операционной системы и другие. Работа с разделами и дисками: GRUB предоставляет команды для работы с разделами и дисками, что позволяет проверять целостность файловой системы, выполнять резервное копирование и другие операции. Обновление конфигурации автоматически: GRUB может автоматически обновлять свою конфигурацию при установке или обновлении операционных систем, что упрощает

управление загрузкой. Работа с разными режимами и разрешениями экрана: GRUB поддерживает разные режимы и разрешения экрана, что полезно для загрузки операционных систем на компьютерах с разными мониторами и видеокартами. Эти возможности делают GRUB мощным и гибким инструментом для управления загрузкой операционных систем и обеспечивают пользователю полный контроль над процессом загрузки.

9) Раскройте сущность фазы ядра.

Ядро Linux управляет главными функциями, такими как управление памятью, диспетчер задач, ввод-вывод, межпроцессное взаимодействие и общее управление системой. Загрузка проходит в два этапа: на первом ядро (в виде сжатого файла-образа) загружается в оперативную память и распаковывается, далее настраиваются такие базовые функции как основное управление памятью. Затем управление в последний раз передается основному процессу запуска ядра. Как только ядро становится полностью работоспособным (т. е. загруженным и выполнившим свой код), оно находит и запускает процесс init, который самостоятельно настраивает пользовательское пространство и процессы, необходимые для функционирования пользовательского окружения и итогового входа в систему. Само ядро переходит в режим бездействия и готовности к вызовам со стороны других процессов.

10) Раскройте сущность этапов загрузки ядра и запуска ядра.

Этап загрузки ядра Ядро при загрузке обычно имеет вид файла-образа, сжатого в формат zImage или bzImage с помощью zlib. В нём содержится головная программа, которая проводит минимальную настройку оборудования, распаковывает образ целиком в верхнюю памятьи монтирует RAM-диск, если он предусмотрен. После этого она выполняет запуск ядра посредством ./arch/x86/boot/head и процесса startup_32() (для процессоров семейства x86). Этап запуска ядра Функция запуска ядра (также называемая своппер или процесс 0) организует управление памятью (таблицы страниц и страничную организацию памяти), определяет тип процессора и дополнительные возможности (например, наличие математического сопроцессора), а затем переключается к архитектурно-независимому функционалу ядра Linux путём вызова start_kernel() . start_kernel() выполняет множество задач инициализации. Она настраивает обработчики прерываний (IRQ), затем настраивает память, запускает процесс init (первый процесс пользовательского режима), а затем запускает задачу бездействия вызовом cpu_idle() . Следует заметить, что процесс запуска ядра также монтирует инициирующий RAM-диск («initrd»), который ранее был загружен в роли временной корневой файловой системы в фазе загрузки. Это позволяет загружать модули драйверов, не опираясь на другие физические устройства и

драйверы, и поддерживать небольшой размер ядра. Корневая файловая система впоследствии подменяется с помощью вызова pivot_root() , который размонтирует временную и заменяет её настоящей корневой ФС, как только последняя станет доступна. Использованная временной системой память затем освобождается. Таким образом, ядро инициализирует устройства, монтирует указанную загрузчиком файловую систему в режиме «только чтение» и запускает процесс init (/sbin/init), который обозначается как первый процесс, запущенный системой (с идентификатором процесса PID = 1). Соответствующие сообщения выводит ядро (при монтировании файловой системы) и init (при запуске одноимённого процесса). Ядро также может выполнить initrd для обработки настроек и инициализации устройств до монтирования корневой файловой системы. По заявлению компании «Red Hat», детали процесса загрузки на этом этапе можно подытожить так: «Когда загружается ядро, оно сразу же инициализирует и конфигурирует память компьютера и настраивает различное подключённое к системе оборудование, включая все процессоры, подсистемы ввода-вывода и устройства хранения данных. Затем оно ищет сжатый образ initrd в заранее определённом участке памяти, распаковывает его, монтирует и загружает все необходимые драйверы. Затем оно инициализирует виртуальные 14 устройства, связанные с файловой системой, например LVM или программные RAID□массивы, прежде чем демонтировать образ диска initrd и освободить всю память, ранее занимаемую образом. Потом ядро создает корневое устройство, монтирует корневой раздел только для чтения и освобождает всю неиспользованную память. К этому времени ядро загружено в память и работоспособно. Тем не менее, поскольку нет пользовательских программ для осуществления осмысленного ввода данных в систему, с ней мало что можно делать.» Теперь, когда включены прерывания, диспетчер может принять общее управление системой, чтобы обеспечить вытесняющую многозадачность, а процесс init остается продолжать загрузку пользовательского окружения в пространстве пользователя.

11) Охарактеризуйте процесс init, укажите основную задачу процесса.

Init является родителем всех процессов. Его главная задача — создавать процессы по сценарию из файла /etc/inittab . В этом файле обычно содержатся записи, указывающие init породить getty для каждой линии, по которой пользователи могут входить в систему. Он также контролирует автономные процессы, требуемые какой-либо системе. Уровень выполнения — программная конфигурация системы, которая позволяет существовать только заданной группе процессов. Процессы, порождаемые init на каждом из таких уровней выполнения, определяются в файле /etc/inittab . По сути init организует и поддерживает всё пользовательское пространство, что включает в себя также проверку и монтирование файловых систем, запуск нужных

пользовательских служб и, переключение в пользовательскую среду, когда запуск системы завершится. Он похож на процессы init в Unix и BSD, от которых произошёл, но в некоторых случаях он изменён или переделан. В обычной системе Linux init имеет параметр, известный как уровень выполнения, принимающий значения от 1 до 6 и определяющий, какие подсистемы следует включить. Для каждого уровня выполнения есть собственные сценарии, которые регламентируют различные процессы, участвующие в установлении или снятии данного уровня, и именно эти сценарии считаются необходимыми для процесса загрузки. Сценарии init обычно хранятся в каталогах с именами вида /etc/rc… Главный файл конфигурации уровней для init — /etc/inittab . Во время загрузки системы он проверяет, описан ли уровень по умолчанию в /etc/inittab , а если же нет — запрашивает его через системную консоль. Затем он продолжает выполнять все соответствующие сценарии загрузки для этого уровня, включая загрузку модулей, проверку целостности файловой системы (которая монтировалась только для чтения), перемонтирование её для чтения-записи и настройку сети. В частности, по сообщению Red Hat, процесс init следует такой схеме: 1. Он просматривает сценарий sysinit , который "устанавливает путь к среде, запускает swap , проверяет файловые системы и делает всё, что необходимо для инициализации системы. Это, в частности, системные и аппаратные часы, специальные процессы для последовательного порта и т. п. 2. Затем init просматривает конфигурацию, указанную для заданного уровня выполнения. 3. После этого init устанавливает исходную библиотеку функций для системы. Это определяет, как следует запустить или снять программу и как определить её PID. 4. Затем он запускает все предусмотренные процессы и создает сессию входа пользователя в систему. После того, как он породил все заданные процессы, init переходит в режим ожидания и ждет одного из трёх событий: 1. Нормального или аварийного завершения порождённых процессов. 2. Сигнала аварии питания. 15 3. Запроса от /sbin/telinit на изменение уровня выполнения. Это относится к программе init в стиле UNIX System V. Другие программы init могут вести себя иначе.

Вывод: в ходе лабораторной работы была установлены операционная система Linux, дистрибутив Ubuntu на виртуальную машину. Была произведена настройка пользовательского интерфейса.