МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МАГНИТОГОРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМ. Г. И. НОСОВА»
(ФГБОУ ВО «МГТУ ИМ. Г.И. НОСОВА»)

Кафедра вычислительной техники и программирования

КУРСОВАЯ РАБОТА

по дисциплине «Операционные системы семейства *nix»

на тему: «Администрирование в операционной системе Fedora»

Исполнитель: Варламов М.Н, студент 4 курса, группа АВб–19–1

Руководитель: Кандидат пед. Наук, доцент кафедры ВТиП Ильина Е.А.

Работа допущена к защите «____» _____2022 г. _____

Работа защищена «____» _____ 2022 г. с оценкой _____

Магнитогорск, 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МАГНИТОГОРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМ. Г.И. НОСОВА»
(ФГБОУ ВО «МГТУ ИМ. Г.И. НОСОВА»)

Кафедра вычислительной техники и программирования

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Тема: «Администрирование в операционной системе Fedora»

Обучающемуся Варламову Максиму Николаевичу

Исходные данные: Документация по Fedora Workstation 36, документация по текстовому редактору vi, документация по администрированию в unix–подобных операционных системах, СМК-О-СМГТУ-42-09 Курсовой проект (работа): структура, содержание, общие правила выполнения и оформления.

Срок сдачи: «_____» _____ 2022 г.

Руководитель: _____ /Е.А. Ильина/

Задание получил: _____ /М.Н. Варламов/

Магнитогорск, 2022

СОДЕРЖАНИЕ

Введение

В современном мире операционные системы имеют огромный вес в мире информационных технологий. Они являются одной из основополагающих вещей, на основе которых выстраиваются собственные системы различной сложности и важности. Операционные системы являются основой для управления компьютером пользователями.

Умение использования операционной системы необходимо для пользователя. Как и в любом другом деле, необходимо знать основы той области, в которой производится работа. Основы использования операционными системами не является исключением. Пользователь должен понимать базовые концепции ОС, такие как:

- файловая система;
- запуск программ;
- создание файлов;
- редактирование файлов;
- пользователи в системе.

Как известно, компьютер может использоваться несколькими пользователями, как физически, так и удаленно. Поэтому безопасность является важным фактором в операционных системах. Данную тему также необходимо понимать при настройке ОС, для дальнейшего наилучшего и безопасного взаимодействия с ней.

Поэтому, настройка и администрирование в операционных системах является важными навыками для пользователя этой ОС, которые необходимо иметь для обеспечения правильной и безопасной работы с операционной системой.

1 Операционная система Fedora Linux 36 Workstation

1.1 Определение и функции операционной системы

Началом истории развития ЭВМ принято считать сороковые годы XX века. Тогда, вычислительная техника работала на основе электронных лампах. Это такие устройства, которые работают на принципе колебаний интенсивности электронного потока, перемещающегося в вакуумном пространстве стеклянной колбы по направлению от катода к аноду. Первой ЭВМ, построенной на данной технологии является “<„ , собранная в 1945 году. Главным отличием от всех механических предшественников являлась возможность программировать главные настройки. Главное программируемое устройство отправляло импульсы функциональным модулям, вызывавшие запуск предварительно установленных последовательностей, и получало ответные импульсы по завершению необходимой работы [1].

Хотя возможности “ENIAC„ превосходили изначальные задачи, для которых создавалась данная ЭВМ, она имела ряд значительных проблем, в число которых входила сложность поддержания работоспособности и перепрограммирования. Все «базовые», на данный момент, операции ввода–вывода, хранения информации, вычисления, требовали присутствия целой команды специалистов–операторов ЭВМ.

Последующие поколения ЭВМ имели меньшие размеры, более высокую производительность и уже не требовали присутствия целой команды специалистов. Однако было понятно, что требуется разработать такую систему, которая позволяла бы управлять работоспособностью без механического воздействия. Требуется система, которая должна автоматически управлять памятью, системами ввода–вывода, подключаемыми периферийными устройствами и выполнением ПО.

Так, в 1955 году была выпущена первая операционная система GM–HAA. Она была основана на системном мониторе и работала на больших машинах. Основная функция GM–HAA – автоматическое выполнение новой программы, когда старая программа завершилась.

В последующие года, с выпуском новых ЭВМ, появлялись и новые операционные системы, все более и более совершенные. Тогда же и сформировались основные требования к операционной системе и ее функции.

Операционная система – это комплекс программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.

К основными требованиям к операционной системе можно отнести:

Эффективность. Под эффективностью понимается степень соответствия системы своему назначению, которая оценивается некоторым множеством показателей эффективности.

Надежность и отказоустойчивость. Операционная система должна быть, по меньшей мере, так же надежна, как компьютер, на котором она работает. Система должна быть защищена как от внутренних, так и от внешних сбоев и отказов. В случае ошибки в программе или аппаратуре система должна обнаружить ошибку и попытаться исправить положение или, по крайней мере, постараться свести к минимуму ущерб, нанесенный этой ошибкой пользователям.

Безопасность. Ни один пользователь не хочет, чтобы другие пользователи ему мешали. ОС должна защищать пользователей и от воздействия чужих ошибок, и от попыток злонамеренного вмешательства (несанкционированного доступа).

Предсказуемость. Требования, которые пользователь может предъявить к системе, в большинстве случаев непредсказуемы. В то же время пользователь предпочитает, чтобы обслуживание не очень сильно менялось в течение предположительного времени.

Расширяемость. В отличие от аппаратных средств компьютера полезная жизнь операционных систем измеряется десятками лет. Примером может служить ОС Unix, да и MS–DOS. Операционные системы изменяются со временем, как правило, за счет приобретения новых свойств, например, поддержки новых типов внешних устройств или новых сетевых технологий.

Переносимость. В идеальном случае код ОС должен легко переноситься с процессора одного типа на процессор другого типа и с аппаратной платформы (которые различаются не только типом процессора, но и способом организации всей аппаратуры компьютера) одного типа на аппаратную платформу другого типа. Переносимые ОС имеют несколько вариантов реализации для разных платформ, такое свойство ОС называется также многоплатформенностью.

Удобство. Средства ОС должны быть простыми и гибкими, а логика ее работы ясна пользователю. Современные ОС ориентированы на обеспечение пользователю максимально возможного удобства при работе с ними.

Масштабируемость. Если ОС позволяет управлять компьютером с различным числом процессоров, обеспечивая линейное (или почти такое) возрастание производительности при увеличении числа процессоров, то такая ОС является масштабируемой. В масштабируемой ОС реализуется симметричная многопроцессорная обработка [2].

Основными же функциями операционной системы являются:

- запуск программ и контроль за их прохождением;
- управление оперативной памятью;
- управление устройствами ввода и вывода;
- управление внешней памятью;
- управление взаимодействием одновременно работающих задач;
- обработка вводимых команд для взаимодействия с пользователем [3].

1.2 Unix и Unix–подобные операционные системы

Работу первых ЭВМ с собственными ОС можно описать следующим алгоритмом:

1) компьютер включался;
2) находил на носителе информации первую команду программы, которую нужно выполнить;
3) выполнял эту команду и переходил к следующей;
4) так происходило, пока в программе не заканчивались шаги или она сама не останавливалась. Тогда специальными командами оператор говорил компьютеру, где найти код для другой программы, или запускал первую программу заново.

Получается, что они работали в однозадачном режиме: работает только одна программа, а для запуска второй нужно остановить первую. Однако вскоре такой подход стал неудобен.

Чтобы компьютер работал более эффективно, был реализован многозадачный подход к выполнению ПО.

Первые операционные системы, реализующие данный подход, были псевдомногозадачными. Это значит, что они не запускали одновременно несколько программ, а в цикле по очереди брали по одной команде из каждой программы и выполняли их. Так как переключение между командами из разных программ происходит быстро, то создаётся впечатление, что они работают одновременно. Однако, данный подход также просуществовал недолго. Выявились определенные критичные проблемы, которые и подтолкнули к созданию ОС Unix.

Создатели Unix Кен Томпсон и Деннис Ритчи решили проблему так:

- есть один центральный компьютер – сервер, на котором выполняются все программы;
- если кто–то хочет поработать за этим компьютером, то он подключается к нему не напрямую, а через терминал. Терминал – это монитор и клавиатура, которые соединены с сервером. Сам терминал ничего не считает, а только отправляет и получает результаты с сервера;
- пользователь вводит свой логин и пароль и получает право запускать на сервере программы, которые ему нужны. При этом на сервере может быть запущено одновременно много программ от разных пользователей;
- сервер помнит, кто на каком терминале зашёл под каким логином, поэтому результаты работы программы он отправляет в нужный терминал.

Со стороны пользователя кажется, что весь сервер в его распоряжении, но на самом деле сервером могут пользоваться одновременно десятки человек и не знать о том, что сервер выполняет что–то ещё. Отсюда и вытекает название операционной системы, которое произошло от сокращения Uniplexed Information and Computing Service (единый информационно–вычислительный сервис).

Помимо реализации полноценной многозадачности, ОС Unix привнесла в развитие операционных систем следующие, уже сейчас считающиеся базовыми, модели и концепции:

Файловая система с любой глубиной вложенности. Мы сейчас привыкли к папкам, в которых можно создавать другие папки, а в них третьи и так почти до бесконечности. Но до Unix глубина вложенности была ограничена – нельзя было создать, например, папку внутри другой папки.

Модель работы с файлами. Пользователю раньше нужно было самому предусмотреть формат, размер и физическое размещение файлов на диске. В Unix это всё взяла на себя операционная система.

Работа с программами напрямую. До Unix настройку работы всех программ можно было сделать только в командной строке. В новой системе можно было менять настройки программ прямо внутри них – именно так и устроены сейчас все программы.

Вывод всего как текста. Раньше компьютеры работали с битами и выводили битовые последовательности. Их нужно было отдельно разбивать на нужные фрагменты или использовать встроенные программы для перевода битов в байты, а из них – в текст.

В Unix единица вывода – это не бит, а байт. А в байт как раз умещается символ текста, а значит, с ним можно работать как с текстом: искать, склеивать с другими, отправлять в файл и так далее.

Язык С. Этот язык появился в Unix как замена языка B. Но B был интерпретируемым языком, и для запуска программ нужен был его интерпретатор. Язык C – компилируемый, а значит, готовые программы можно запускать на любом совместимом компьютере, даже если на нём нет компилятора C.

Протокол TCP/IP. До Unix этот протокол не был популярен, и компьютеры связывались друг с другом по более старому протоколу, который не имел столько возможностей. Теперь благодаря этой операционной системе весь мир пользуется интернетом, построенным на протоколе TCP/IP. Справедливости ради, этот протокол появился не в первой поставке Unix [4].

Политика безопасности. Ограничение или исключение несанкционированного доступа к информации и программным средствам.

При этом используются два основных понятия: объект и субъект системы. Объектом системы называют любой её идентифицируемый ресурс (например, файл или устройство). Доступом к объекту системы – некоторую заданную в ней операцию над этим объектом (чтение или запись). Действительным субъектом системы называют любую сущность, способную выполнять действия над объектами (имеющую к ним доступ).

Принимая во внимания всё вышеперечисленное, Unix является важной частью развития операционных систем. На ее основе был создан ряд других систем, продолжающих и дополняющих основы, введенные в данной ОС. Такие ОС стали называть Unix–подобные операционные системы. На рисунке 1 представлены все Unix–подобные ОС в виде древовидной структуры.

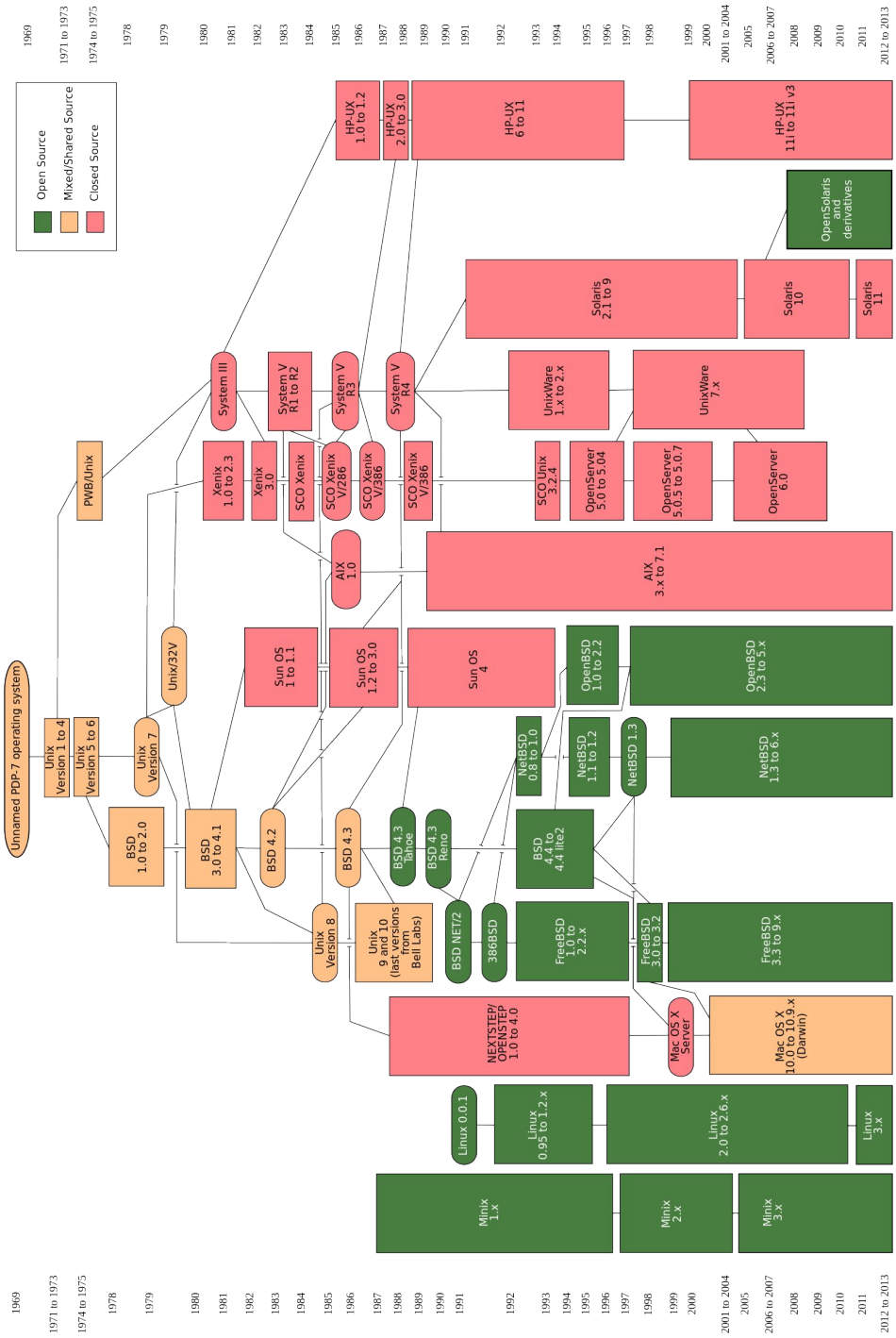

Рисунок 1 – Unix–подобные операционные системы в виде древовидной структуры

К минусам ОС Unix, на то время, можно отнести то, что распространение данной системы происходило на коммерческой основе. По этому, в начале 90–х годов, финским программистом Линусом Торвальдсом, была разработана открытая альтернатива Unix под названием Linux.

1.3 Ядро Linux

Главная, постоянно находящаяся в оперативной памяти часть ОС называется ядром (Kernel). Ядро ОС обрабатывает прерывания от устройств, выполняет запросы системных процессов и пользовательских приложений, распределяет виртуальную память, создает и уничтожает процессы, обеспечивает многозадачность посредством переключения между ними, содержит драйверы устройств, обслуживает файловую систему.

Ядро Linux монолитное, большая его часть хранится в одном файле. Однако, это не признак монолитного ядра, модули вполне могут храниться отдельно. Основная его особенность заключается в том, что оно обрабатывает все процессы, кроме пользовательских приложений. То есть управление процессами и памятью, драйверы, виртуальная файловая система, сетевой стек и многое другое – это всё заботы ядра, которые к тому же имеют самый высокий уровень доступа к аппаратной части компьютера.

Интерфейсы, имена переменных и структура каталогов системы определяются стандартами POSIX, что делает Linux UNIX–подобной системой. Линус Торвальдс, создатель ядра, выбрал UNIX по той причине, что имелась база приложений, необходимых для функционирования операционной системы, утилиты GNU.

Как было сказано ранее, у монолитного ядра самый широкий спектр задач. На верхнем уровне ядро обрабатывает поступающие системные вызовы, которые являются интерфейсом между ядром и пользовательскими приложениями. На нижнем уровне ядро обрабатывает аппаратные прерывания, сигналы, поступающие от периферии, процессора, памяти и так далее. На рисунке 2 можно увидеть увидеть основные составные части ядра Linux.

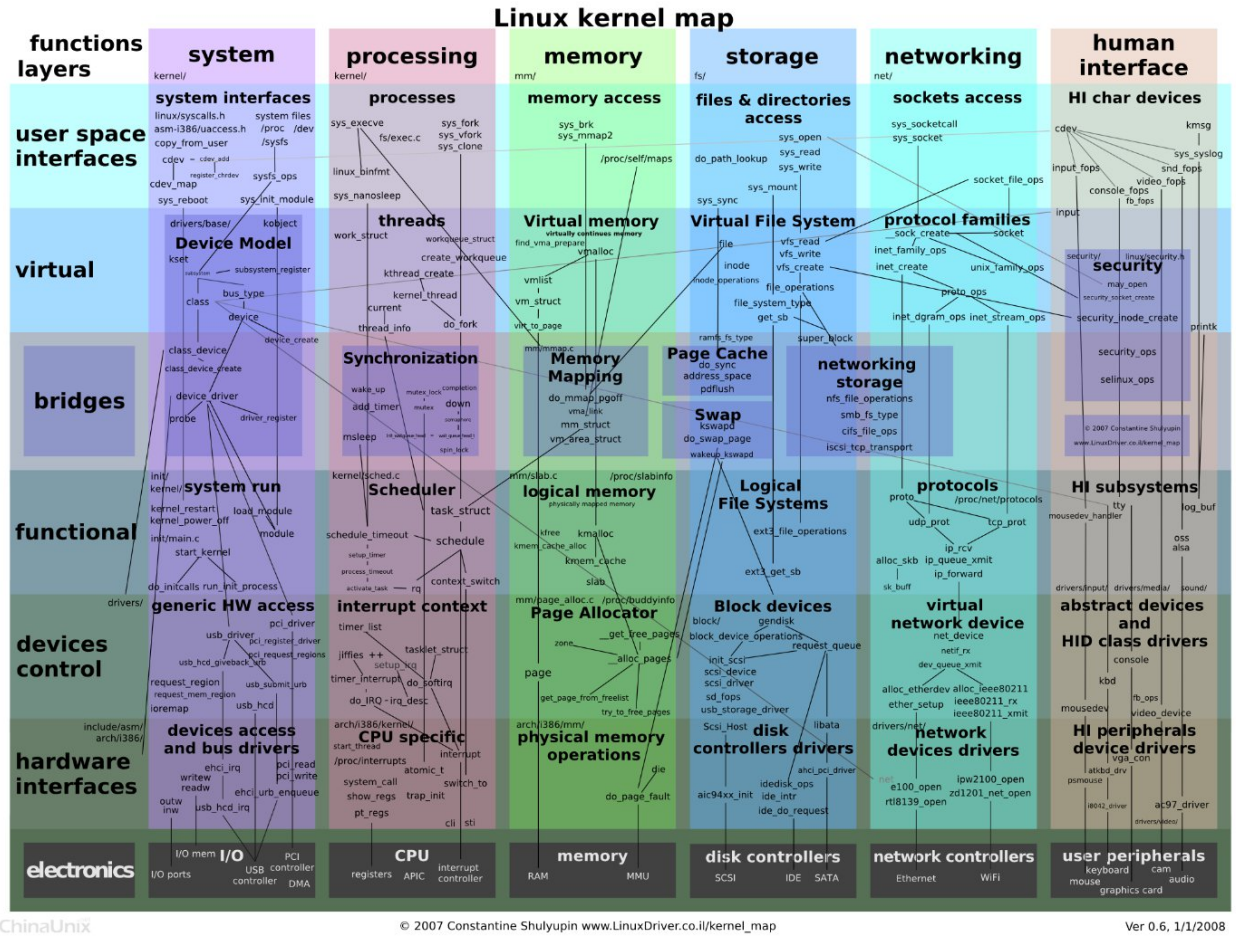

Рисунок 2 – Составные части ядра Linux

На обработке прерываний задачи ядра не заканчиваются, оно содержит в себе драйверы устройств. Драйверы нужны для того, чтобы обработать поступающие с устройств сигналы, а команды приложений перевести в машинный код.

Драйверы занимают большую часть ядра. Некоторые из них представлены сразу в виде бинарных файлов, что противоречит идеям фонда СПО. Версия ядра без закрытых драйверов называется Linux–libre, на практике его использование крайне затруднительно, так как собрать компьютер на основе комплектующих только с открытыми драйверами у вас едва ли получится.

Остальные задачи ядра – это работа с абстракциями. Например, планировщик создаёт виртуальные потоки, менеджер памяти выделяет и изолирует часть оперативной памяти под процесс, виртуальная файловая система создаёт единое пространство для хранения файлов, а сетевой модуль создаёт сокеты. Это одно из условий обеспечения высокого уровня безопасности, иначе одна программа могла бы беспрепятственно взять конфиденциальные данные из другой, например, ключи шифрования.

Система межпроцессного взаимодействия следит за тем, чтобы не возникало конфликтов при обращении к одним и тем же ресурсам компьютера, а также обеспечивает обмен данными между процессами.

Со стороны пользовательских приложений всё это выглядит как настоящее оборудование, с той лишь разницей, что общение с процессором и памятью происходит не напрямую, а с помощью системных вызовов. Для периферийных устройств имеются символьные и блочные ссылки в каталоге /dev, последние отличает то, что ни работают с блоками фиксированного размера.

Несмотря на то, что ядро контролирует все процессы, само по себе оно ничего не делает, ему нужны пользовательские программы и их процессы [5].

На основе данного ядра реализуются так называемые дистрибутивы Linux. Дистрибутив (distribute – распространять) – форма распространения программного обеспечения. В данном случае, форма распространения операционной системы Linux. Дистрибутив Linux – установочный пакет для развёртывания операционной системы, состоящей из ядра Linux, утилит GNU, дополнительного ПО и диспетчера пакетов. Он также может включать в себя пакет для установки дисплейного сервера и развёртывания среды рабочего стола.

Таким образом, Linux – ядро ОС, а дистрибутив Linux – установочный пакет какой–то из разновидностей этой операционной системы плюс дополнительные компоненты. Такие разновидности называют операционными системами на базе Linux [6].

1.4 Fedora Linux

История «Федоры» невозможна без упоминания компании Red Hat, которая приобрела многообещающий дистрибутив. Red Hat – это IT–компания, выпускающая программное обеспечение с 1995 года. Основным и наиболее известным ее продуктом является RHEL, или Red Hat Enterprise Linux. В отличие от большинства дистрибутивов, RHEL поддерживается на коммерческой основе и имеет платный доступ к обновлениям.

RHEL всегда пользовалась спросом, была достаточно передовой и качественной операционной системой. Это не могло не вызвать желания собрать свою собственную версию – бесплатную, но не уступающую по надежности. Как раз этой целью задался в 2002 году студент Уоррен Тагами. Он начал работу над проектом дистрибутива, представляющего собой единый репозиторий стабильных пакетов программ, при этом не относящийся напрямую к компании Red Hat. Таким образом появилась FedoraOS.

Проект быстро разросся при поддержке сообщества IT–специалистов и энтузиастов, разделяющих его философию. А в 2003 году Red Hat увидела в проекте перспективы для своих продуктов и взяла «Федору» под свое крыло, позволив сообществу продолжать развивать свое детище, не беспокоясь о финансах и претензиях со стороны компании [7].

Fedora Linux подразделяется на несколько видов. Каждый из низ предназначен для разных задач:

- fedora Workstation – это отточенная, легкая в использовании операционная система для переносных и настольных компьютеров с полным набором инструментов для разработчиков и производителей всех видов.
- fedora Server – мощная, гибкая операционная система, в которую вошли лучшие и самые новые технологии для центров обработки данных. Она дает вам контроль над всей пользовательской инфраструктурой и услугами.
- fedora IoT – предоставляет проверенную платформу с открытым исходным кодом в качестве надежной основы для экосистем IoT.
- fedora CoreOS – автоматически обновляемая, минимальная, ориентированная на контейнеры операционная система.
- fedora Labs – это набор тщательно подобранных пакетов спициализированного программного обеспечения и контента, отобранных и поддерживаемых членами сообщества Fedora. Они могут устанавливаться как самостоятельные версии Fedora или как дополнения к существующим установкам Fedora.

Рядовому пользователю предлагается использование именно Fedora Workstation. В качестве пакетного менеджера, с помощью которого в дистрибутив устанавливается ПО, предлагается DNF и Flatpak.

DNF – следующее поколение приложения Yum, менеджер пакетов для дистрибутивов ОС Linux на основе RPM–пакетов. DNF разрабатывался с 2011 года и был представлен в Fedora 18 и используется как основная система управления пакетами начиная с версии Fedora 22. Предыдущий YUM имел несколько недостатков, и DNF был призван их решить. Среди них: низкая производительность, высокое потребление памяти и низкая скорость итеративного разрешения зависимостей. DNF применил libsolv – внешний решатель зависимостей.

Flatpak (ранее известный как xdg–app) – это утилита для развёртывания, управления пакетами и виртуализации для Linux. Предоставляет собой песочницу, в которой пользователи могут запускать приложения без влияния на основную систему. Приложения, использующие Flatpak, требуют дополнительных разрешений на использование дискового пространства.

Fedora – дистрибутив, использующий только свободное программное обеспечение. В репозитории запрещено любое проприетарное программное обеспечение, хотя устанавливать утилиты с закрытым исходным кодом остается возможным, для этого придется использовать сторонние репозитории, например RPM Fusion.

1.5 Среда рабочего стола GNOME

Как было сказано ранее, дистрибутив Linux может включать в себя пакет для развёртывания среды рабочего стола (графические оболочки). Графическая оболочка Linux – это именно то, что пользователь видит в мониторе при использовании операционной системы.

Графические оболочки в Linux или, также можно сказать, графическое окружение Linux принято разделять на два типа, а именно: оконные менеджеры и среды рабочих столов.

Основное отличие оконного менеджера от среды состоит в том, что оконный менеджер отвечает лишь за отображение окон на экране, а среда рабочего стола, кроме функционала отображения окон, включает в себя целый набор приложений и компонентов. Так, среда рабочего стола включает как оконный менеджер, так и другие компоненты, необходимые пользователю для нормальной работы в операционной системе на базе Linux.

Одной из сред рабочего стола является GNOME (GNU Network Object Model Environment) – популярная среда рабочего стола для Linux. Включает в себя набор утилит для настройки среды, прикладное программное обеспечение, системные утилиты и другие компоненты. На рисунках 3 и 4 можно увидеть примеры рабочего окружения GNOME.

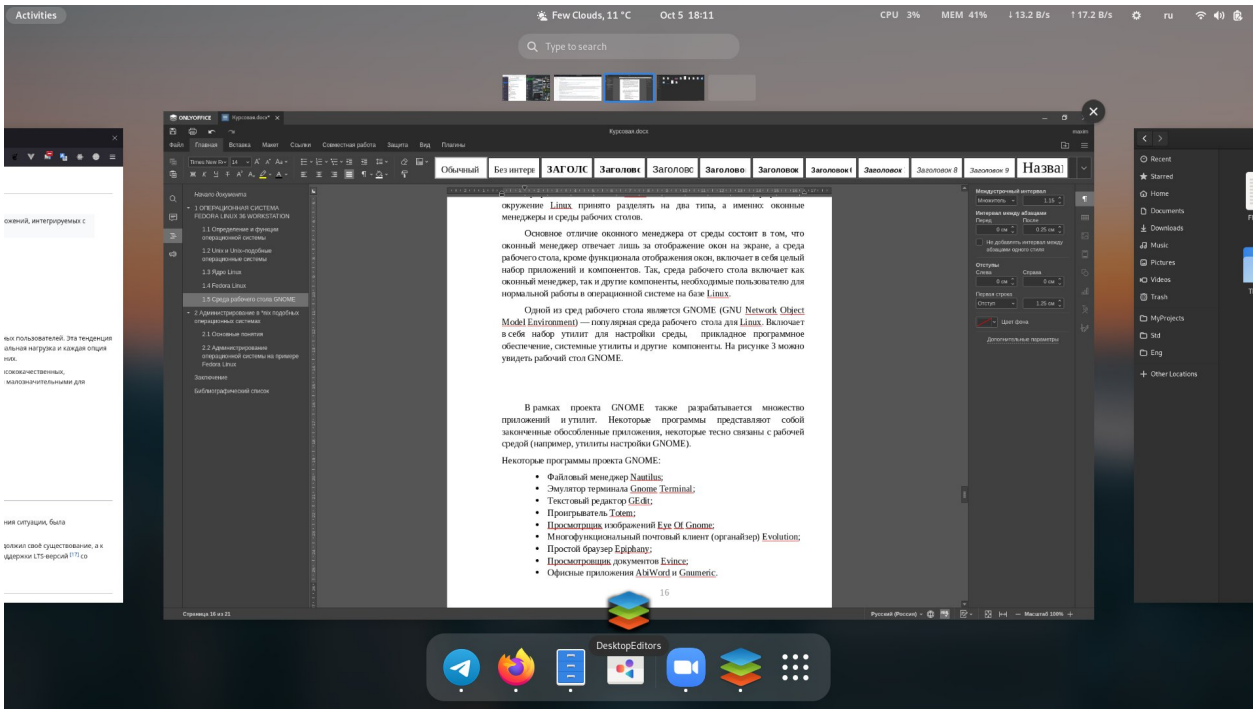

Рисунок 3 – Пример окружения рабочего стола GNOME

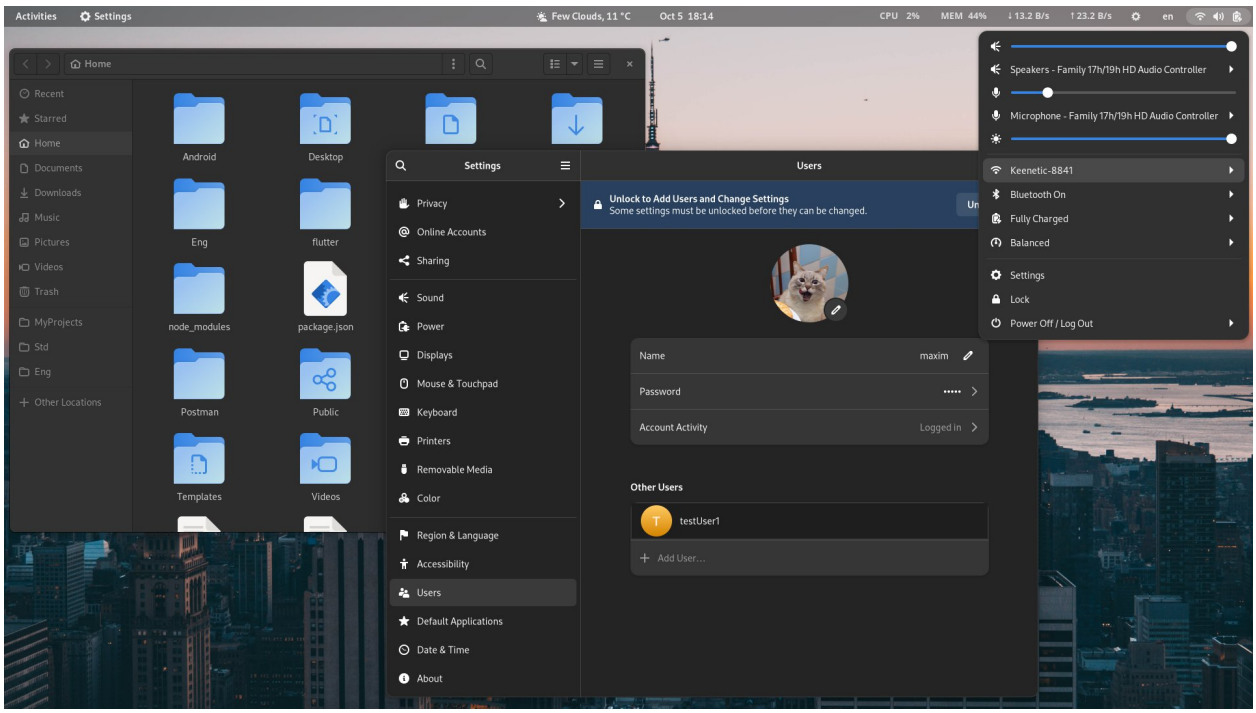

Рисунок 4 – Пример окружения рабочего стола GNOME

В рамках проекта GNOME также разрабатывается множество приложений и утилит. Некоторые программы представляют собой законченные обособленные приложения, некоторые тесно связаны с рабочей средой (например, утилиты настройки GNOME).

Некоторые программы проекта GNOME:

- файловый менеджер Nautilus;
- эмулятор терминала Gnome Terminal;
- текстовый редактор GEdit;
- проигрыватель Totem;
- просмотрщик изображений Eye Of Gnome;
- многофункциональный почтовый клиент (органайзер) Evolution;
- простой браузер Epiphany;
- просмотровщик документов Evince;
- офисные приложения AbiWord и Gnumeric.

2 Администрирование в Unix подобных операционных системах

2.1 Основные понятия

Пользователи и группы

UNIX – многопользовательская операционная система. Пользователи, занимающиеся общими задачами, могут объединяться в группы. Каждый пользователь обязательно принадлежит к одной или нескольким группам. Все команды выполняются от имени определенного пользователя, принадлежащего в момент выполнения к определенной группе.

В многопользовательских системах необходимо обеспечивать защиту объектов (файлов, процессов), принадлежащих одному пользователю, от всех остальных. ОС UNIX предлагает базовые средства защиты и совместного использования файлов на основе отслеживания пользователя и группы, владеющих файлом, трех уровней доступа (для пользователя–владельца, для пользователей группы–владельца, и для всех остальных пользователей) и трех базовых прав доступа к файлам (на чтение, на запись и на выполнение). Базовые средства защиты процессов основаны на отслеживании принадлежности процессов пользователям.

Для отслеживания владельцев процессов и файлов используются числовые идентификаторы. Идентификатор пользователя и группы – целое число (обычно) в диапазоне от 0 до 65535. Присвоение уникального идентификатора пользователя выполняется при заведении системным администратором нового регистрационного имени. Значения идентификатора пользователя и группы – не просто числа, которые идентифицируют пользователя, – они определяют владельцев файлов и процессов. Среди пользователей системы выделяется один пользователь – системный администратор или суперпользователь, обладающий всей полнотой прав на использование и конфигурирование системы. Это пользователь с идентификатором 0 и регистрационным именем root.

При представлении информации человеку удобнее использовать вместо соответствующих идентификаторов символьные имена – регистрационное имя пользователя и имя группы. Соответствие идентификаторов и символьных имен, а также другая информация о пользователях и группах в системе (учетные записи), как и большинство другой информации о конфигурации системы UNIX, по традиции, представлена в виде текстовых файлов. Эти файлы – /etc/passwd, /etc/group и /etc/shadow.

Файлы и каталоги

Операционная система выполняет две основные задачи: манипулирование данными и их хранение. Большинство программ в основном манипулирует данными, но, в конечном счете, они где–нибудь хранятся. В системе UNIX таким местом хранения является файловая система. Более того, в UNIX все устройства, с которыми работает операционная система, также представлены в виде специальных файлов в файловой системе.

Логическая файловая система в ОС UNIX (или просто файловая система) – это иерархически организованная структура всех каталогов и файлов в системе, начинающаяся с корневого каталога. Файловая система UNIX обеспечивает унифицированный интерфейс доступа к данным, расположенным на различных носителях, и к периферийным устройствам. Логическая файловая система может состоять из одной или нескольких физических файловых (под)систем, являющихся разделами физических носителей (дисков, CD–ROM или дискет).

Файловая система контролирует права доступа к файлам, выполняет операции создания и удаления файлов, а также выполняет запись/чтение данных файла. Поскольку большинство прикладных функций выполняется через интерфейс файловой системы, следовательно, права доступа к файлам определяют привилегии пользователя в системе.

Файловая система обеспечивает перенаправление запросов, адресованных периферийным устройствам, соответствующим модулям подсистемы ввода–вывода [8].

2.2 Администрирование операционной системы на примере Fedora Linux

На примере Fedora Workstation 36 рассмотрим администрирование операционной системы на базе ядра Linux.

Для начала выведем список все содержимого корневой директории системы с подробной информацией. Для этого введем в команду ls. Также воспользуемся следующими ключами:

- l – вывод подробной информации о файле (тип файла, права доступа, кол–во ссылок, пользователь, которому принадлежит этот файл, группа пользователей, которая имеет доступ к этому файлу, вес файла (байт), время последнего изменения, название);
- a – вывод скрытых файлов в директории (скрытыми файлами считаются те, что начитаются с символа ‘.’).

Результат работы команды представлен на рисунке 5.

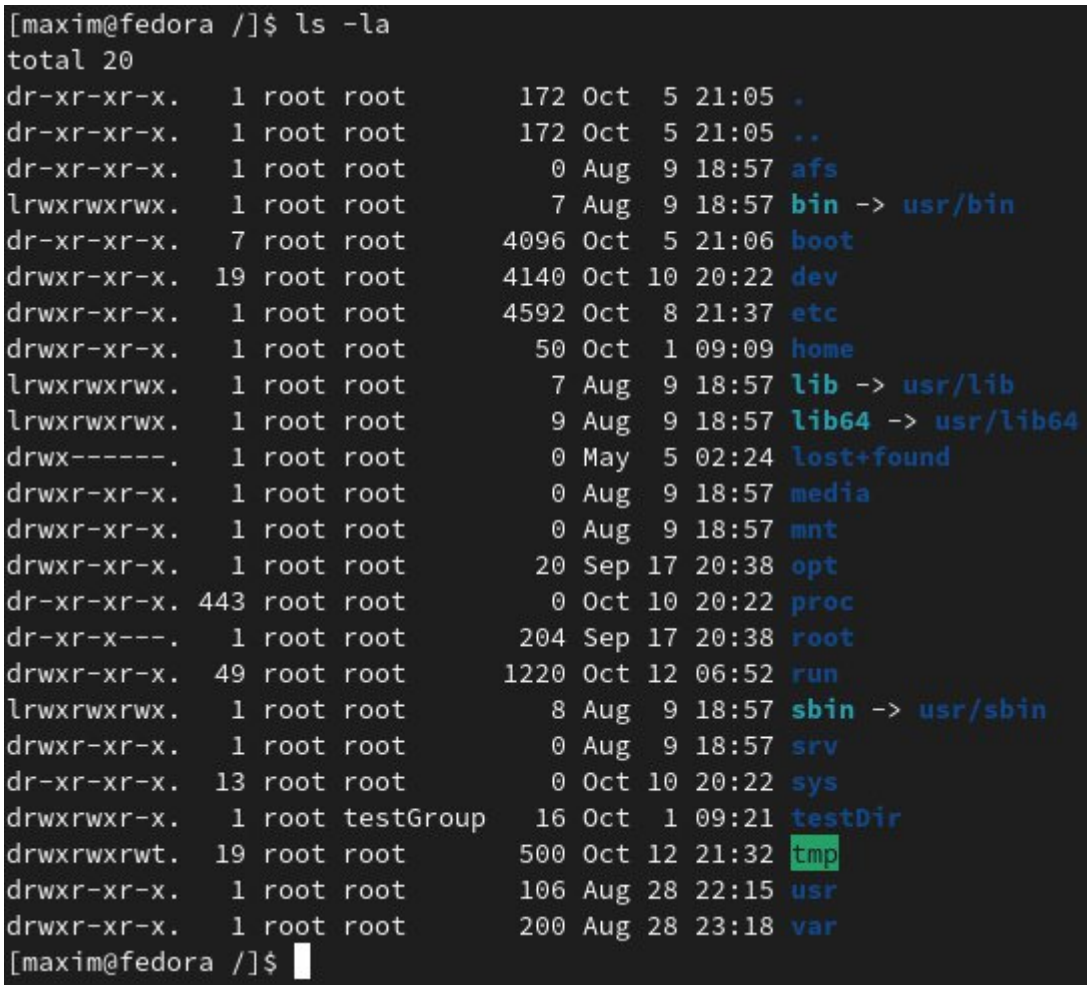

Рисунок 5 – Результат работы команды ls –la

Чтобы увидеть список файлов определенного типа (обычной файл, директория, FIFO и т.д.) можно воспользоваться командой find со ключем type, который на вход принимает следующие значения:

- f – обычный файл;
- d – директория;
- l – символическая ссылка;
- p – файл ввода–вывода;
- b – блочное устройство;
- s – сокет.

Найдем список символических ссылок в директории /bin. Для это воспользуемся командой cd (сменить директорию), а также командой find, описанной выше. Результат выполнения команд представлен на рисунке 6.

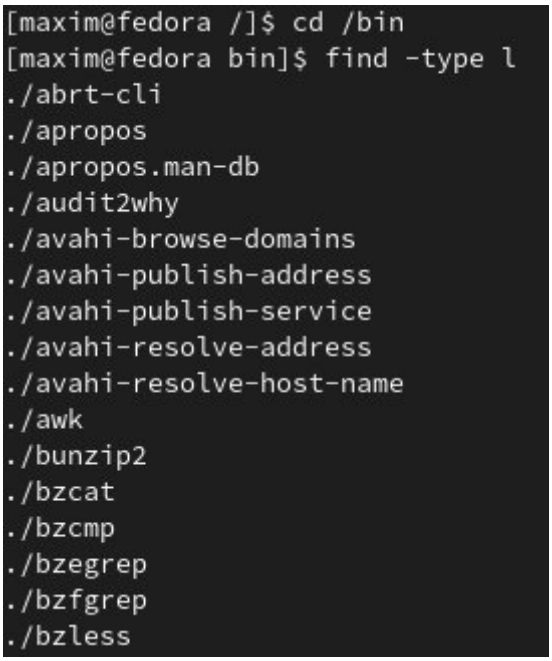

Рисунок 6 – Результат работы команд cd и find –type l

Перейдем в заранее подготовленную директорию и создадим в ней файл, мягку и жесткую ссылку на этот файл и директорию. Чтобы создать файл, используем команду touch, на вход которой передадим название файла “test_file.txt„.

За создание ссылок отвечает команда ln, которая принимает название файла, а вторым аргументом – название ссылки. Чтобы ссылка была мягкой (символической), передадим команде ключ s.

Для создания директории используем команду mkdir, которой передадим название директории “test_dir„. Результат работы команд представлен на рисунке 7.

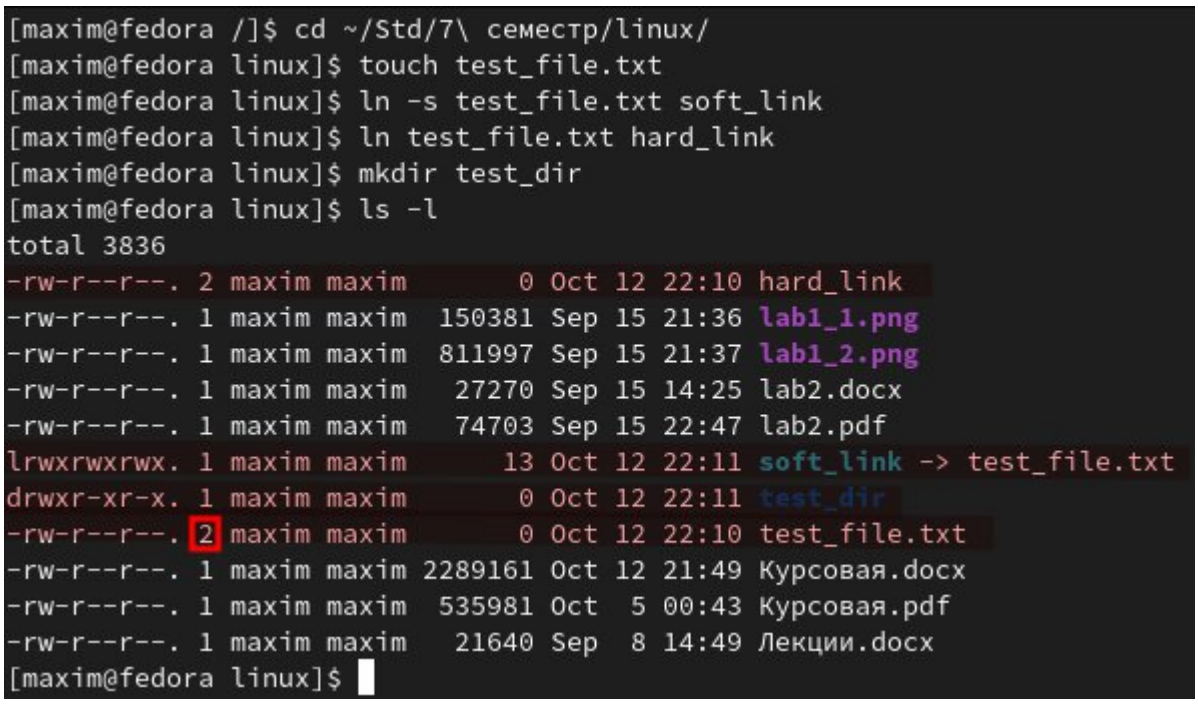

Рисунок 7 – Результат работы команд touch, ln, mkdir

В результате можно увидеть, что в директории было создано 4 файла, которые различаются типом. А также, можно наблюдать, что у файла test_file.txt увеличилось количество ссылок на него.

Внесем изменения в файл test_file.txt, напишем в него несколько строк текста. После чего, выведем содержимое файла в терминал. Для редактирования файла воспользуемся консольным текстовым редактором vi, на вход которому передадим название файла. Чтобы вывести содержимое файла, используем команду cat. Результат работы команд можно увидеть на рисунках 8 и 9.

Рисунок 8 – Окно консольного текстового редактора vi

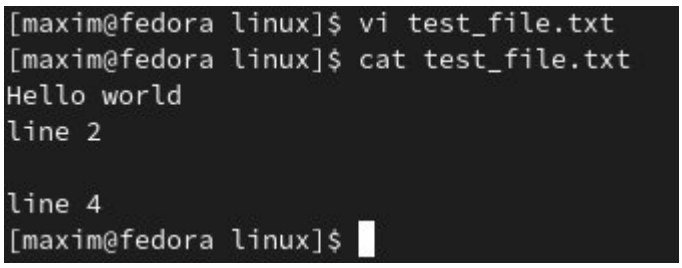

Рисунок 9 – Результат работы команд vi и cat

Для создания пользователя и группы существует несколько способов. Первым способом является использование соответствующих команд useradd и groupadd. Вторым же способом является редактирование файлов /etc/passwd и /etc/group. Для создания группы изменим файл /etc/group, а для создания пользователя воспользуемся командой useradd.

Для редактирования файла воспользуемся уже знакомой утилитой vi. Открыв данный файл можно увидеть список всех групп, которые были созданы в данной системе. Для создания собственной группы допишем в конец файла строку, состоящую из полей, разделенных символом ‘:’. В качестве полей используются:

- название группы;
- зарезервированное поле, поставим символ ‘x’;
- числовой идентификатор группы;
- список участников группы, разделенный символом ‘,’.

Результат редактирования файла представлен на рисунке 10. В качестве участников группы были прописаны пользователи maxim и testUser, который, далее, будет создан командой userAdd.

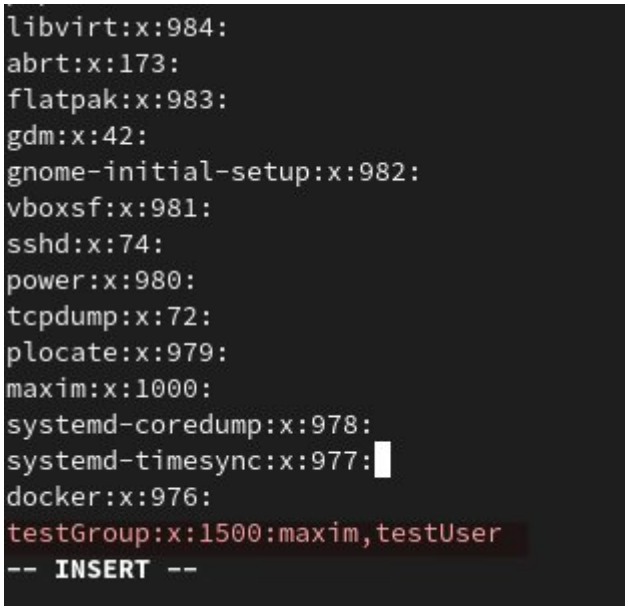

Рисунок 10 – Создание группы с помощью файла /etc/group

Для создания пользователя, как было сказано раннее, воспользуемся командой useradd. В качестве ключей будем использовать следующее:

- m – Создает домашнюю директорию пользователя в каталоге /home;
- S – Устанавливает оболочку пользователя, укажем /bin/bash
- G – Устанавливает группу, к которой принадлежит пользователь, укажем раннее созданную группу testGroup с id = 1500.

В качестве имени пользователя укажем testUser. Результат работы команд можно увидеть на рисунке 11

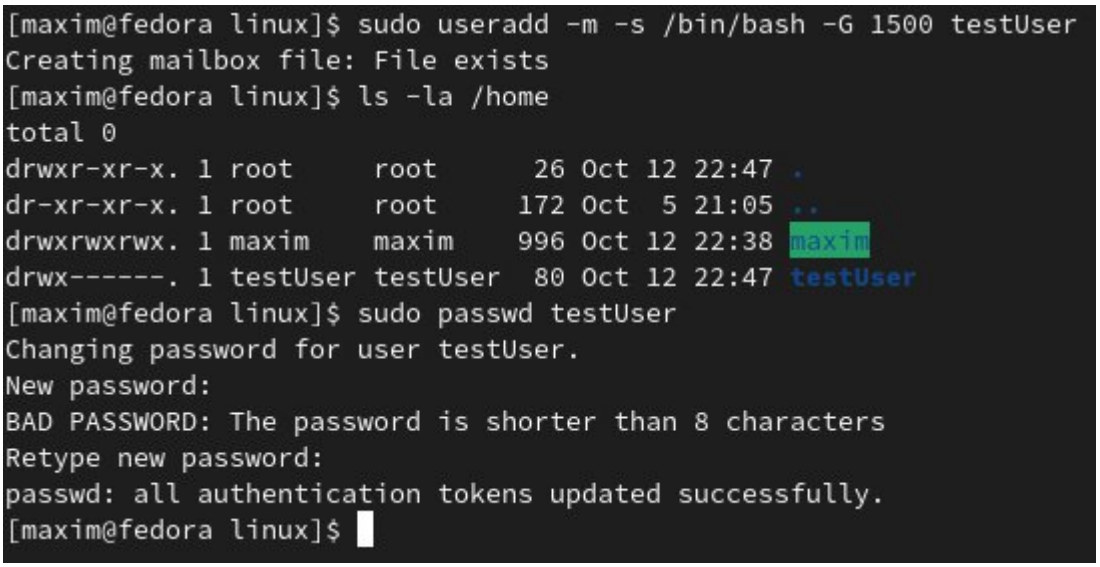

Рисунок 11 – Результат создания пользователя с помощью команды useradd

Как видно на рисунке 11, пользователь был успешно создан. Также, в директории /home появился домашний каталог созданного пользователя. Помимо этого, с помощью команды passwd, мы установили пароль для данного пользователя.

Также, на рисунке 12 можно увидеть, что в файле /etc/passwd появилась запись о созданном пользователе.

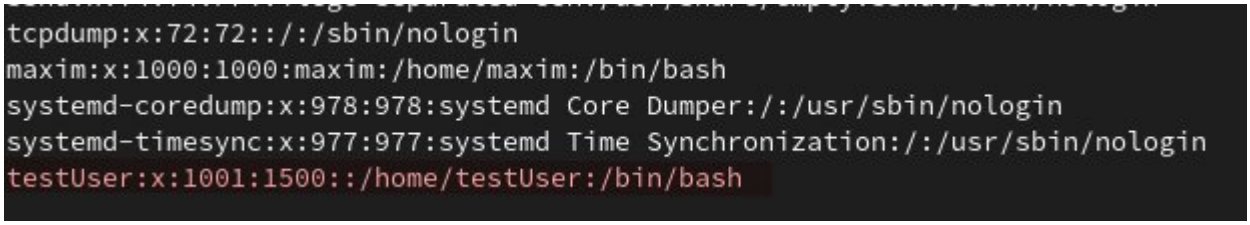

Рисунок 12 – Запись пользователя testUser в файле /etc/passwd

Теперь выдадим определенные права пользователю testUser, на созданные раннее файл и директорию.

Для начала, у файла test_file.txt и директории test_dir поменяем группу. Это можно сделать с помощью команды chgrp. Первым параметром данная утилита принимает название группы, вторым параметром – название файла.

Сменим группу на ранее созданную testGroup, в которую входит пользователь testUser. Результат работы команд можно увидеть на рисунке 13.

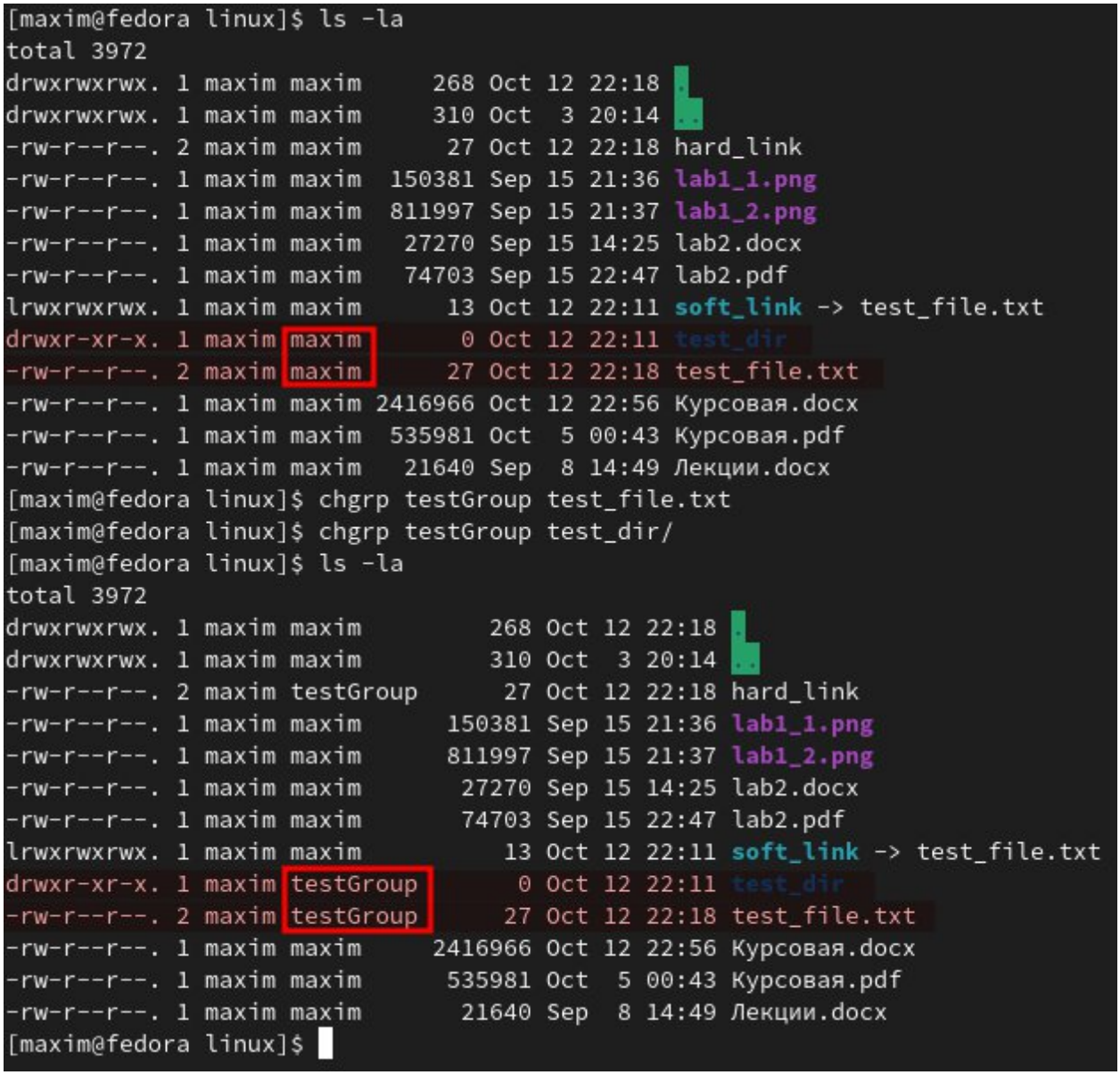

Рисунок 13 – Результат работы команды testGroup

Теперь, для файла test_file.txt, для группы выдадим права только на чтение. А для директории, только на запись и исполнение. Для этого воспользуемся командой chmod, которая на вход принимает строку в определенном формате для определения прав, а также название файла. На рисунке 14 можно увидеть результат работы команд.

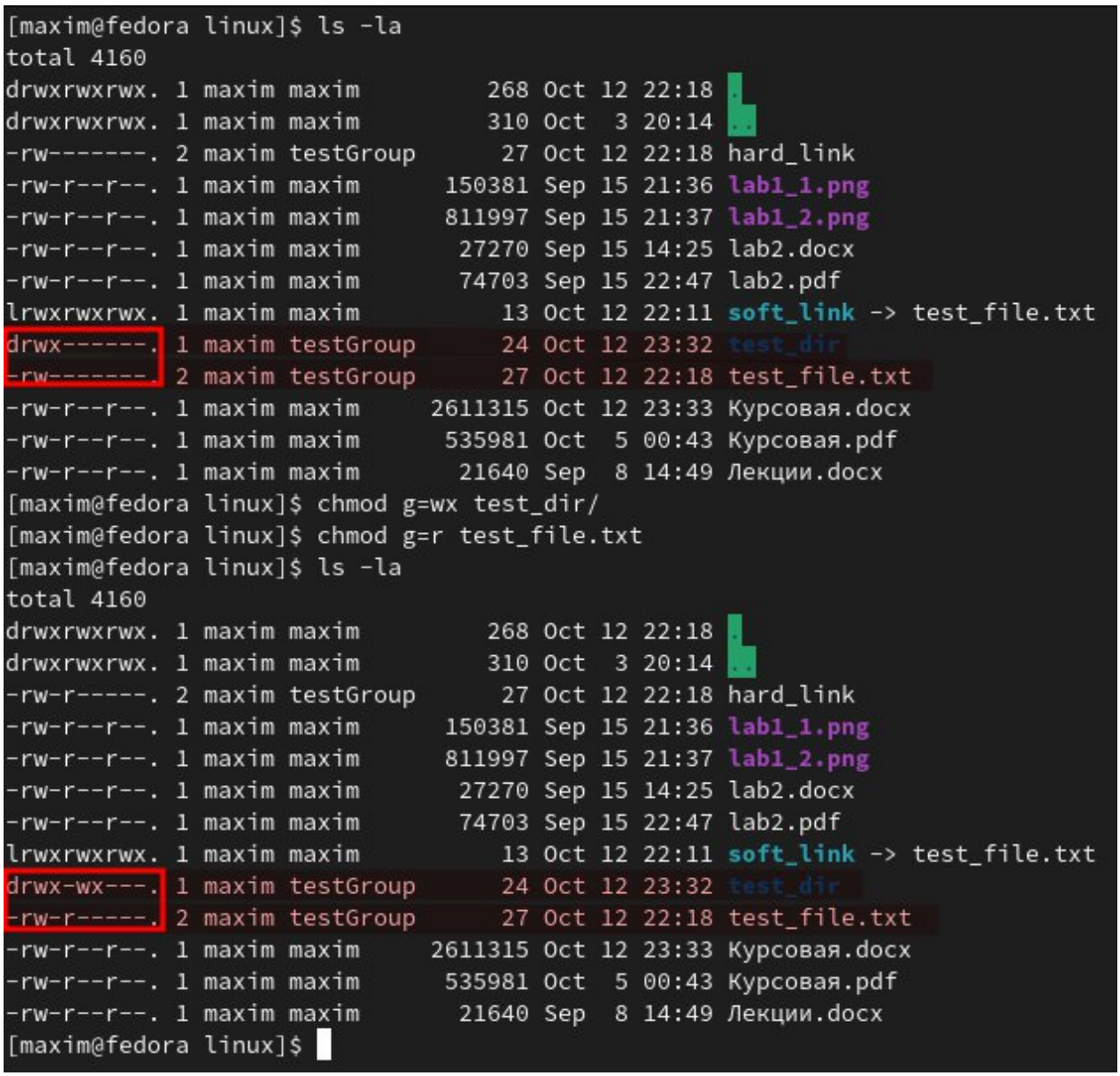

Рисунок 14 – Результат работы команды chmod

После выдачи прав, зайдем в систему под пользователем testUser (команда su) и проверим результат (рис. 15, рис.16).

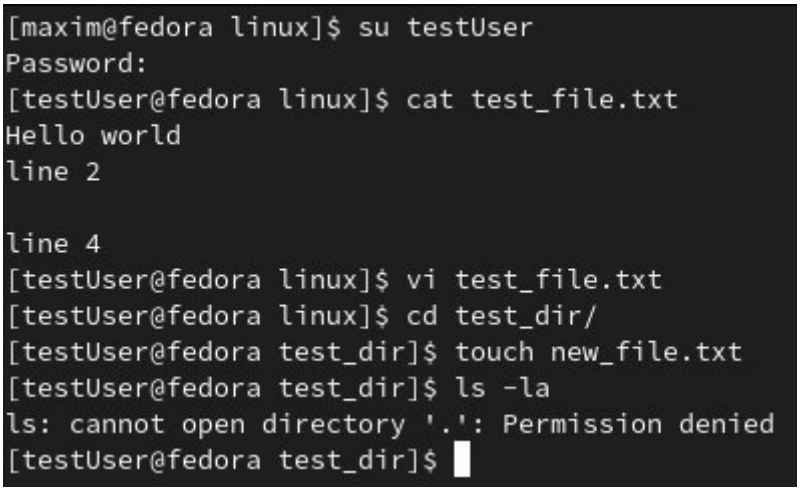

Рисунок 16 – Проверка доступа к фалу test_file.txt и директории test_dir

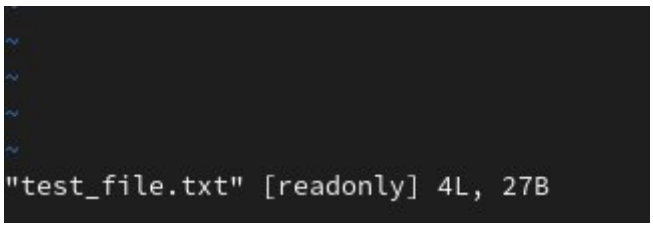

Рисунок 17 – Редактирование файла test_file.txt

На рисунках, представленных выше, мы можем наблюдать следующие возможности пользователя testUser:

- может видеть (команда cat) содержимое файла test_file.txt;
- не может менять (команда vi) содержимое файла test_file.txt;
- может создавать файлы и вставать (команды touch и cd) в директорию test_dir;
- не может просматривать (команда ls) содержимое директории test_dir.

Заключение

В результате выполнения курсовой работы была рассмотрена краткая история операционных систем и выделены основные требования к ним:

- надежность и отказоустойчивость;
- эффективность;
- предсказуемость;
- расширяемость;
- переносимость;
- удобство;
- масштабируемость.

Мы рассмотрели пример реализации некоторых требований на примере ОС Unix. Сравнили данную систему с ядром Linux, выявили их различия и просмотрели особенности строения, которые были перенесены из Unix в Linux.

Была выполнена установка и настройка Unix–подобной операционной системы на базе ядра Linux, Fedora Workstation 36. Были рассмотрены особенности данной операционной системы. Также, мы познакомились со средой графического окружения GNOME и базовыми программным обеспечением, которое входит в поставку с данным окружением.

На примере данной операционной системы познакомились с администрированием в Unix–подобных ОС. Были проделаны следующие действия:

- просмотр содержимого директории;
- создание файла и директории;
- редактирование файла с помощью консольного текстового редактора vi;
- создание пользователя и группы различными способами;
- настройка пользователя;
- выдача прав пользователю и группе;
- настройка доступа к файлам и директориям.

Библиографический список

1. История электронных компьютеров. [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/408597/

2. Операционная система. [Электронный ресурс] – Режим доступа: https://blog.skillfactory.ru/glossary/operaczionnaya–sistema/

3. Основные функции и виды операционных систем. [Электронный ресурс] – Режим доступа: http://dit.isuct.ru/IVT/sitanov/Literatura/InformLes/Pages/Glava4_2.htm

4. Что такое UNIX. [Электронный ресурс] – Режим доступа: https://thecode.media/Unix/

5. Что такое ядро Linux. [Электронный ресурс] – Режим доступа: https://losst.ru/chto–takoe–yadro–linux

6. Что такое дистрибутив Linux. [Электронный ресурс] – Режим доступа: https://losst.ru/chto–takoe–distributiv–linux

7. Дистрибутив Fedora Linux. [Электронный ресурс] – Режим доступа: https://selectel.ru/blog/fedora–linux/

8. Основы операционной системы UNIX. [Электронный ресурс] – Режим доступа: https://www.opennet.ru/docs/RUS/unix_basic/