МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМЕНИ МУХАММАД АЛ-ХОРАЗМИЙ

ФАКУЛЬТЕТ ТЕЛЕКОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

Кафедра Аппаратное и программное обеспечение систем управления в телекоммуникации

Х.Х.Ахмедова, Ф.К.Тожиева, Б.У.Акмурадов

ОПЕРАЦИОННЫЕ СИСТЕМЫ

МЕТОДИЧЕСКОЕ ПОСОБИЕ

по выполнению лабораторных работ для студентов направления 5350100 – Телекоммуникационные технологии ("Телекоммуникации", "Телерадиовещание", "Мобильные системы").

Ташкент 2021

Авторы: Х.Х.Ахмедова, Ф.К.Тожиева, Б.У.Акмурадов

Методическое пособие по выполнению лабораторных работ по предмету " Операционные системы" -Ташкент: ТУИТ. 2021. - 80 стр.

В методическом пособии рассматривается установка, настройка и загрузка операционной системы Ubuntu, управление сетью, файлами, внешними устройствами в операционной системе, настройка почтовой почтовой службой и сервисом, а также администрирование удаленных систем. Даны необходимые справочные данные, приведены примеры диалоговых процедур, даны задания и вопросы для выполнения лабораторных работ и приведены пояснения, необходимые для выполнения лабораторных работ.

Методическое пособие предназначено для студентов направления 5350100 – Телекоммуникационные технологии ("Телекоммуникации", "Телерадиовещание", "Мобильные системы").

Решением научно-методического совета Ташкентского университета информационных технологий имени Мухаммада ал-Хоразмий методическое пособие рекомендуется для публикации. ("___" - протоколом "___" _____ 2021 год).

Ташкентский университет информационных технологий имени Мухаммада Ал-Хоразмий,

2021

ВВЕДЕНИЕ

Эволюция сетей телекоммуникаций включает в себя закономерный переход к цифровым сетям, происходивший в нашей Республике поэтапно. Развитие сферы ИКТ невозможно без активной научной и инновационной деятельности. Известно, что исследование проблем информатизации, разработка технологий передачи, обработки и защиты информации, создания интеллектуальных систем является одним из приоритетных направлений развития науки и технологий.

Технологическое развитие учитывает мировые тенденции развития отрасли связи и ИКТ и основывается на опережающем подходе в выборе технологий. В обозначенный период предусматривается повысит производительность, отказоустойчивость и защищенность телекоммуникационной сети республики сети за счет внедрения новейших технологий, с учетом текущих мировых тенденций, которая будет основываться на постепенной миграции всех сегментов сети на IP платформы.

В методическом пособии представлены инструкции по выполнению лабораторных работ по предмету «Операционные системы». Оно включает название лабораторной работы, цель работы, контрольные работы и задания. Изложены основные принципы организации современных операционных систем на примере ОС Linux.

В методическом пособии рассматривается установка, настройка и загрузка операционной системы Ubuntu, управление сетью, файлами, внешними устройствами в операционной системе, настройка почтовой почтовой службой и сервисом, а также администрирование удаленных систем. Даны необходимые справочные данные, приведены примеры диалоговых процедур, даны задания и вопросы для выполнения лабораторных работ и приведены пояснения, необходимые для выполнения лабораторных работ.

ЛАБОРАТОРНАЯ РАБОТА № 1

Тема: Установка операционной системы

Цель работы:

Ознакомиться с операционной системой Linux и изучить установку Linux Ubuntu.

Задание:

Освоить краткие теоретические сведения и пошагово выполнить установку операционной системы Linux .

Теоретическая часть:

Операционная система Linux набирает популярность, и она интересна даже обычным, т.е. не продвинутым пользователям компьютера, поэтому сегодня специально для начинающих мы подробно рассмотрим процесс установки на компьютер операционной системы Linux на примере популярного дистрибутива Ubuntu.

Linux – это бесплатная операционная система с открытым исходным кодом (если быть точнее, Linux – это ядро операционной системы). Поэтому любой разработчик может абсолютно свободно создать свою операционную систему на базе Linux, и такая система будет называться дистрибутивом Linux.

В связи с этим существует много различных дистрибутивов Linux, как удобных и популярных, так и менее популярных, для продвинутых пользователей.

Linux Ubuntu – это один из самых популярных дистрибутивов операционной системы Linux, который отлично подходит для домашнего компьютера. Ubuntu имеет отличную локализацию, с поддержкой русского языка, огромное сообщество, т.е. Вы легко можете найти любую интересующую Вас информацию об этом дистрибутиве.

Системные требования Linux Ubuntu

В Ubuntu используется среда рабочего стола GNOME, она красивая, современная и функциональная, поэтому Linux Ubuntu не входит в число легковесных дистрибутивов, которые можно использовать

на «слабом» оборудовании. У Ubuntu, по сравнению с другими дистрибутивами, достаточно серьезные системные требования, однако для современных ПК — это не проблема. Требования следующие:

- Двухъядерный процессор 2 ГГц или выше;
- Оперативной памяти 2 ГБ или больше;
- Рекомендовано 25 ГБ свободного места на жестком диске.

Порядок выполнения работы:

Шаг 1 – Скачивание установочного образа Linux Ubuntu

Практически все дистрибутивы Linux распространяются в виде ISO образов дисков, поэтому для того чтобы установить Linux Ubuntu, сначала необходимо скачать установочный ISO файл.

Шаг 2 – Запись установочного ISO образа на диск или флешку

После того как Вы загрузили установочный ISO образ Linux Ubuntu, его необходимо записать на DVD диск или USB флешку, для того чтобы создать загрузочный установочный носитель, с которого и будет производиться установка. Для этого существует много различных программ

В Linux:

- Для записи на USB флешку: Etcher или стандартная программа «Запись образа на USB-накопитель»;
- Для записи на диск: k3b или Brasero.

Шаг 3 – Загрузка с установочного носителя и запуск программы установки

Установочный носитель с Linux Ubuntu подготовили, теперь можно переходить к процессу установки. Первое, что нужно сделать – это загрузиться с этого установочного носителя, т.е. с диска или USB флешки. Для этого в BIOS необходимо выставить данный носитель на первое место в порядке загрузки устройств. Когда Вы загрузитесь с носителя, Вам сразу предложат выбрать язык, выбираем нужный и нажимаем ввод (Enter).

После этого откроется меню. Для установки на жесткий диск компьютера необходимо нажать на пункт «Установить Ubuntu». Если Вы хотите запустить Linux Ubuntu без установки, в режиме Live, например, для

того чтобы протестировать систему или просто посмотреть на нее, то нажимайте на пункт «Запустить Ubuntu без установки».

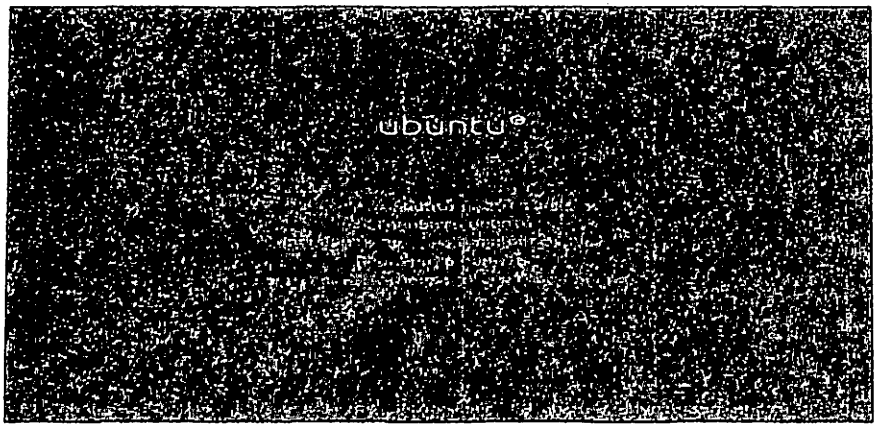

Рисунок 1.1. Установка Ubuntu

Шаг 4 – Выбор языка

Программа установки Linux Ubuntu запустится. Сначала нам нужно выбрать язык системы, по умолчанию выбран тот, который Вы выбрали, когда только загрузились с носителя. Нажимаем «Продолжить».

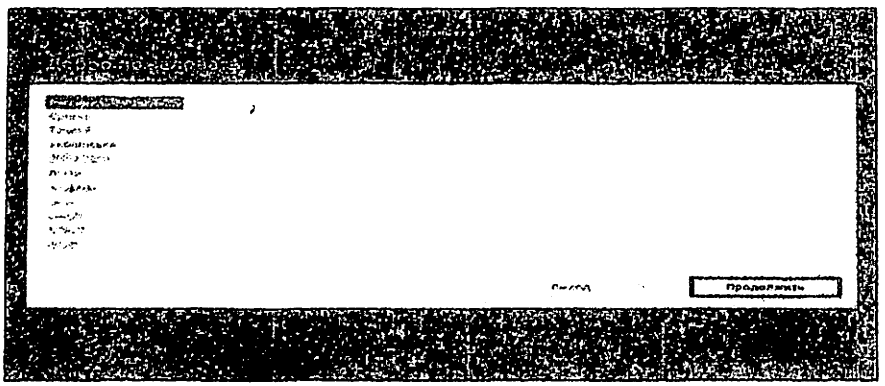

Рисунок 1.2. Выбор языка

Шаг 5 – Выбор раскладки клавиатуры

На этом шаге выбираем раскладку клавиатуры, по умолчанию она уже выбрана. Нажимаем «Продолжить».

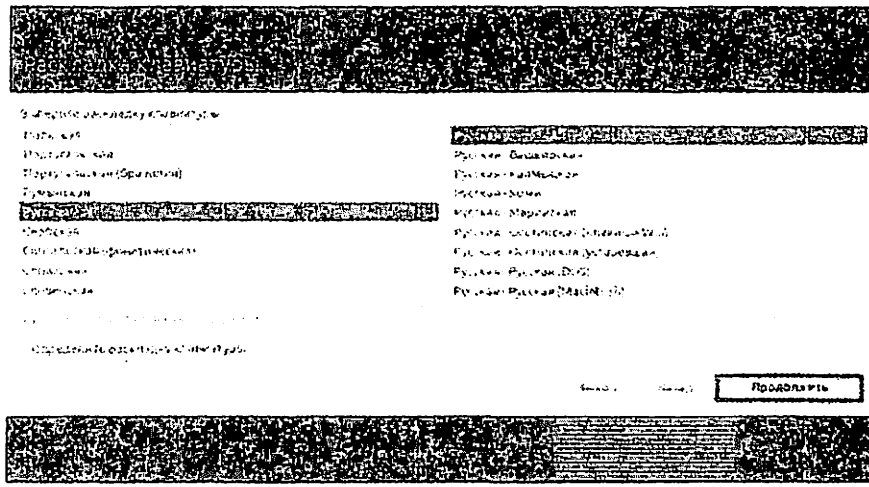

Рисунок 1.3. Выбор раскладки клавиатуры.

Шаг 6 – Параметры установки приложений и обновлений

Теперь нам необходимо выбрать приложения, которые мы хотим установить, для этого мы выбираем режим установки программного обеспечения:

- Обычная установка – это установка системы со стандартным набором приложений. Рекомендована обычным пользователям ПК, так как в данном случае будут автоматически установлены все необходимые для работы программы;
- Минимальная установка – это установка системы с минимальным набором приложений. Данный режим Вы можете использовать, если Вам нужна чистая система только с основными утилитами, все приложения Вы будете устанавливать самостоятельно. Режим для тех, кто любит настраивать систему под себя, т.е. устанавливать только те приложения, которые ему нужны.

Оставляем по умолчанию, т.е. «Обычная установка», также галочку «Загрузить обновления во время установки Ubuntu» тоже лучше оставить, нажимаем «Продолжить».

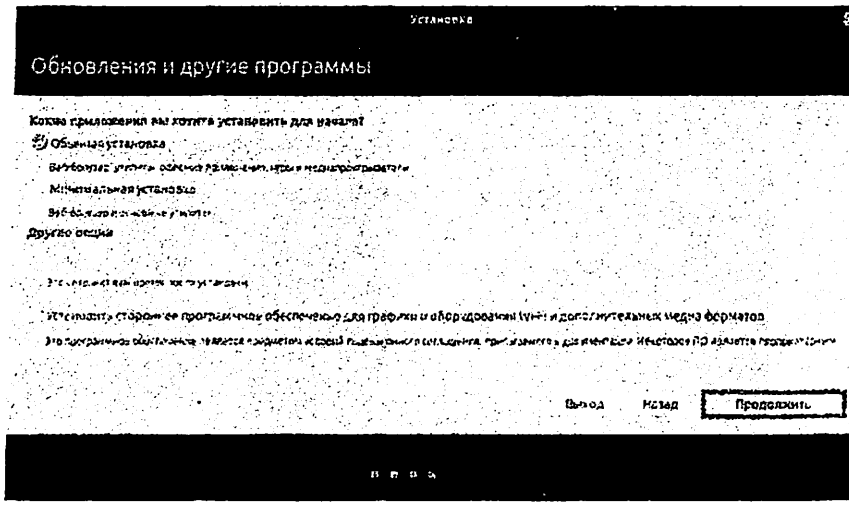

Рисунок 1.4. Параметры установки приложений и обновлений.

Шаг 7 – Разметка жесткого диска в Ubuntu

Затем нам нужно разметить жесткий диск, иными словами, создать необходимые разделы. Все необходимые операции мы будем делать вручную, т.е. Вы сами сможете создать разделы и указать их размер, в этом нет ничего сложного. Для этого необходимо нажать пункт «Другой вариант».

Необходимо выбрать пункт – «Стереть диск и установить Ubuntu».

В случае если у Вас уже установлена какая-нибудь система, программа установки может предложить Вам еще несколько вариантов, например, «Переустановить систему», «Удалить систему и совершить переустановку», а также «Установить Ubuntu рядом с уже установленной системой» (в этом случае при включении компьютера Вы будете выбирать, какая система должна загрузиться).

Выбираем пункт «Другой вариант» и нажимаем «Продолжить».

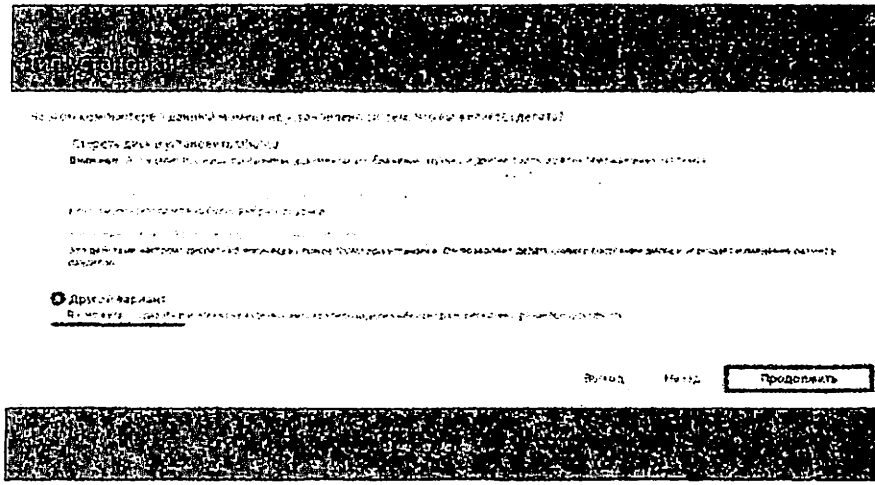

Рисунок 1.5. Создание самостоятельно разделов в Ubuntu

На чистом жестком диске нет таблицы разделов, поэтому нам ее необходимо создать. Нажимаем кнопку «Новая таблица разделов».

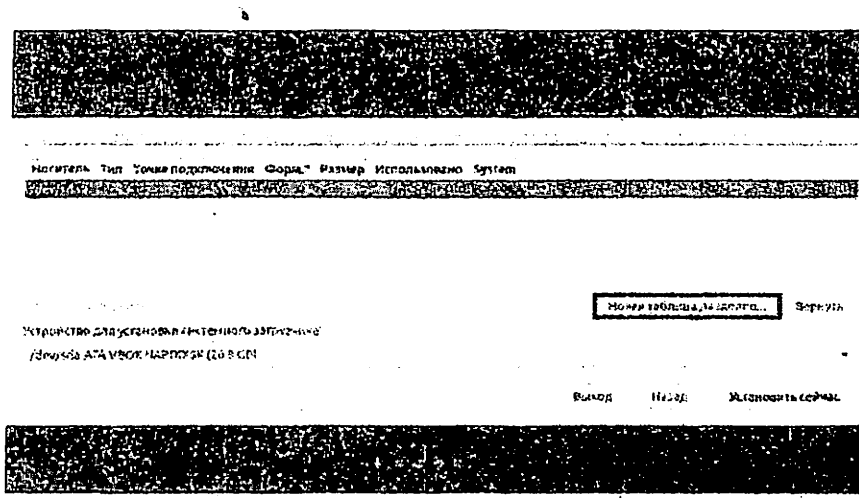

Рисунок 1.6. Создание таблицы разделов в Ubuntu

Программа установки предупреждает нас о том, что будет создана новая таблица разделов, и все существующие разделы на этом диске будут удалены, в нашем случае (чистый жёсткий диск) разделов просто нет, поэтому нажимаем «Продолжить».

Затем, для того чтобы создать новый раздел, выбираем «Свободное место» и нажимаем плюсик.

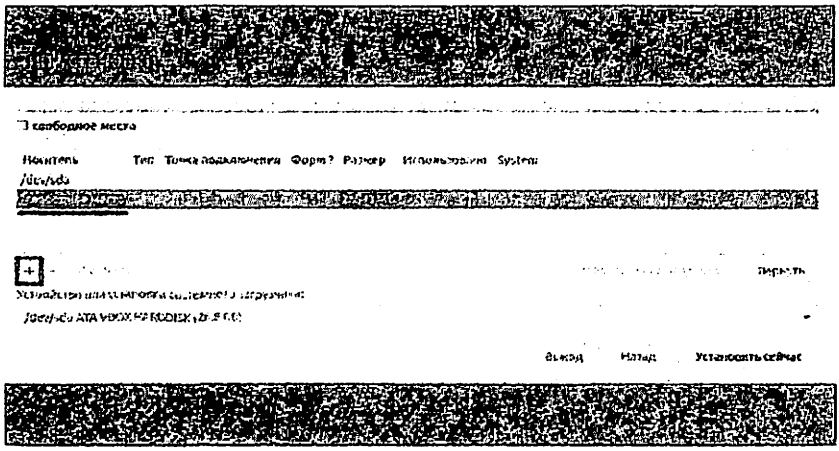

Рисунок 1.7. Создание нового раздела в Ubuntu

Сначала нам необходимо создать системный раздел (корневой раздел) для самой системы. Для этого указываем следующие данные:

- Размер – для корневого раздела нужно указывать минимум 10-15 гигабайт, но лучше указывать больше, например, 50 гигабайт
- Тип нового раздела – указываем «Первичный»;
- Местоположение нового раздела – указываем «Начало этого пространства»;
- Использовать как – выбираем журналируемая файловая система Ext4, данная файловая система лучше всего подходит для корневого раздела;
- Точка монтирования – для корневого раздела указываем «/».

Нажимаем «ОК».

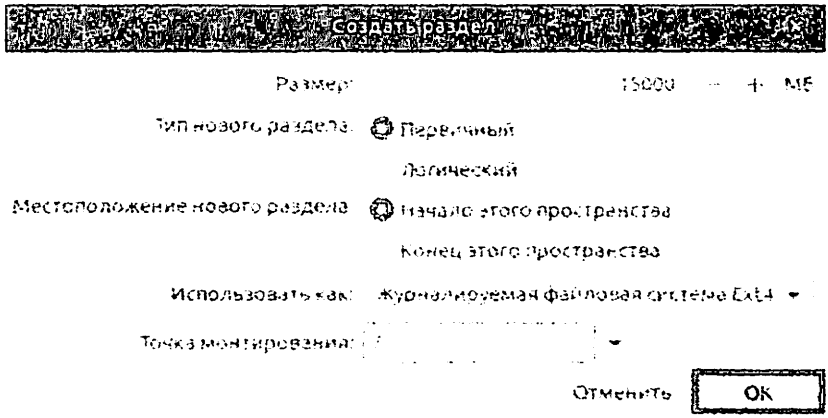

Рисунок 1.8. Настройки нового системного раздела в Ubuntu

Системный раздел создан, теперь нам нужно создать раздел для пользовательских данных, т.е. «Домашний раздел». Он необходим для того, чтобы в случае переустановки системы или даже смены дистрибутива Linux все наши личные данные (документы, фото, видео) остались на месте.

В данном случае точно также выбираем свободное место и нажимаем на плюсик.

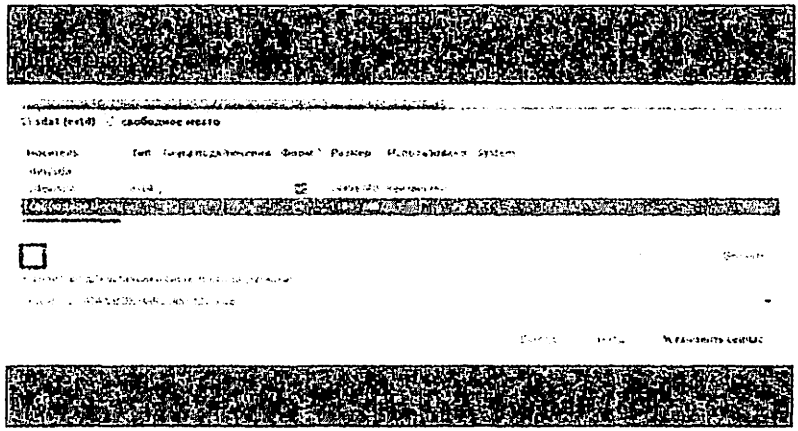

Рисунок 1.9. Настройки нового раздела в Ubuntu

Для создания домашнего раздела необходимо указать:

- Размер – по возможности максимальный, иными словами, можете указать все оставшееся место;
- Тип нового раздела – указываем «Логический»;
- Местоположение нового раздела – указываем «Начало этого пространства»;
- Использовать как – также выбираем файловую систему Ext4;
- Точка монтирования – указываем «/home».

Нажимаем «ОК».

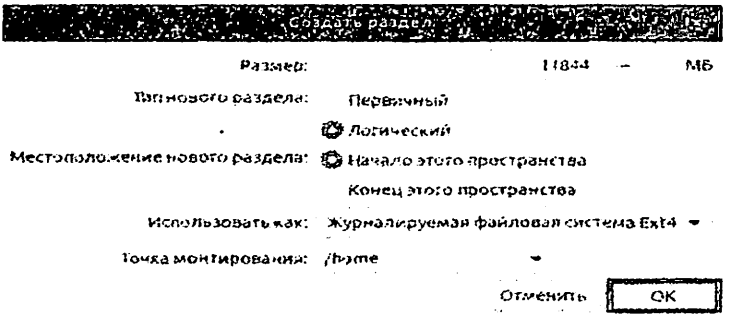

Рисунок 1.10. Настройки нового домашнего раздела в Ubuntu

Разметку жесткого диска в Linux Ubuntu мы выполнили , теперь можно запускать установку дистрибутива, нажимаем «Установить сейчас».

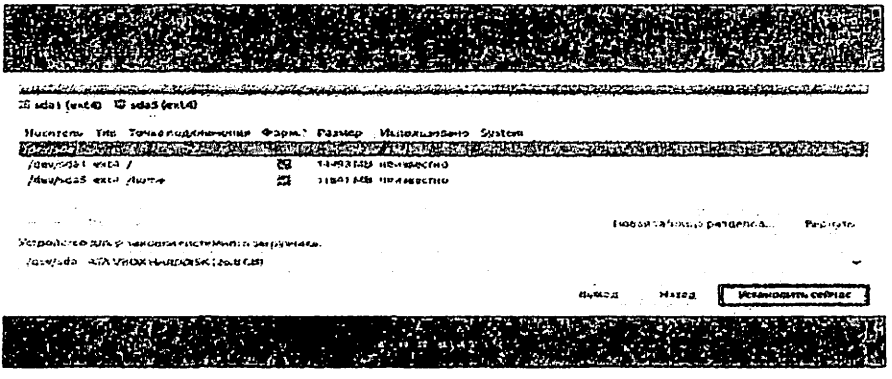

Рисунок 1.11. Установка дистрибутива

Далее, программа установки Ubuntu спросит у нас, хотим ли мы записать все внесенные изменения на диск, так как до этого момента все можно отменить, нажимаем «Продолжить».

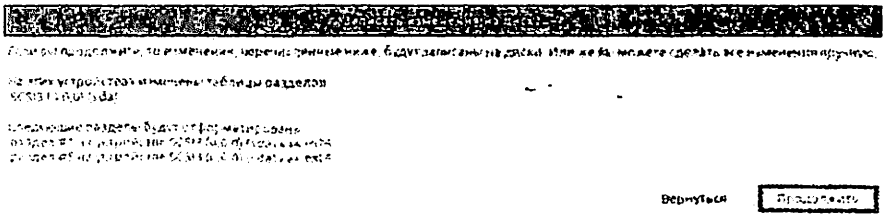

Рисунок 1.12. Утверждение изменения на диске

Шаг 8 – Выбор часового пояса

После этого нам нужно выбрать часовой пояс, выбираем и нажимаем «Продолжить».

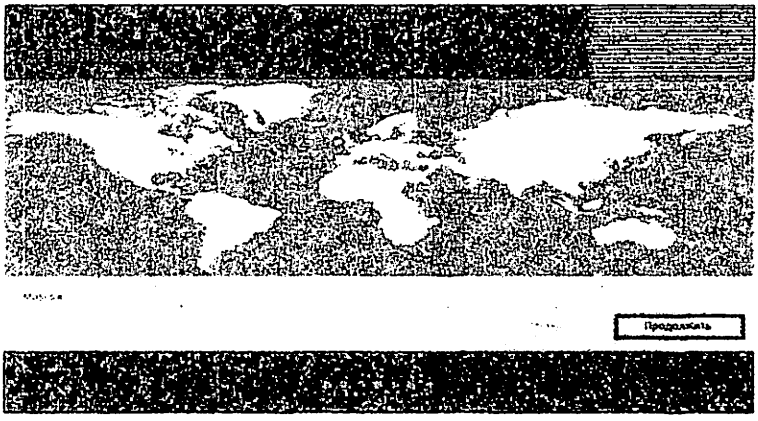

Рисунок 1.13. Выбор часового пояса

Шаг 9 – Создание пользователя

Теперь нам нужно создать учетную запись, т.е. пользователя, под которым мы будем работать. Вводим имя, имя компьютера, логин, пароль и подтверждаем его. В целях безопасности рекомендую пункт «Требовать

пароль для входа в систему» оставить включенным. Если Вы не хотите каждый раз при входе в систему вводить пароль, то можете отметить пункт «Входить в систему автоматически»

Вводим данные и нажимаем «Продолжить».

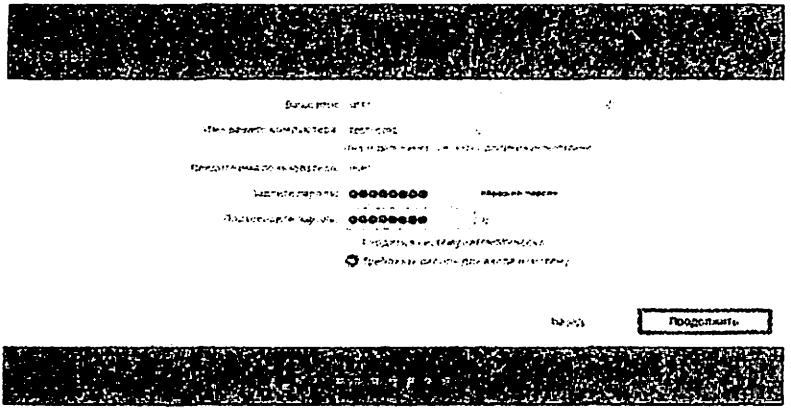

Рисунок 1.14. Создание пользователя в Ubuntu

Установка Linux Ubuntu началась.

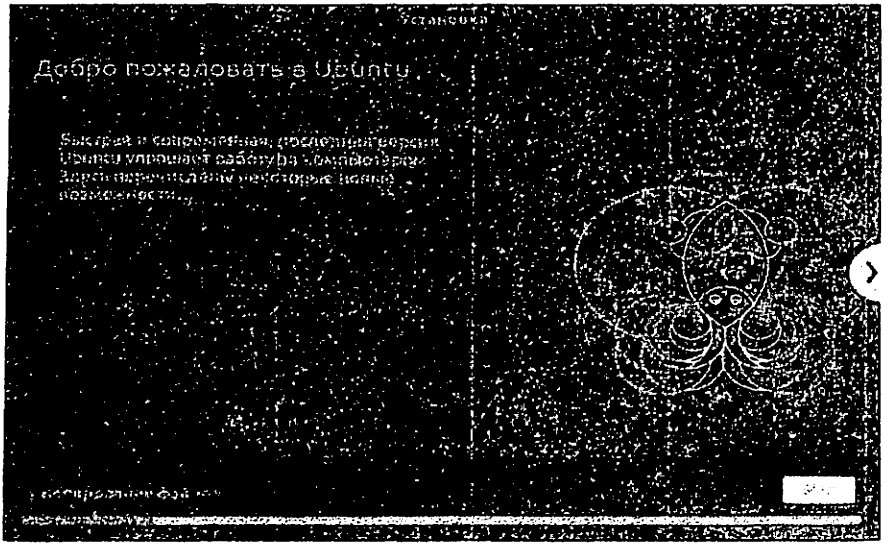

Рисунок 1.15. Загрузка Ubuntu

Шаг 10 – Завершение установки

Установка будет завершена, когда появится соответствующее сообщение. Нажимаем «Перезагрузить».

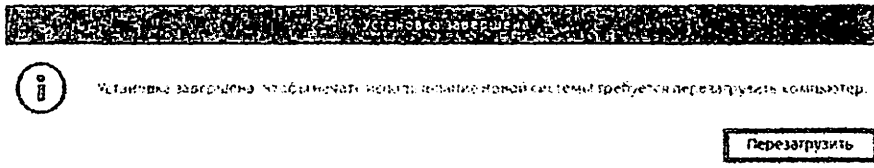

Рисунок 1.16. Завершение установки

Рабочий стол Linux Ubuntu

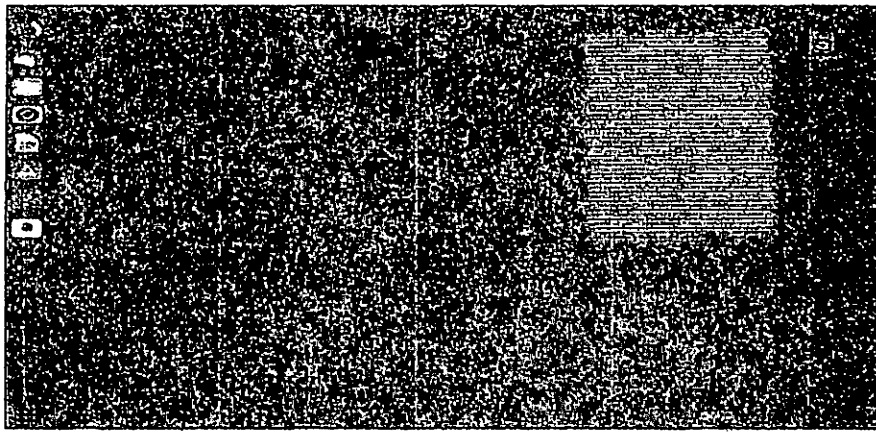

Рисунок 1.17. Рабочий стол ОС Linux Ubuntu

Контрольные вопросы:

1. Что такое Linux?
2. Что такое дистрибутив?
3. Перечислите основные дистрибутивы Linux;
4. Как создать пользователя?
5. Перечислите системные требования Linux Ubuntu.

ЛАБОРАТОРНАЯ РАБОТА № 2

Тема: Настройка и загрузка операционной системы Linux

Цель работы:

Изучить настройку сети операционной системы Linux, преобрести навыки в установке Extension Pack и ознакомиться с подключением к Ubuntu по RDP.

Заданве:

Ознакомиться с краткими теоретическими сведениями, изучить работы в Ubuntu и выполнить настройки при подключении по RDP.

Теоретическая часть:

После установки Ubuntu на виртуальную машину VirtualBox, сталкиваемся с тем, что не знаем, как переключиться на другую раскладку. Для этого нужно установить утилиты «Дополнительные настройки Gnome». Выполняем:

$ sudo apt install gnome-tweak-tool

Запускаем:

$ gnome-tweaks

Переходим на вкладку «Клавиатура и мышь», затем выбираем «Дополнительные параметры раскладки»:

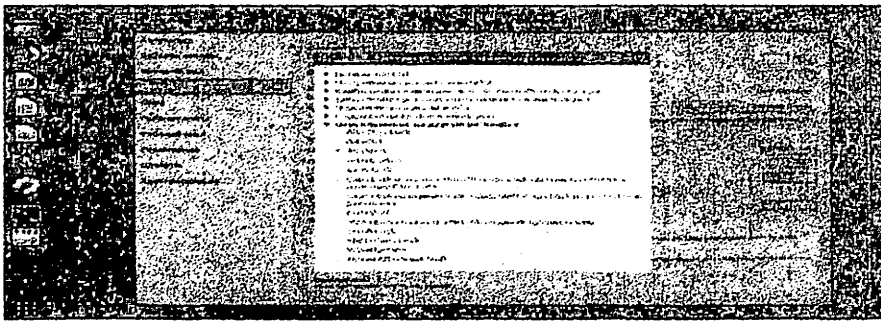

Рисунок 2.1. Переключение на другую раскладку в Ubuntu

Дополнения гостевой ОС

Дополнения VirtualBox для Linux представляют собой набор драйверов устройств и системных приложений, которые могут быть установлены в гостевой операционной системе — для повышения производительности и удобства использования.

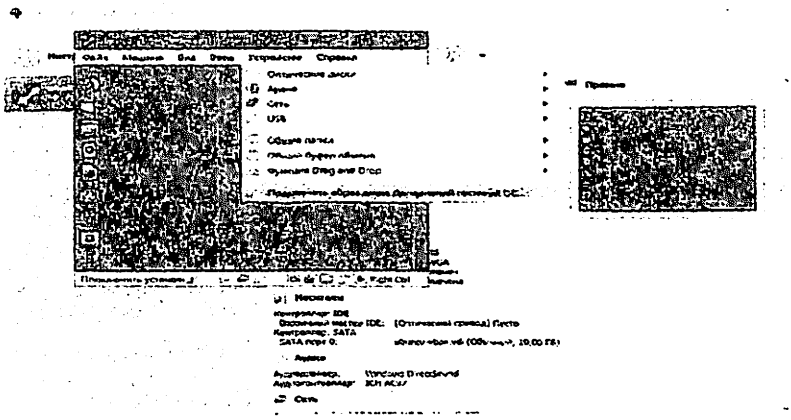

Рисунок 2.2. Подключение образа диска Дополнение гостевого ОС

После подключения диска Ubuntu предложит установить дополнения.

Настройка сети:

Существует несколько способов, как настроить сеть в VirtualBox, и каждый из них подходит для решения одной задачи и меньше для другой. Рассмотрим некоторые из них:

- NAT (преобразование сетевых адресов) — этот способ используется по умолчанию. Для каждой машины создается отдельная внутренняя локальная сеть, в которой машина получает ip-адрес 10.0.2.15. Это позволяет посещать web-страницы, скачивать файлы, просматривать электронную почту. Однако извне невозможно напрямую соединиться с такой системой.

- Принцип преобразования сетевых адресов заключается в следующем. Когда гостевая ОС отправляет пакеты на конкретный адрес удаленной машины в сети, сервис NAT, работающий под VirtualBox, перехватывает эти пакеты, извлекает из них сегменты, содержащие в себе адрес пункта отправки (IP-адрес гостевой операционной системы) и производит их замену на IP-адрес машины-хоста. Затем заново упаковывает их и отправляет по указанному адресу.
- Виртуальный адаптер хоста — создается виртуальный сетевой адаптер для хост-системы, к которому можно подключить несколько виртуальных машин, тем самым объединив их в локальную сеть. Виртуальный адаптер при этом работает как обычный сетевой коммутатор, соединяя между собой хост-систему и виртуальные машины. Доступа к интернету нет, но зато машины находятся в одной сети, и каждая имеет свой ip-адрес.

- основная система доступна по ip-адресу 192.168.56.1
- ip-адреса виртуальных машин: 192.168.56.101, 192.168.56.102
- ip-адрес DHCP-сервера VirtualBox: 192.168.56.100

- Сетевой мост — при таком подключении виртуальная машина становится полноценным членом локальной сети, к которой подключена основная система. Виртуальная машина получает адрес у роутера и становится доступна для других устройств, как и основной компьютер, по своему ip-адресу.

NAT (преобразование сетевых адресов)

NAT имитирует подключение к маршрутизатору. Маршрутизатором выступает сетевой модуль VirtualBox, обрабатывающий исходящие пакеты и пересылающий их хост-системе, точно так же происходит обработка входящего трафика. Маршрутизатор создается между каждой виртуальной

машиной и хост-системой. Посредством такого разделения виртуальная машина становится защищенной от контактов с другими машинами и проникновений со стороны внешней сети.

Виртуальная машина получает сетевой адрес от встроенного DHCP-сервера. Машине присваивается адрес из диапазона 10.0.XXX.0/24, где XXX обозначает адрес интерфейса, определяемый по формуле +2. Таким образом XXX будет равен 2, если имеется только один активный NAT-интерфейс. Гостевая операционная система получит адрес 10.0.2.15, сетевому шлюзу назначается адрес 10.0.2.2, сервер имен (DNS) получит 10.0.2.3.

Виртуальный адаптер хоста

Чтобы иметь выход в интернет, для виртуальной машины потребуются два адаптера. Первый будет обеспечивать выход в интернет, а второй — служить для организации сети между виртуальными машинами и хост-системой.

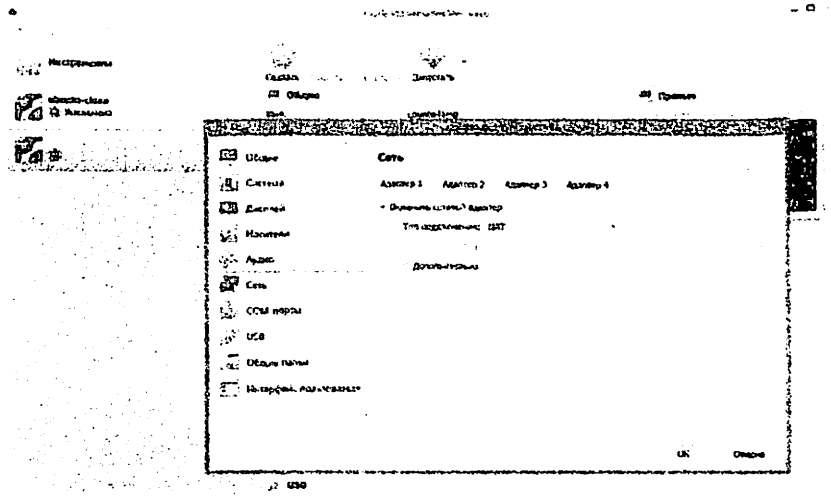

Рисунок 2.3. Настройка адаптера 1.

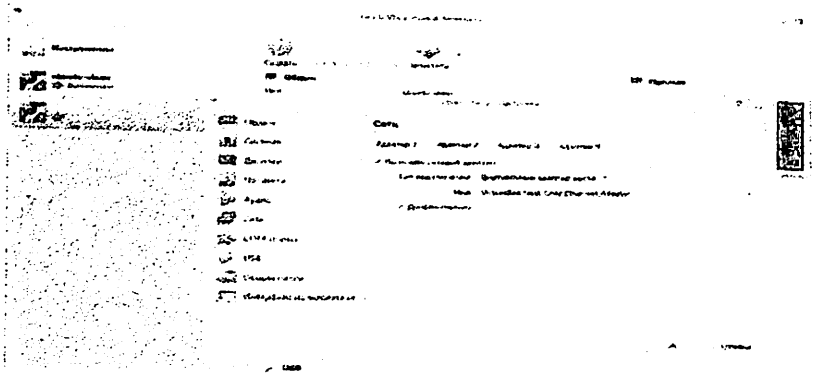

Рисунок 2.4. Настройка адаптера 2.

По умолчанию ip-адрес виртуальной машине выдается DHCP-сервером VirtualBox. Для удобства можно отключить DHCP-сервер и задать статический ip-адрес в самой виртуальной машине.

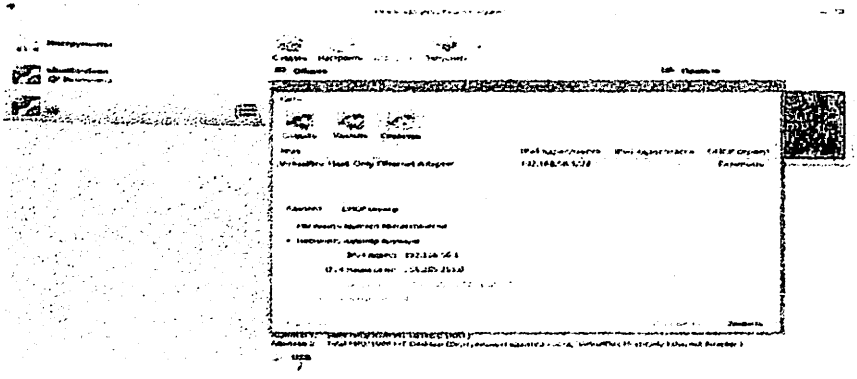

Рисунок 2.5. Установка IP-адреса.

Сетевой мост

В этом случае виртуальная машина работает также, как и все остальные компьютеры в локальной сети. Адаптер подключается, минуя хост-систему, к роутеру, который распределяет IP-адреса внутри локальной сети для всех устройств.

VirtualBox соединяется с сетевой картой хост-системы и передает пакеты через нее напрямую. Адаптер получает от DHCP-сервера на роутере стандартный адрес из диапазона 192.168.XXX.XXX. Поэтому виртуальная машина в сети выглядит так, как будто это обычное физическое устройство, неотличимое от остальных.

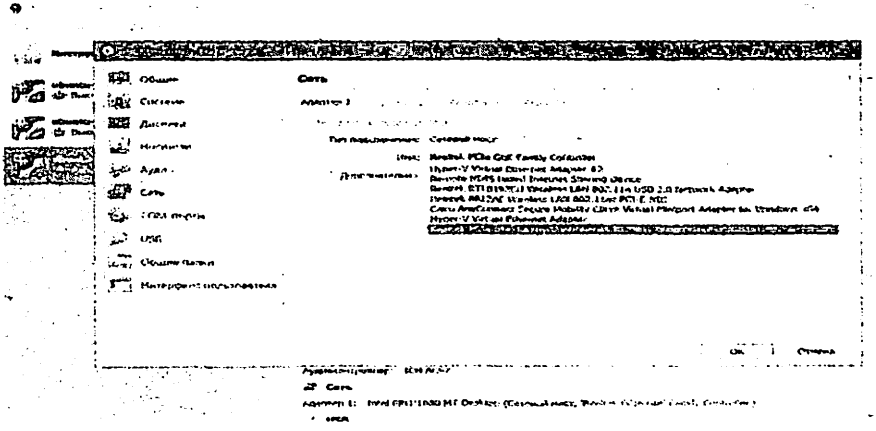

Рисунок 2.6. Тип подключение «Сетевой мост».

Установка Extension Pack

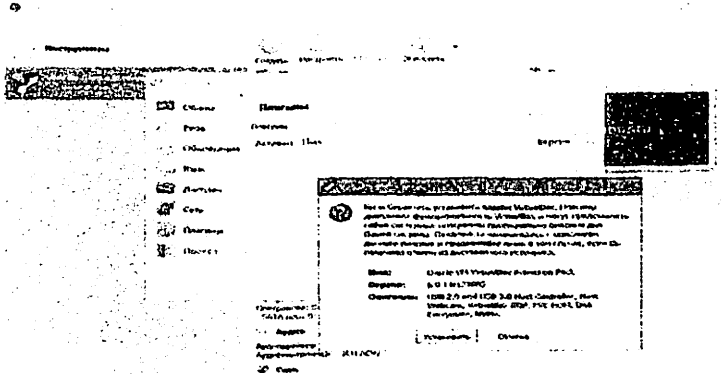

Рисунок 2.7. Установка Extension Pack

Порядок выполнения работы:

Подключение к Ubuntu по RDP

Для подключения по RDP переходим в настройки виртуальной машины, пункт меню «Дисплей», вкладка «Удаленный доступ»:

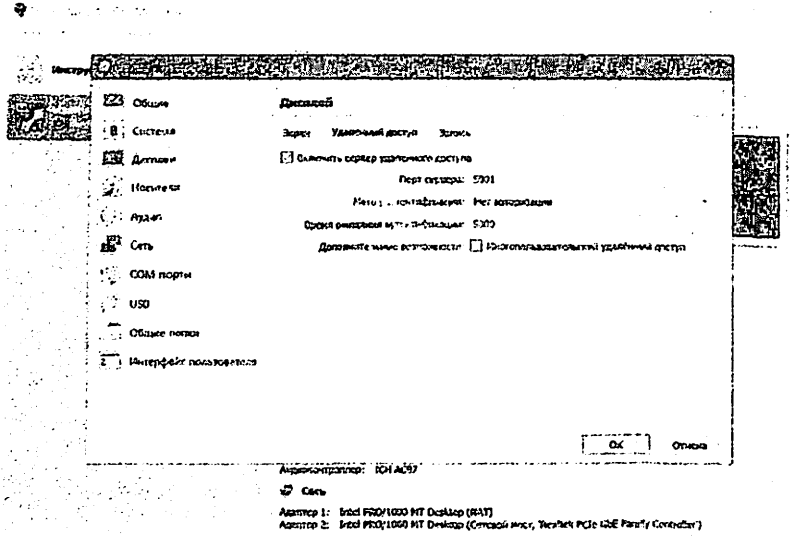

Рисунок 2.8. Подключение к Ubuntu по RDP

Теперь можно подключаться, набираем в командной строке:

> mstsc /v:192.168.110.2:5001

Здесь 192.168.110.2 — IP-адрес хост-системы, а 5001 — порт, который мы указали в настройках.

Рисунок 2.9. Рабочий стол Ubuntu

При подключении по RDP окно нам не нужно, поэтому виртуальную машину нужно запускать в фоновом режиме:

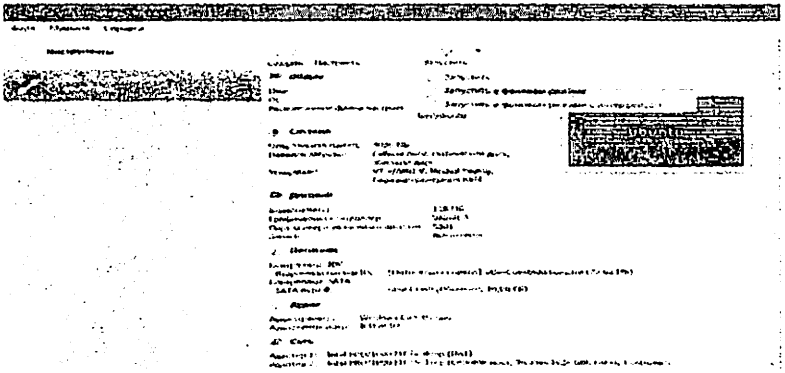

Рисунок 2.10. Рабочий стол Oracle VM VirtualBox

Контрольные вопросы:

1. Перечислите настройки сети Ubuntu;
2. Дать определение DHCP сервер?
3. Объясните подключение Ubuntu по RDP;
4. Дать определение термину сетевой мост;
5. Объясните настройку адаптера2.

ЛАБОРАТОРНАЯ РАБОТА № 3

Тема: Работа с интерфейсом операционной системы и командной строкой

Цель работы:

Формирование умения работать с интерфейсом командной строки, с терминалом и командами операционной системы Linux.

Задание:

Ознакомиться с краткими теоретическими сведениями, приобрести навыки работы в терминале. Создать новых пользователей при помощи терминала Linux, должен уметь задавать несложные команды.

Теоретическая часть:

В Ubuntu существует два вида интерфейса: графический интерфейс пользователя и интерфейс командной строки.

Графический интерфейс пользователя *(англ. Graphical user interface, GUI).* - управление программами с помощью графических кнопок, всплывающих меню, окон и других элементов. Множество действий можно выполнять с помощью мыши.

Преимущества: визуальное отображение программ и их содержимого, возможности программ можно изучать без чтения документации.

Интерфейс командной строки *(англ. Command Line Interface, CLI).* - управление программами с помощью команд. Команды состоят из букв, цифр, символов, набираются построчно, выполняются после нажатия клавиши Enter. Основной инструмент здесь клавиатура. Данный интерфейс встроен в ядро системы, он будет доступен, даже если графический интерфейс не запустится.

Преимущества: небольшой расход ресурсов, гибкость при составлении перечня действий из команд, возможность автоматического выполнения команд, возможность копировать и вставлять команды.

Если сравнивать интерфейсы в разных системах, то можно заметить, что основные команды одинаковы во всех дистрибутивах семейства Linux, а вот графические программы в каждой системе могут очень сильно различаться. Добраться до командной строки можно двумя способами: через консоль или терминал.

Консоль

Во время загрузки Ubuntu запускаются семь полноэкранных консолей, у каждой свой независимый сеанс, с первой по шестую с интерфейсом командной строки, в седьмой запускается графический режим. Пользователь во время загрузки видит только графический режим. Переключиться на одну из виртуальных консолей можно нажав сочетание клавиш:

Ctrl+Alt+F1 - первая виртуальная консоль;
Ctrl+Alt+F2 - вторая виртуальная консоль;
Ctrl+Alt+F3 - третья виртуальная консоль;
Ctrl+Alt+F4 - четвертая виртуальная консоль;
Ctrl+Alt+F5 - пятая виртуальная консоль;
Ctrl+Alt+F6 - шестая виртуальная консоль;
Ctrl+Alt+F7 - седьмая виртуальная консоль, возврат в графический режим.

Терминал

Терминал - графическая программа эмулирующая консоль. Такие программы позволяют, не выходя из графического режима выполнять команды. Терминал по сравнению с консолью имеет дополнительный функционал (управление мышью, контекстное меню, полоса прокрутки, вкладки, запуск нескольких окон, главное меню, графические настройки).

Запустить терминал можно следующим образом:

В Unity:

Главное меню → Набрать в поисковой строке слово Терминал

или нажать комбинацию клавиш: Ctrl+Alt+T

В Gnome Fallback

Приложения → Стандартные → Терминал

В Xfce (Xubuntu):

Главное меню → Приложения → Система → Терминал

В KDE (Kubuntu):

Главное меню → Приложения → Система → Терминал

В LXDE (Lubuntu):

Главное меню → Системные → LXTerminal

После запуска терминала мы видим строку с приглашением к вводу команд, например:

vladimir@Zotac-Zbox-Nano:~$

vladimir - имя учетной записи пользователя

@ - разделитель между учетной записью и именем компьютера

Zotac-Zbox-Nano - имя компьютера

: - разделитель

~ - в какой папке выполняется команда, ~ это домашняя папка пользователя, если выполните команду ls то получите список файлов из этой папки

$ - приглашение к выполнению команды с правами простого пользователя (# будет означать приглашение на выполнение команд с правами администратора)

Горячие клавиши

Копирование команд

Вставить текст в терминал можно тремя способами: Ctrl+Shift+V, нажатием средней кнопки мыши или правой кнопки мыши и выбором строки «Вставить».

Таблица 3.1

Экономия набора

↑ или Ctrl+P	прокрутка недавно использованных команд вверх
↓ или Ctrl+N	прокрутка недавно использованных команд вниз
Enter	выполнение выбранной команды

Tab	крайне удобная возможность - автозаподстановка команд и имён файлов. Если с выбранных символов начинается только одна команда, подставится именно она, а если их несколько, то по двойному нажатию tab выведется список всех возможных вариантов.
Ctrl+R	поиск по командам, которые вы вводили раньше. Если вам нужно повторно выполнить очень длинную и сложную команду, вы можете ввести только её часть, а эта комбинация клавиш поможет найти команду целиком.
History	Команда history выводит список всех команд, которые вы вводили. Каждой команде будет присвоен номер. Чтобы выполнить команду под номером x, просто введите «!x». Если у вас получилась слишком длинная история, можно попробовать «history I less», это сделает список прокручиваемым.

Таблица 3.2

Изменение текста

ctrl+a или Home	перемещает курсор в начало строки
ctrl+e или End	перемещает курсор в конец строки
ctrl+b	перемещает курсор в начало предыдущего или текущего слова
ctrl+k	удаляет текст с текущей позиции курсора до конца строки
ctrl+u	удаляет всю текущую строку
ctrl+w	удаляет слово перед курсором

Программная оболочка

Консоль и терминал обрабатывают команды с помощью программной оболочки. Программная оболочка - интерпретатор команд, он распознает команды, введенные в командной строке, и запускает программы для выполнения команды. В Ubuntu по умолчанию используется оболочка bash, он распознает команды на языке bash. Bash можно заменить на другую оболочку, их существует несколько. Каждая оболочка имеет свой набор настроек и

возможностей.

Команды

Команды — это предопределенный набор букв, цифр, символов, которые можно ввести в командной строке и выполнить, нажав энтер.

Команды делятся на два вида:

- команды встроенные в программную оболочку (например history)
- команды управляющие программами, установленными в системе

Название программы - это название исполняемого файла из каталогов записанных в переменной $PATH (/bin, /sbin, /usr/bin, /usr/sbin, /usr/local/bin, /usr/local/sbin и др.) или полный путь к исполняемому файлу (/opt/deadbeef/bin/deadbeef)

Ключ - пишется после названия программы, например -h, у каждой программы свой набор ключей, они перечислены в справке к программе, ключи используются для указания какие настройки использовать или какое действие выполнить

Значение - адрес, цифры, текст, спецсимволы (*, ~, \, &, « », _), переменные *($HOME, $USER, $PATH)*

Выполнить команды можно следующим образом:

- набрать команду в командной строке и нажать Enter
- скопировать команду из инструкции и вставить ее в командную строку, затем нажать Enter
- создать скрипт и выполнить двойным нажатием мыши (создать текстовый файл, в первой строке написать #!/bin/bash, ниже написать команды в столбик, сохранить, в свойствах файла разрешить выполнение, нажать два раза по файлу для выполнения всех перечисленных команд)

Помните, что терминал чувствителен к регистру! Слова User, user и USER в Linux различаются!

Таблица 3.3

Файловые команды

cd ../..	перейти в директорию двумя уровнями выше
cd	перейти в домашнюю директорию
cd ~user	перейти в домашнюю директорию пользователя user
cd -	перейти в директорию, в которой находились до перехода в текущую директорию
pwd	показать текущую директорию
mkdir dir	создать каталог dir
mkdir dir1	создать директорию с именем 'dir1'
mkdir dir1 dir2	создать две директории одновременно
mkdir -p /tmp/dir1/dir2	создать дерево директорий
rm file	удалить file
rm -r dir	удалить каталог dir
rm -f file	удалить форсированно file
rm -rf dir	удалить форсированно каталог dir
rm -f file1	удалить файл с именем 'file1'
rmdir dir1	удалить директорию с именем 'dir1'
rm -rf dir1	удалить директорию с именем 'dir1' и рекурсивно всё её содержимое
rm -rf dir1 dir2	удалить две директории и рекурсивно их содержимое
cp file1 file2	скопировать file1 в file2
cp -r dir1 dir2	скопировать dir1 в dir2; создаст каталог dir2, если он не существует
cp dir/	копировать все файлы директории dir в текущую директорию
cp -a /tmp/dir1	копировать директорию dir1 со всем содержимым в текущую директорию

cp -a dir1 dir2	копировать директорию dir1 в директорию dir2
mv dir1 new_dir	переименовать или переместить файл или директорию
mv file1 file2	переименовать или переместить file1 в file2. если file2 существующий каталог - переместить file1 в каталог file2
ln -s file1 lnk1	создать символическую ссылку на файл или директорию
ln file1 lnk1	создать «жёсткую» (физическую) ссылку на файл или директорию
touch file	создать file
touch -t 0712250000 fileditest	модифицировать дату и время создания файла, при его отсутствии, создать файл с указанными датой и временем (YYMMDDhhmm)
cat > file	направить стандартный ввод в file
more file	вывести содержимое file
head file	вывести первые 10 строк file
tail file	вывести последние 10 строк file
tail -f file	вывести содержимое file по мере роста, начинает с последних 10 строк

Таблица 3.4

Просмотр содержимого файлов

cat file1	вывести содержимое файла file1 на стандартное устройство вывода
tac file1	вывести содержимое файла file1 на стандартное устройсво вывода в обратном порядке (последняя строка становиться первой и т.д.)
more file1	постраничный вывод содержимого файла file1 на стандартное устройство вывода

less file1	постраничный вывод содержимого файла file1 на стандартное устройство вывода, но с возможностью пролистывания в обе стороны (вверх-вниз), поиска по содержимому и т.п.
head -2 file1	вывести первые две строки файла file1 на стандартное устройство вывода. По умолчанию выводится десять строк
tail -2 file1	вывести последние две строки файла file1 на стандартное устройство вывода. По умолчанию выводится десять строк
tail -f /var/log/messages	выводить содержимое файла /var/log/messages на стандартное устройство вывода по мере появления в нём текста

Таблица 3.5

Преобразование наборов символов и файловых форматов

dos2unix filedos.txt fileunix.txt	конвертировать файл текстового формата из MSDOS в UNIX (разница в символах возврата каретки)
unix2dos fileunix.txt filedos.txt	конвертировать файл текстового формата из UNIX в MSDOS (разница в символах возврата коретки)
recode ..HTML < page.txt > page.html	конвертировать содержимое тестового файла page.txt в html-файл page.html
recode -l | more	вывести список доступных форматов

Таблица 3.6

Анализ файловых систем

badblocks -v /dev/hda1	проверить раздел hda1 на наличие bad-блоков
fsck /dev/hda1	проверить/восстановить целостность linux-файловой системы раздела hda1
fsck.ext2 /dev/hda1	проверить/восстановить целостность файловой системы ext2 раздела hda1
e2fsck -j /dev/hda1	проверить/восстановить целостность файловой системы ext3 раздела hda1 с указанием, что журнал расположен там же
fsck.ext3 /dev/hda1	проверить/восстановить целостность файловой системы ext3 раздела hda1
fsck.vfat /dev/hda1	проверить/восстановить целостность файловой системы fat раздела hda1

Таблица 3.7

Монтирование файловых систем

mount /dev/hda2 /mnt/hda2	монтирует раздел 'hda2' в точку монтирования '/mnt/hda2'. Убедитесь в наличии директории-точки монтирования '/mnt/hda2'
umount /dev/hda2	размонтирует раздел 'hda2'. Перед выполнением, покиньте '/mnt/hda2'
fuser -km /mnt/hda2	принудительное размонтирование раздела. Применяется в случае, когда раздел занят каким-либо пользователем
umount -n /mnt/hda2	выполнить размонтирование без занесения информации в /etc/mtab. Полезно когда файл имеет

	атрибуты «только чтение» или недостаточно места на диске
mount /dev/fd0 /mnt/floppy	монтировать флоппи-диск
mount /dev/cdrom /mnt/cdrom	монтировать CD или DVD
mount /dev/hdc /mnt/cdrecorder	монтировать CD-R/CD-RW или DVD-R/DVD-RW(+-)
mount -o loop file.iso /mnt/cdrom	смонтировать ISO-образ
mount -t vfat /dev/hda5 /mnt/hda5	монтировать файловую систему Windows FAT32

Монтирование файловой системы — процесс, подготавливающий раздел диска к использованию операционной системой.

Операция монтирования состоит из нескольких этапов:

1. определение типа монтируемой системы;
2. проверка целостности монтируемой системы;
3. считывание системных структур данных и инициализация соответствующего модуля файлового менеджера (драйвера файловой системы);
4. установка флага, сообщающего об окончании монтирования. При корректном размонтировании этот флаг сбрасывается;
5. включение новой файловой системы в общее пространство имен;

Таблица 3.10

Форматирование файловых систем

mkfs /dev/hda1	создать linux-файловую систему на разделе hda1
mke2fs /dev/hda1	создать файловую систему ext2 на разделе hda1

mke2fs -j /dev/hda1	создать журналирующую файловую систему ext3 на разделе hda1
mkfs -t vfat 32 -F /dev/hda1	создать файловую систему FAT32 на разделе hda1
fdformat -n /dev/fd0	форматирование флоппи-диска без проверки
mkswap /dev/hda3	создание swap-пространства на разделе hda3

Форматирование — программный процесс разметки области хранения данных электронных носителей информации, расположенной на магнитной поверхности (жёсткие диски, дискеты), оптических носителях (CD/DVD/Blu-ray-диски), твердотельных накопителях (флэш-память — flash module, SSD) и др. Существуют разные способы этого процесса.

Само форматирование заключается в создании (формировании) структур доступа к данным, например, структур файловой системы. При этом возможность прямого доступа к находящейся (находившейся до форматирования) на носителе информации теряется, часть её безвозвратно уничтожается. Некоторые программные утилиты дают возможность восстановить некоторую часть (обычно — большую) информации с отформатированных носителей. В процессе форматирования также может проверяться и исправляться целостность носителя.

Таблица 3.8

Создание резервных копий (backup).

dump -0aj -f /tmp/home0.bak /home	создать полную резервную копию директории /home в файл /tmp/home0.bak
dump -1aj -f /tmp/home0.bak /home	создать инкрементальную резервную копию директории /home в файл /tmp/home0.bak

restore -if /tmp/home0.bak	восстановить из резервной копии /tmp/home0.bak
rsync -rogpav –delete /home /tmp	синхронизировать /tmp с /home
rsync -rogpav -e ssh –delete /home ip_address:/tmp	синхронизировать через SSH-туннель
rsync -az -e ssh –delete ip_addr:/home/public /home/local	синхронизировать локальную директорию с удалённой директорией через ssh-туннель со сжатием
rsync -az -e ssh –delete /home/local ip_addr:/home/public	синхронизировать удалённую директорию с локальной директорией через ssh-туннель со сжатием
dd bs=1M if=/dev/hda | gzip | ssh user@ip_addr 'dd of=hda.gz'	сделать «слепок» локального диска в файл на удалённом компьютере через ssh-туннель
tar -Puf backup.tar /home/user	создать инкрементальную резервную копию директории '/home/user' в файл backup.tar с сохранением полномочий
(cd /tmp/local/ && tar c .) | ssh -C user@ip_addr 'cd /home/share/ && tar x -p'	копирование содержимого /tmp/local на удалённый компьютер через ssh-туннель в /home/share/
(tar c /home) | ssh -C user@ip_addr 'cd /home/backup-home && tar x -p'	копирование содержимого /home на удалённый компьютер через ssh-туннель в /home/backup-home
tar cf - . | (cd /tmp/backup ; tar xf -)	копирование одной директории в другую с сохранением полномочий и линков

find /home/user1 -name '*.txt' | xargs cp -av –target-directory=/home/backup/ –parents	поиск в /home/user1 всех файлов, имена которых оканчиваются на '.txt', и копирование их в другую директорию
find /var/log -name '*.log' | tar cv –files-from= | bzip2 > log.tar.bz2	поиск в /var/log всех файлов, имена которых оканчиваются на '.log', и создание bzip-архива из них
dd if=/dev/hda of=/dev/fd0 bs=512 count=1	создать копию MBR (Master Boot Record) с /dev/hda на флоппи-диск
dd if=/dev/fd0 of=/dev/hda bs=512 count=1	восстановить MBR с флоппи-диска на /dev/hda

Резервное копирование (англ. backup copy) — процесс создания копии данных на носителе (жёстком диске, дискете и т. д.), предназначенном для восстановления данных в оригинальном или новом месте их расположения в случае их повреждения или разрушения.

Таблица 3.9

Пользователи и группы

groupadd group_name	создать новую группу с именем group_name
groupdel group_name	удалить группу group_name
groupmod -n new_group_name old_group_name	переименовать группу old_group_name в new_group_name
useradd -c «Nome Cognome» -g admin -d /home/user1 -s /bin/bash user1	создать пользователя user1, назначить ему в качестве домашнего каталога /home/user1, в качестве shell'a /bin/bash, включить его в группу admin и добавить комментарий Nome Cognome
useradd user1	создать пользователя user1

userdel -r user1	удалить пользователя user1 и его домашний каталог
usermod -c «User FTP» -g system -d /ftp/user1 -s /bin/nologin user1	изменить атрибуты пользователя
passwd	сменить пароль
passwd user1	сменить пароль пользователя user1 (только root)
chage -E 2005-12-31 user1	установить дату окончания действия учётной записи пользователя user1
pwck	проверить корректность системных файлов учётных записей. Проверяются файлы /etc/passwd и /etc/shadow
grpck	проверяет корректность системных файлов учётных записей. Проверяется файл/etc/group
newgrp [-] group_name	изменяет первичную группу текущего пользователя. Если указать «-», ситуация будет идентичной той, в которой пользователь вышил из системы и снова вошёл. Если не указывать группу, первичная группа будет назначена из /etc/passwd

Таблица3.10

Выставление/изменение полномочий на файлы

ls -lh	просмотр полномочий на файлы и директории в текущей директории
ls /tmp | pr -T5 -WSCOLUMNS	вывести содержимое директории /tmp и разделить вывод на пять колонок
chmod ugo+rwx directory1	добавить полномочия на директорию directory1 ugo(User Group Other)+rwx(Read Write eXecute) - всем

	полные права. Аналогичное можно сделать таким образом chmod 777 directory1
chmod go-rwx directory1	отобрать у группы и всех остальных все полномочия на директорию directory1
chown user1 file1	назначить владельцем файла file1 пользователя user1
chown -R user1 directory1	назначить рекурсивно владельцем директории directory1 пользователя user1
chgrp group1 file1	сменить группу-владельца файла file1 на group1
chown user1:group1 file1	сменить владельца и группу владельца файла file1
find / -perm -u+s	найти, начиная от корня, все файлы с выставленным SUID

chmod (от англ. change mode) — программа для изменения прав доступа к файлам и директориям. Название происходит от программы ОС Unix chmod, которая, собственно, изменяет права доступа к файлам, директориям и символическим ссылкам.

Контрольные вопросы:

1. Объясните понятие терминал?
2. Дать определение понятию программная оболочка?
3. Объясните создание резервных копий;
4. Перечислите команды выставления/изменения полномочий на файлы;
5. Перечислите команды форматирования файловых систем.

ЛАБОРАТОРНАЯ РАБОТА № 4

Тема: Управление процессами в Linux

Цель работы:

Получение навыков в работе с процессами в Linux, изменить приоритета процессов и ознакомиться с запущенными процессами.

Задание:

Ознакомиться с запущенными процессами и при помощи команды *htop* посмотреть запущенные процессы и изменить их приоритет. Дать комментарии каждому процессу.

Теоретическая часть:

В Linux для каждой отдельной программы, при ее запуске создается процесс. Неважно запускаете программу вы вручную самостоятельно или это делает система или ядро. Например, программа инициализации, которая запускается сразу после завершения загрузки ядра тоже имеет свой процесс с идентификатором 0. Процессы в linux можно описать как контейнеры, в которых хранится вся информация о состоянии и выполнении программы. Если программа работает хорошо, то все нормально, но если она зависла или вам нужно настроить ее работу может понадобиться управление процессами в Linux.

Процесс — это каждая программа. Для каждой запускаемой программы создается отдельный процесс. В рамках процесса программе выделяется процессорное время, оперативная память и другие системные ресурсы. У каждого процесса есть свой идентификатор, Proccess ID или просто PID, по ним, чаще всего и определяются процессы Linux. PID определяется неслучайно, как я уже говорил, программа инициализации получает PID 1, а каждая следующая запущенная программа - на единицу больше. Таким образом PID пользовательских программ доходит уже до нескольких тысяч.

На самом деле, процессы Linux не настолько абстрактны, какими они вам сейчас кажутся. Их вполне можно попытаться пощупать. Откройте ваш файловый менеджер, перейдите в корневой каталог, затем откройте папку

/proc. Видите здесь кучу номеров? Так вот это все - PID всех запущенных процессов. В каждой из этих папок находится вся информация о процессе.

Например, посмотрим папку процесса 1. В папке есть другие под каталоги и много файлов. Файл cmdline содержит информацию о команде запуска процесса:

cat /proc/1/cmdline

/usr/lib/systemd/systemd

Поскольку у меня используется система инициализации Systemd, то и первый процесс запускается для нее. С помощью каталога /proc можно сделать все. Но это очень неудобно, особенно учитывая количество запущенных процессов в системе. Поэтому для реализации нужных задач существуют специальные утилиты. Перейдем к рассмотрению утилит, которые позволяют реализовать управление процессами в Linux.

Управление процессами в linux

В Linux есть очень большое количество утилит для решения различных задач по управлению процессами. Это и такие многофункциональные решения, как htop, top, а также простые утилиты, например, ps, kill, killall, who и т д. Я не буду рассматривать в этой статье графические утилиты, и top тоже рассматривать не буду. Первое потому что слишком просто, второе - потому что htop лучше. Мы остановимся на работе с программой htop и ее аналогами в форме утилит в стиле GNU, одна утилита - одна функция.

Давайте установим htop, если она у вас еще не установлена. В Ubuntu это делается так:

sudo apt install htop

В других дистрибутивах вам нужно просто использовать свой менеджер пакетов. Имя пакета такое же.

Порядок выполнения работы:

Посмотреть запущенные процессы

Для этого существует утилиты, начиная от обычной ps, до более продвинутых интерактивных top, htop и так далее.

Открыв htop, мы сразу видим список запущенных процессов. Конечно, здесь отображены не все процессы linux, их-то в системе очень много, вы уже знаете, все они на один экран не поместятся. По умолчанию выводятся процессы, запущенные от имени вашего пользователя:

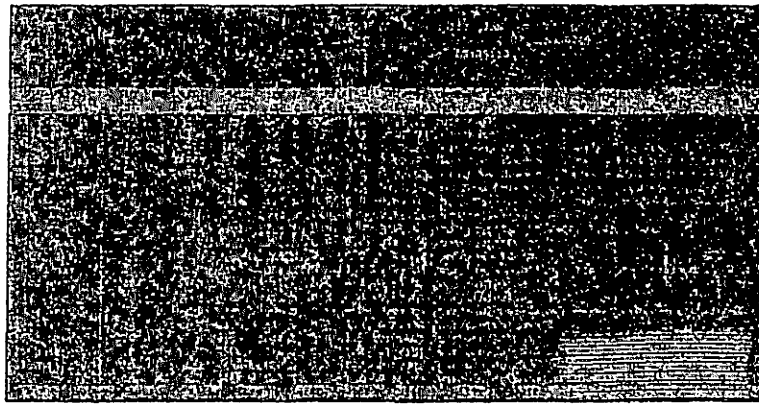

Рисунок 4.1. Запущенные процессы в терминале

Вы можете увидеть такую информацию о процессе:

- **PID** - идентификатор процесса
- **USER** - пользователь, от которого был запущен процесс
- **PRI** - приоритет процесса linux на уровне ядра (обычно NI+20)
- **NI** - приоритет выполнения процесса от -20 до 19
- **S** - состояние процесса
- **CPU** - используемые ресурсы процессора
- **MEM** - использованная память
- **TIME** - время работы процесса

К отображению можно добавить и дополнительные параметры, но эти главные. Добавить параметры можно с помощью меню Setup. Там все очень просто, читайте подсказки и следуйте указаниям. Например, добавлен параметр PPID:

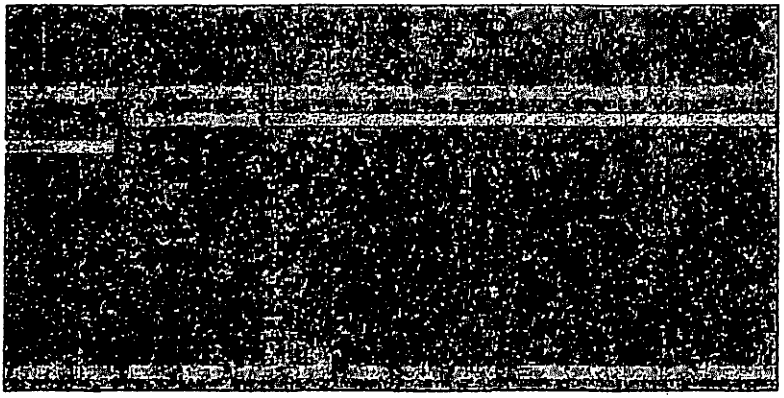

Рисунок 4.2. Добавление дополнительных параметров

Очень важной особенностью программы есть то, что вы можете сортировать процессы в Linux по нужному параметру. Просто кликните по названию параметра, оно выделится зеленым и будет выполнена сортировка. Например, хотите посмотреть в каком порядке запускались процессы, сортируем по PID:

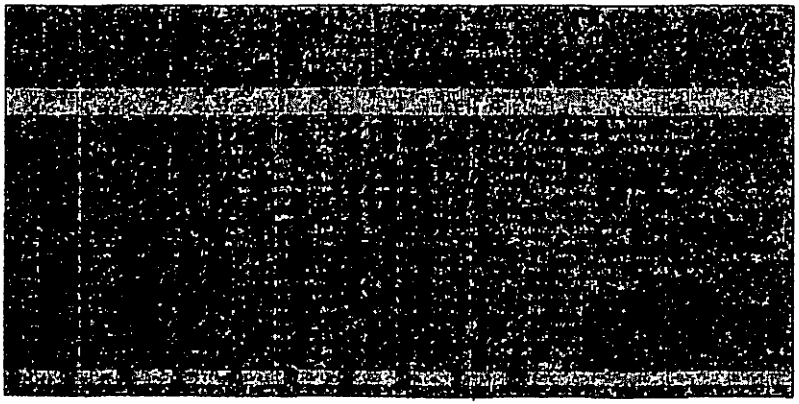

Рисунок 4.3. Сортировка по PID

Также есть интересная возможность разместить процессы в виде дерева. Вы сможете увидеть, каким процессом был запущен тот или иной процесс. Для отображения дерева нажмите кнопку F5:

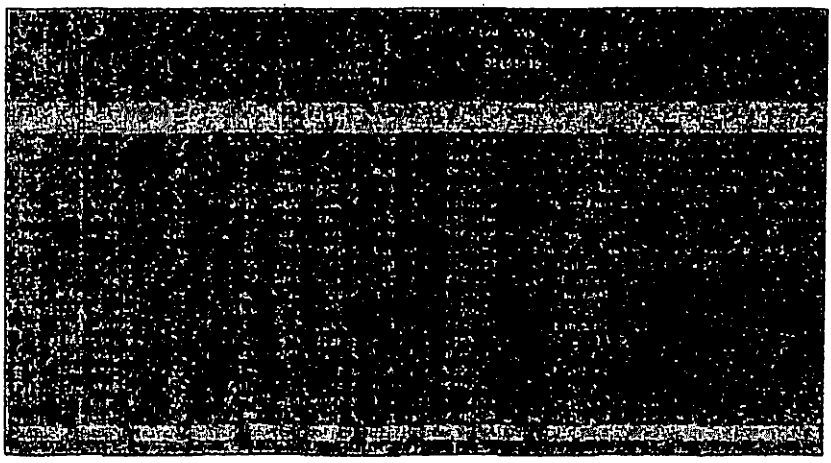

Рисунок 4.4. Сортировка в виде «Дерево»

Почти те же действия вы можете выполнять с помощью программы ps. Только здесь нет такого удобного интерактивного режима. Все делается с помощью опций.

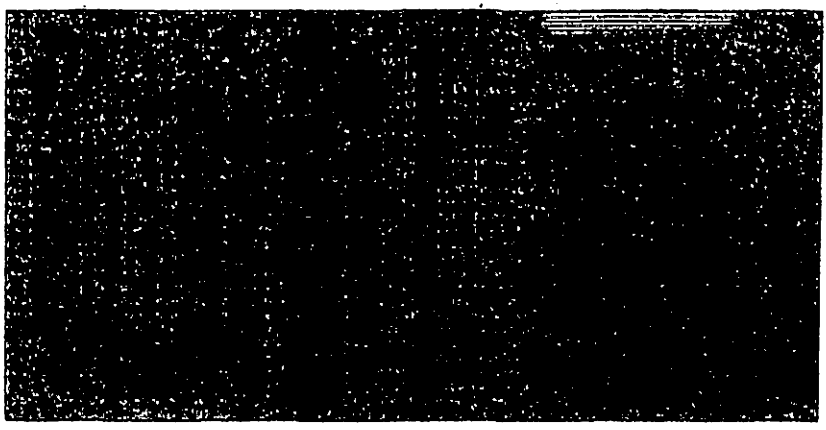

Рисунок 4.5. Сортировка с помощью программы «ps»

Рассмотрим основные опции, которые будем использовать:

- -e - вывести информацию обо всех процессах;
- -a - вывести информацию обо всех наиболее часто запрашиваемых процессах;

- **-t** - показывать только процессы из этого терминала;
- **-p** - показывать информацию только об указанном процессе;
- **-u** - показывать процессы только определенного пользователя;

Чтобы посмотреть все активные на данный момент процессы в linux, используется сочетание опций aux:

ps aux

Программа показывает все те же параметры, только здесь нет интерактивного интерфейса. Думаете здесь нельзя отсортировать процессы, но ошибаетесь, можно. Для этого есть опция sort. Вы можете сортировать их по любому полю, например:

ps aux --sort=%mem

Список будет отсортирован в обратном порядке, внизу значения больше, вверху - меньше. Если нужно в обратном порядке, добавьте минус:

ps aux --sort=-%cpu

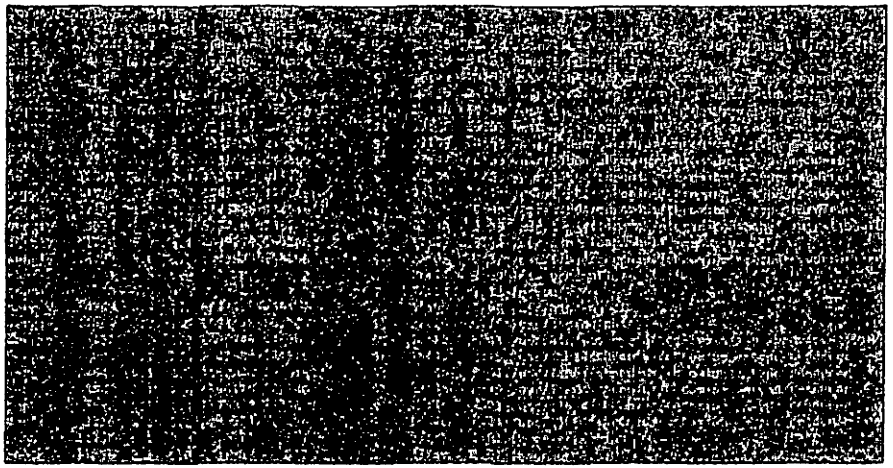

Рисунок 4.6. Сортировка с помощью команды «*ps aux --sort=-%cpu*»

В качестве поля для сортировки могут быть использованы приоритеты процессов Linux или любые другие параметры. Также вы можете обрезать вывод, если не нужно выводить всю информацию:

ps aux | tail

Казалось бы, у *ps* нет возможности стоить деревья процессов. Но не совсем, для этого существует отдельная команда:

pstree

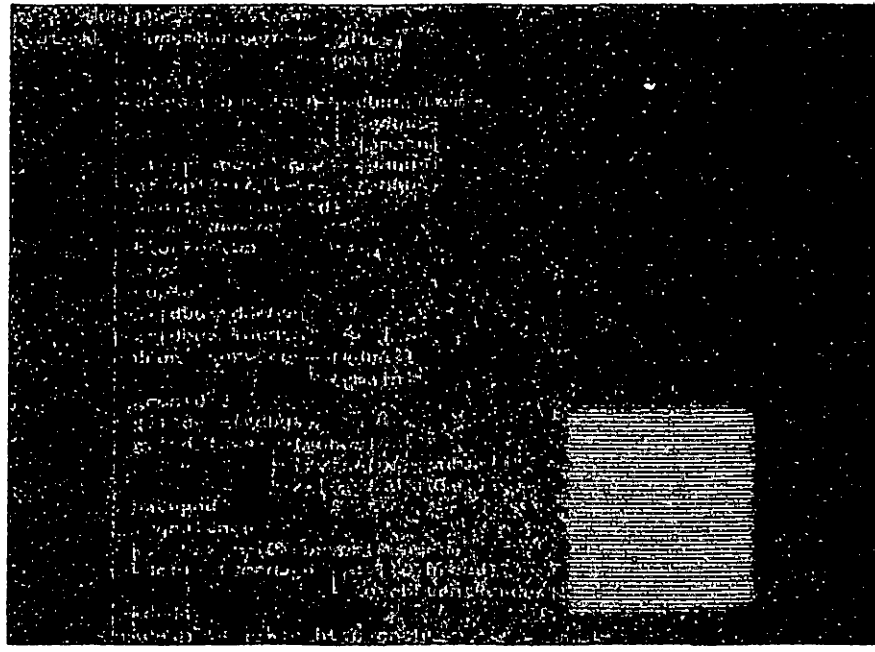

Рисунок 4.7. Сортировка в виде дерево с помощью команды «*pstree*»

Поиск процессов в linux

Когда какой-нибудь процесс завис и нужно убить процесс Linux или нам нужно провести с ним какие-либо действия, нужно выделить этот процесс из списка, узнать его PID и информацию о нем.

Чтобы найти процесс linux в htop можно использовать кнопку F3. Нажмите F3 и наберите нужное слово. Дальше чтобы перейти к следующему вхождению нажимайте F2 или Esc для завершения поиска:

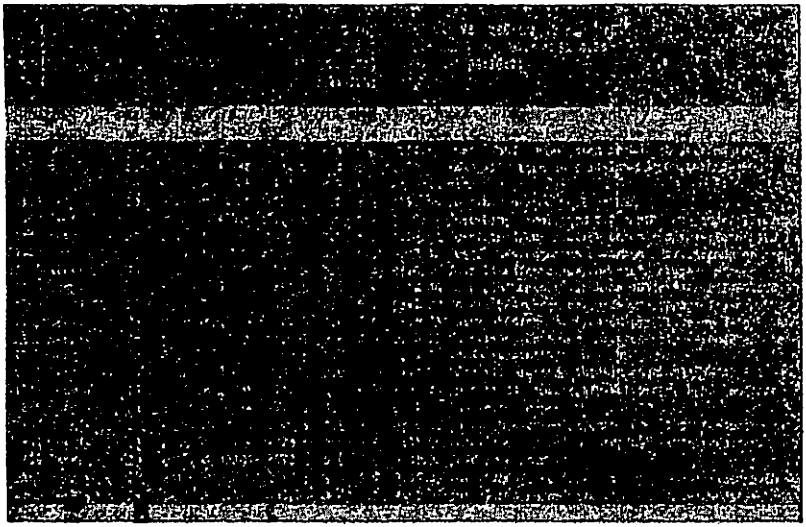

Рисунок 4.8.Поиск процессов в LINUX

Для поиска процессов в htop можно использовать также фильтр htop. Нажмите F4, введите слово и будут выведены только процессы linux, имя которых включает это слово.

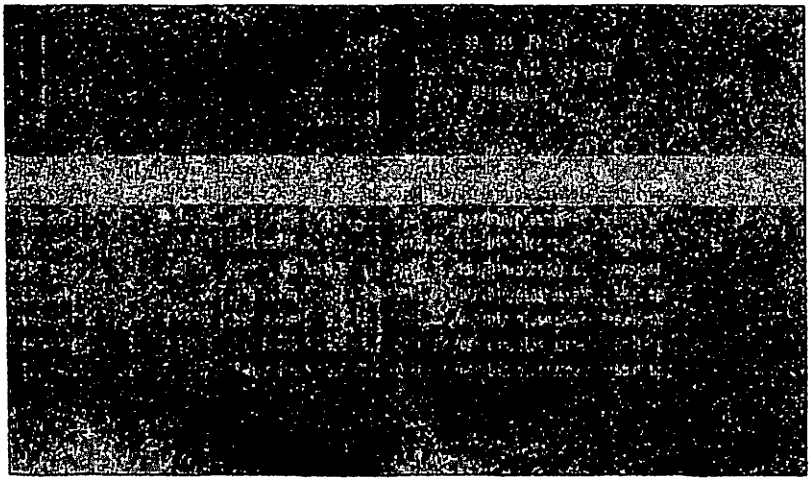

Рисунок 4.9. Фильтрация htop

Изменение приоритета процессов

Приоритет процесса linux означает, насколько больше процессорного времени будет отдано этому процессу по сравнению с другими. Так мы можем очень тонко настроить какая программа будет работать быстрее, а какая медленнее. Значение приоритета может колебаться от 19 (минимальный приоритет) до -20 - максимальный приоритет процесса linux. Причем, уменьшать приоритет можно с правами обычного пользователя, но чтобы его увеличить нужны права суперпользователя.

В htop для управления приоритетом используется параметр Nice. Напомню, что Priv, это всего лишь поправка, она в большинстве случаев больше за Nice на 20. Чтобы изменить приоритет процесса просто установите на него курсор и нажимайте F7 для уменьшения числа (увеличения приоритета) или F8 - для увеличения числа.

Но и для решения этой задачи управления процессами Linux необязательно использовать htop. Вы можете сделать все и другими командами. Например, команда nice. С помощью нее вы можете указать приоритет для запускаемого процесса:

nice -n 10 apt-get upgrade

Или изменить приоритет для уже существующего по его pid:

renice -n 10 -p 1343

Завершение процессов в Linux

Если процесс завис и не отвечает, его необходимо завершить. В htop, чтобы убить процесс Linux, просто установите курсор на процесс и нажмите F9:

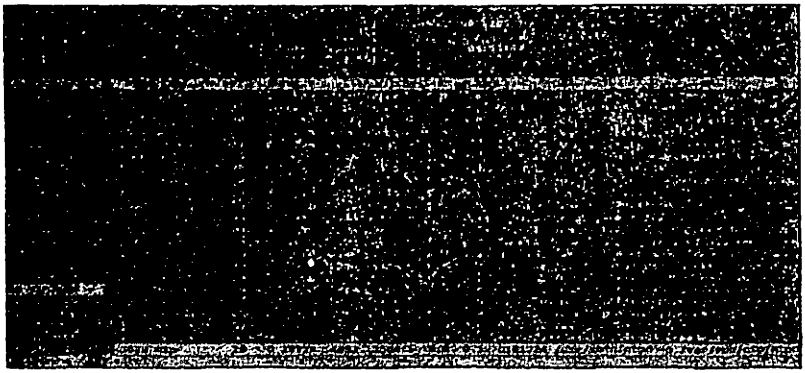

Рисунок 4.10. Принужденное завершение процессов в htop

Система для управления процессами использует определенные сигналы, есть сигналы, которые указывают процессу завершиться. Вот несколько основных сигналов:

- **SIGKILL** - попросить процесс сохранить данные и завершится
- **SIGTERM** - завершить процесс немедленно, без сохранения

Вообще сигналов есть несколько десятков, но мы не будем их рассматривать. Отправим сигнал SIGKILL:

Также можно воспользоваться утилитой kill:

kill -TERM 1943

Также можно уничтожить процесс по имени:

killall chromium

Ограничение процессов

Управление процессами в Linux позволяет контролировать практически все. Вы уже видели что можно сделать, но можно еще больше. С помощью команды ulimit и конфигурационного файла /etc/security/limits.conf вы можете ограничить процессам доступ к системным ресурсам, таким как память, файлы и процессор. Например, вы можете ограничить память процесса Linux, количество файлов и т д.

Запись в файле имеет следующий вид:

<домен> <тип> <элемент> <значение>

- **домен** - имя пользователя, группы или UID
- **тип** - вид ограничений - soft или hard
- **элемент** - ресурс который будет ограничен
- **значение** - необходимый предел

Жесткие ограничения устанавливаются суперпользователем и не могут быть изменены обычными пользователями. Мягкие, soft ограничения могут меняться пользователями с помощью команды ulimit.

Рассмотрим основные ограничения, которые можно применить к процессам:

- **nofile** - максимальное количество открытых файлов
- **as** - максимальное количество оперативной памяти
- **stack** - максимальный размер стека
- **cpu** - максимальное процессорное время
- **nproc** - максимальное количество ядер процессора
- **locks** - количество заблокированных файлов
- **nice** - максимальный приоритет процесса

Например, ограничим процессорное время для процессов пользователя sergiy:

sergiy hard nproc 20

Посмотреть ограничения для определенного процесса вы можете в папке proc:

cat /proc/PID/limits

Max cpu time unlimited unlimited seconds
Max file size unlimited unlimited bytes
Max data size unlimited unlimited bytes
Max stack size 204800 unlimited bytes
Max core file size 0 unlimited bytes
Max resident set unlimited unlimited bytes
Max processes 23562 23562 processes
Max open files 1024 4096 files

Max locked memory 18446744073708503040 18446744073708503040 bytes

Max address space unlimited unlimited bytes

Max file locks unlimited unlimited locks

Max pending signals 23562 23562 signals

Max msgqueue size 819200 819200 bytes

Max nice priority 0 0

Max realtime priority 0 0

Max realtime timeout unlimited unlimited us

Ограничения, измененные, таким образом вступят в силу после перезагрузки. Но мы можем и устанавливать ограничения для текущего командного интерпретатора и создаваемых им процессов с помощью команды ulimit.

Вот опции команды:

- **-S** - мягкое ограничение;
- **-H** - жесткое ограничение;
- **-a** - вывести всю информацию;
- **-f** - максимальный размер создаваемых файлов;
- **-n** - максимальное количество открытых файлов;
- **-s** - максимальный размер стека;
- **-t** - максимальное количество процессорного времени;
- **-u** - максимальное количество запущенных процессов;
- **-v** - максимальный объем виртуальной памяти;

Контрольные вопросы:

1. Как вы понимаете термин процесс?
2. Как выполнить просмотр запущенных процессов?
3. Как выполнить изменение приоритета процессов?
4. Как посмотреть ограничения процессов?
5. Какие сигналы указывают завершению процесса?

ЛАБОРАТОРНАЯ РАБОТА № 5

Тема: Управление памятью в Linux

Цель работы:

Получить краткие теоретические сведения о типах адресов в Linux и ознакомиться картой памяти и структурой page используемых в Linux.

Задание:

Освоить краткие теоретические сведения. С помощью команды */proc/<pid>/maps* посмотреть области памяти процесса, посмотреть поля *struct vm_area_struct* и дать им комментарии.

Теоретическая часть:

типы адресов

Linux- система с виртуальной памятью, а это означает, что адреса, видимые пользовательскими программами, не соответствуют напрямую физическим адресам, используемым оборудованием. Виртуальная память вводит слой косвенности, который позволяет ряд приятных вещей. С виртуальной памятью программы, выполняющиеся в системе, могут выделить гораздо больше памяти, чем доступно физически; более того, даже один процесс может иметь виртуальное адресное пространство больше физической памяти системы. Виртуальная память позволяет также программе использовать разные ухищрения с адресным пространством процесса, в том числе отображение памяти программы в память устройства.

До сих пор мы говорили о виртуальных и физических адресах, но подробности умалчивались. Система Linux имеет дело с несколькими типами адресов, каждый со своей собственной семантикой. К сожалению, в коде ядра не всегда чётко понятно, какой именно тип адреса используется в каждой ситуации, так что программист должен быть осторожным.

Ниже приведён список типов адресов используемых в Linux. Рисунок 5.1. показывает, как эти типы адресов связаны с физической памятью.

Пользовательские виртуальные адреса

Это обычные адреса, видимые программами пространства пользователя. Пользовательские адреса имеют размерность 32 или 64 бита, в зависимости от архитектуры используемого оборудования, и каждый процесс имеет своё собственное виртуальное адресное пространство.

Физические адреса

Адреса, используемые между процессором и памятью системы. Физические адреса 32-х или 64-х разрядные; даже 32-х разрядные системы могут использовать большие физические адреса в некоторых ситуациях.

Адреса шин

Адреса, используемые между периферийными шинами и памятью. Зачастую они такие же, как физические адреса, используемые процессором, но это не обязательно так. Некоторые архитектуры могут предоставить блок управления памятью ввода/вывода (I/O memory management unit, IOMMU), который переназначает адреса между шиной и оперативной памяти. IOMMU может сделать жизнь легче несколькими способами (делая разбросанный в памяти буфер выглядящим непрерывным для устройства, например), но программирование IOMMU является дополнительным шагом, который необходимо выполнить при настройке DMA операций. Конечно, адреса шин сильно зависят от архитектуры.

Логические адреса ядра

Они составляют обычное адресное пространство ядра. Эти адреса отображают какую-то часть (возможно, всю) основной памяти и часто рассматриваются, как если бы они были физическими адресами. На большинстве архитектур логические адреса и связанные с ними физические адреса отличаются только на постоянное смещение. Логические адреса используют родной размер указателя оборудования и, следовательно, могут быть не в состоянии адресовать всю физическую память на 32-х разрядных системах, оборудованных в большей степени. Логические адреса обычно

хранятся в переменных типа unsigned long или void *. Память, возвращаемая *kmalloc*, имеет логический адрес ядра.

Виртуальные адреса ядра

Виртуальные адреса ядра похожи на логические адреса в том, что они являются отображением адреса пространства ядра на физический адрес. Однако, виртуальные адреса ядра не всегда имеют линейную, взаимно-однозначную связь с физическими адресами, которая характеризует логическое адресное пространство. Все логические адреса *являются* виртуальными адресами ядра, но многие виртуальные адреса ядра не являются логическими адресами. Так, например, память, выделенная *vmalloc*, имеет виртуальный адрес (но без прямого физического отображения). Функция *kmap* (описываемая далее в этой главе) также возвращает виртуальные адреса. Виртуальные адреса обычно хранятся в переменных указателей.

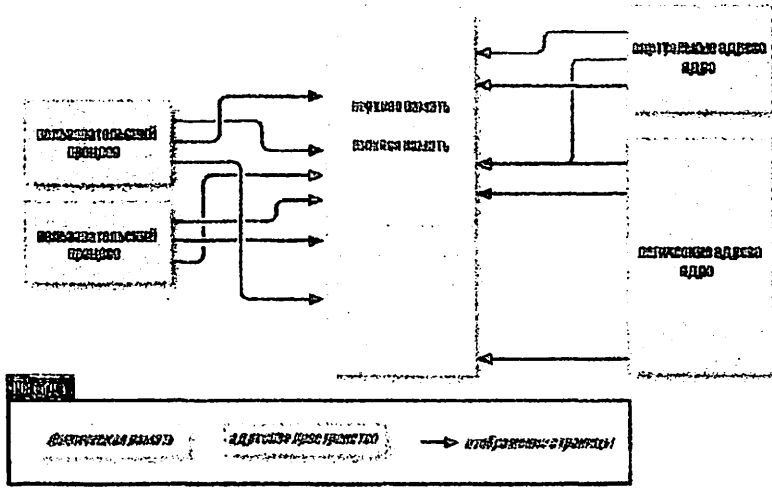

Рисунок 5.1. Типы адресов, используемые в Linux

Если у вас есть логический адрес, макрос *__pa( )* (определённый в *<asm/page.h>*) возвращает соответствующий ему физический адрес.

Физические адреса могут быть преобразованы обратно в логические адреса с помощью ***__va()***, но только для нижних страниц памяти.

Различные функции ядра требуются разные типы адресов. Было бы неплохо, если бы были определены различные типы Си, так, чтобы необходимый тип адреса был бы явным, но этого нет. В этой главе мы стараемся ясно указывать, какой тип адресов, где используется.

Физические адреса и страницы

Физическая память разделена на отдельные модули, называемые ***страницами***. Значительная часть внутренней системной обработки памяти производится на постраничной основе. Размер страницы варьируется от одной архитектуры к другой, хотя большинство систем в настоящее время использует страницы по 4096 байт. Постоянная **PAGE_SIZE** (определённая в ***<asm/page.h>***) задаёт размер страницы для любой архитектуры.

Если вы посмотрите на адрес памяти, виртуальный или физический, он делится на номер страницы и смещение внутри этой страницы. Например, если используются страницы по 4096 байт, 12 младших значащих бит являются смещением, а остальные, старшие биты, указывают номер страницы. Если отказаться от смещения и сдвинуть оставшуюся часть адреса вправо, результат называют ***номером страничного блока*** (page frame number, PFN). Сдвиг битов для конвертации между номером страничного блока и адресами является довольно распространённой операцией; макрос **PAGE_SHIFT** сообщает, сколько битов должны быть смещены для выполнения этого преобразования.

Верхняя и нижняя память

Разница между логическими и виртуальными адресами ядра хорошо видна на 32-х разрядных системах, которые оборудованы большими объёмами памяти. Используя 32 бита можно адресовать 4 ГБ памяти. Однако, Linux на 32-х разрядных системах до недавнего времени был ограничен

значительно меньшей памятью, чем это, из-за способа инициализации виртуального адресного пространства.

Ядро (на архитектуре x86, в конфигурации по умолчанию) разделяет 4 ГБ виртуальное адресное пространство между пространством пользователя и ядром; в обоих контекстах используется один и тот же набор отображений. Типичное разделение выделяет 3 ГБ для пространства пользователя и 1 ГБ для пространства ядра. (* Многие не-x86 архитектуры способны эффективно обходиться без описанного здесь разделения на ядро/пользовательское пространство, так что они могут работать с адресным пространством ядра до 4 ГБ на 32-х разрядных системах. Однако, ограничения, описанные в этом разделе, до сих пор относятся к таким системам, где установлено более 4 ГБ оперативной памяти.) Код ядра и структуры данных должны вписываться в это пространство, но самым большим потребителем адресного пространства ядра является виртуальное отображение на физическую память. Ядро не может напрямую управлять памятью, которая не отображена в адресное пространство ядра. Ядру, другими словами, необходим свой виртуальный адрес для любой памяти, с которой оно должно непосредственно соприкасаться. Таким образом, на протяжении многих лет, максимальным объёмом физической памяти, которая могла быть обработана ядром, было значение, которое могло быть отображено в часть для ядра виртуального адресного пространства, минус пространство, необходимое для самого кода ядра. В результате, базирующиеся на x86 системы Linux могли работать максимально с немногим менее 1 ГБ физической памяти.

В ответ на коммерческое давление, чтобы поддержать больше памяти, не нарушая в то же время работу 32-х разрядных приложений и совместимость системы, производители процессоров добавили в свои продукты функцию "расширение адреса". Результатом является то, что во многих случаях даже 32-х разрядные процессоры могут адресовать более 4 ГБ физической памяти. Однако, ограничение на объём памяти, которая

может быть непосредственно связана с логическими адресами, остаётся. Только самая нижняя часть памяти (до 1 или 2 ГБ, в зависимости от оборудования и конфигурации ядра) имеет логические адреса; (* Ядро версии 2.6 (с дополнительным патчем) может поддерживать на архитектуре x86 режим "4G/4G", который разрешает большие виртуальные адресные пространства ядра и пользовательского пространства при умеренных затратах производительности.) остальная (верхняя память) не имеет. Перед доступом к заданной странице верхней памяти ядро должно установить явное виртуальное соответствие, чтобы сделать эту страницу доступной в адресном пространстве ядра. Таким образом, многие структуры данных ядра должны быть размещены в нижней памяти; верхняя память имеет тенденцию быть зарезервированной для страниц процесса пространства пользователя.

Термин "верхняя память" может ввести некоторых в заблуждение, особенно поскольку он имеет другое значение в мире персональных компьютеров. Таким образом, чтобы внести ясность, мы определим здесь эти термины:

Нижняя память

Память, для которой существуют логические адреса в пространстве ядра. Почти на каждой системе, с которой вы скорее всего встретитесь, вся память является нижней памятью.

Верхняя память

Память, для которой логические адреса не существуют, потому что она выходит за рамки диапазона адресов, отведённых для виртуальных адресов ядра.

На системах i386 граница между нижней и верхней памятью обычно установлена на уровне только до 1 Гб, хотя эта граница может быть изменена во время конфигурации ядра. Эта граница не связана никаким образом со старым ограничением 640 Кб, имеющимся на оригинальном ПК, и её размещение не продиктовано оборудованием. Напротив, это предел,

установленный в самом ядре, так как он разделяет 32-х разрядное адресное пространство между ядром и пространством пользователя.

Карта памяти и структура page

Исторически сложилось, что для обращения к страницам физической памяти ядро использует логические адреса. Однако, добавление поддержки верхней памяти выявило очевидную проблему с таким подходом - логические адреса не доступны для верхней памяти.

Таким образом, функции ядра, которые имеют дело с памятью, вместо этого всё чаще используют указатели на **struct page** (определённой в *<linux/mm.h>*). Эта структура данных используется для хранения информации практически обо всём, что ядро должно знать о физической памяти; для каждой физической страницы в системе существует одна **struct page**. Некоторые из полей этой структуры включают следующее:

atomic_t count;

Число существующих ссылок на эту страницу. Когда количество падает до 0, страница возвращается в список свободных страниц.

void *virtual;

Виртуальный адрес страницы ядра, если она отображена; в противном случае NULL. Страницы нижней памяти отображаются всегда; страницы верхней памяти - обычно нет. Это поле появляется не на всех архитектурах; оно обычно компилируется только тогда, когда виртуальный адрес страницы ядра не может быть легко вычислен. Если вы хотите посмотреть на это поле, правильный метод заключается в использовании макроса ***page_address***, описанного ниже.

unsigned long flags;

Набор битовых флагов, характеризующих состояние этой странице. К ним относятся **PG_locked**, который указывает, что страница была заблокирована в памяти, и **PG_reserved**, который препятствует тому, чтобы система управления памятью вообще работала с этой страницей.

Внутри **struct page** существует гораздо больше информации, но она является частью более глубокой чёрной магии управления памятью и не представляет интерес для авторов драйверов.

Ядро поддерживает один или несколько массивов записей **struct page**, которые позволяют отслеживать всю физическую память системы. На некоторых системах имеется единственный массив, называемый **mem_map**. На других системах, однако, ситуация более сложная. Системы с неоднородным доступом к памяти (nonuniform memory access, NUMA) и другие с сильно разделённой физической памятью могут иметь более одного массива карты памяти, поэтому код, который предназначен для переносимости, должен избегать прямого доступа к массиву, когда это возможно. К счастью, как правило, довольно легко просто работать с указателями **struct page** не беспокоясь о том, откуда они берутся.

Некоторые функции и макросы, определённые для перевода между указателями **struct page** и виртуальными адресами:

struct page *virt_to_page(void *kaddr);

Этот макрос, определённый в ***<asm/page.h>***, принимает логический адрес ядра и возвращает связанный с ним указатель **struct page**. Так как он требует логического адрес, он не работает с памятью от ***vmalloc*** или верхней памятью.

struct page *pfn_to_page(int pfn);

Возвращает указатель **struct page** для заданного номера страничного блока. При необходимости он проверяет номер страничного блока на корректность с помощью ***pfn_valid*** перед его передачей в ***pfn_to_page***.

void *page_address(struct page *page);

Возвращает виртуальный адрес ядра этой страницы, если такой адрес существует. Для верхней памяти этот адрес существует, только если страница была отображена. Эта функция определена в ***<linux/mm.h>***. В

большинстве случаев вы захотите использовать версию *kmap*, а не *page_address*.

```
#include <linux/highmem.h>
void *kmap(struct page *page);
void kunmap(struct page *page);
```

kmap возвращает виртуальный адрес ядра для любой страницы в системе. Для страниц нижней памяти она просто возвращает логический адрес страницы; для страниц верхней памяти *kmap* создаёт специальное отображение в предназначенной для этого части адресного пространства ядра. Отображения, созданные *kmap*, всегда должны быть освобождены с помощью *kunmap*; доступно ограниченное число таких отображений, так что лучше не удерживать их слишком долго. Вызовы *kmap* поддерживают счётчик, так что если две или более функции вызывают *kmap* на той же странице, всё работает правильно. Отметим также, что *kmap* может заснуть, если отображение недоступно.

```
#include <linux/highmem.h>
#include <asm/kmap_types.h>
void *kmap_atomic(struct page *page, enum km_type type);
void kunmap_atomic(void *addr, enum km_type type);
```

kmap_atomic является высокопроизводительной формой *kmap*. Каждая архитектура поддерживает небольшой список слотов (специализированные записи таблицы страниц) для атомарных kmap-ов; вызывающий *kmap_atomic* должен сообщить системе в аргументе type, какой из этих слотов использовать. Единственными слотами, которые имеют смысл для драйверов, являются KM_USER0 и KM_USER1 (для кода, работающего непосредственно из вызова из пользовательского пространства), и KM_IRQ0 и KM_IRQ1 (для обработчиков прерываний).

Обратите внимание, что атомарные kmap-ы должны быть обработаны атомарно; ваш код не может спать, удерживая её.

Таблицы страниц

В любой современной системе процессор должен иметь механизм для трансляции виртуальных адресов в соответствующие физические адреса. Этот механизм называется ***таблицей страниц***; по существу, это многоуровневый древовидный массив, содержащий отображения виртуального к физическому и несколько связанных флагов. Ядро Linux поддерживает набор таблиц страниц даже на архитектурах, которые не используют такие таблицы напрямую.

Многие операции, обычно выполняемые драйверами устройств, могут включать манипуляции таблицами страниц. К счастью для автора драйвера, ядро версии 2.6 ликвидировало всю необходимость непосредственно работать с таблицами страниц. В результате, мы не описываем их детально; любопытные читатели могут захотеть взглянуть на ***Understanding The Linux Kernel*** от Daniel P. Bovet и Marco Cesati (O'Reilly) для полной информации.

Области виртуальной памяти

Область виртуальной памяти (virtual memory area, VMA) представляет собой структуру данных ядра, используемую для управления различными регионами адресного пространства процесса. VMA представляет собой однородный регион в виртуальной памяти процесса: непрерывный диапазон виртуальных адресов, которые имеют одинаковые флаги разрешения и созданы одним и тем же объектом (скажем, файлом, или пространством для своппинга). Она примерно соответствует концепции "сегмента", хотя это лучше описывается как "объект памяти со своими свойствами". Карта памяти процесса состоит (по крайней мере) из следующих областей:

•Область для исполняемого кода программы (часто называемого текстом).

•Несколько областей для данных, в том числе проинициализированные данные (те, которые имеют явно заданные значения в начале исполнения), неинициализированные данных (BSS), (* имя *BSS* является историческим пережитком старого ассемблерного оператора, означающего "блок, начатый символом" ("block started by symbol"). Сегмент BSS исполняемых файлов не сохраняется на диске и ядро отображает нулевую страницу в диапазоне адресов BSS.) и программный стек.

•По одной области для каждого активного отображения памяти.

Области памяти процесса можно увидеть, посмотрев */proc/<pid>/maps* (где *pid*, конечно, заменяется на ID роцесса). */proc/self* является особым случаем */proc/pid*, потому что он всегда бращается к текущему процессу.

Поля в каждой строке:

start-end perm offset major:minor inode image

Каждое поле в */proc/*/maps* (за исключением имени отображения) соответствует полю в структуре **vm_area_struct**:

start

end

Начало и окончание виртуальных адресов для этой области памяти.

perm

Битовая маска с разрешениями для области памяти на чтение, запись и исполнение. Это поле описывает, что процессу разрешено делать со страницами, которые принадлежат этой области. Последний символ в поле - это либо p для "закрытых" ("private"), или s для "общих" ("shared").

offset

Где начинается область памяти в файле, с которым она связана. Смещение 0 означает, что начало области памяти соответствует началу файла.

major

minor

Старший и младший номера устройства удерживающего файл, который был на отображён. Как ни странно, при отображении устройства, старший и младший номера ссылаются на дисковый раздел, содержащий специальный файл устройства, который был открыт пользователем, а не самого устройства.

inode

Номер inode отображённого файла.

image

Имя файла (обычно это исполняемый файл), который был отображён.

Структура vm_area_struct

Когда процесс пользовательского пространства вызывает ***mmap***, чтобы отобразить память устройства в его адресное пространство, система реагирует, создавая новую VMA для предоставления этого отображения. Драйвер, который поддерживает ***mmap*** (и, таким образом, который реализует метод ***mmap***), должен помочь такому процессу завершая инициализацию этой VMA. Автор драйвера должен, следовательно, иметь по крайней мере минимальное понимание VMA для поддержки ***mmap***.

Давайте посмотрим на наиболее важные поля в **struct vm_area_struct** (определённой в ***<linux/mm.h>***). Эти поля могут быть использованы драйверами устройств в их реализации ***mmap***. Заметим, что ядро поддерживает списки и деревья VMA для оптимизации области поиска и некоторые поля **vm_area_struct** используются для поддержки такой организации. Поэтому VMA не могут быть созданы по желанию драйвера, или эти структуры нарушатся. Основными полями VMA являются следующие (обратите внимание на сходство между этими полями и выводом ***/proc***, который мы только что видели):

unsigned long vm_start;

unsigned long vm_end;

Диапазон виртуальных адресов, охватываемый этой VMA. Эти поля являются первыми двумя полями, показываемыми в */proc/*/maps*.

struct file *vm_file;

Указатель на структуру **struct file**, связанную с этой областью (если таковая имеется).

unsigned long vm_pgoff;

Смещение области в файле, в страницах. Когда отображается файл или устройство, это позиция в файле первой страницы, отображённой в эту область.

unsigned long vm_flags;

Набор флагов, описывающих эту область. Флагами, представляющими наибольший интерес для автора драйвера устройства, являются **VM_IO** и **VM_RESERVED**. **VM_IO** отмечает VMA как являющийся отображённым на память регион ввода/вывода. Среди прочего, флаг **VM_IO** мешает региону быть включенным в дампы процессов ядра. **VM_RESERVED** сообщает системе управления памятью не пытаться выгрузить эту VMA; он должен быть установлен в большинстве отображений устройства.

struct vm_operations_struct *vm_ops;

Набор функций, которые ядро может вызывать для работы в этой области памяти. Его присутствие свидетельствует о том, что область памяти является "объектом" ядра, как и **struct file**, которую мы используем везде в этой книге.

void *vm_private_data;

Области, которые могут быть использованы драйвером для сохранения своей собственной информации.

Как и **struct vm_area_struct**, **vm_operations_struct** определена в *<linux/mm.h>*; она включает операции, перечисленные ниже. Эти операции являются единственными необходимыми для обработки потребностей процесса в памяти и они перечислены в том порядке, как они объявлены.

Карта памяти процесса

Последней частью головоломки управления памятью является структура карты памяти процесса, которая удерживает все другие структуры данных вместе. Каждый процесс в системе (за исключением нескольких вспомогательных потоков пространства ядра) имеет **struct mm_struct** (определённую в ***<linux/sched.h>***), которая содержит список виртуальных областей памяти процесса, таблицы страниц и другие разные биты информации управления домашним хозяйством памяти вместе с семафором (**mmap_sem**) и спин-блокировкой (**page_table_lock**). Указатель на эту структуру можно найти в структуре задачи; в редких случаях, когда драйверу необходим к ней доступ, обычным способом является использование **current->mm**. Обратите внимание, что структура управления памятью может быть разделяемой между процессами; к примеру, таким образом работает реализация потоков в Linux.

Контрольные вопросы:

1. Перечислите типы адресов;
2. Какую функцию выполняет карта памяти и структура page?
3. Перечислите области виртуальной памяти;
4. Роль карты памяти процесса;
5. Какую функцию выполняет структура *vm_area_struct?*

ЛАБОРАТОРНАЯ РАБОТА № 6

Тема: Управление сетью в операционной системе Linux

Цель работы:

Научиться управлять настройками сети из консоли в операционной системе Linux и настройке сети через терминал в Ubuntu.

Задание:

Рассмотреть автоматическую настройку сети для Ubuntu и настроить проводную сеть, а также показать примеры передачи файлов.

Теоретическая часть:

Компьютеры обмениваются между собой информацией с помощью пакетов. Все данные в сети интернет передаются с помощью пакетов небольшого размера. Если не углубляться в подробности, то каждый пакет содержит адрес отправителя, адрес получателя и сами данные. Эти адреса есть не что иное, как привычные нам IP-адреса. Кроме IP, у компьютера есть физический адрес, который используется для общения между компьютерами в локальной сети. Это MAC-адрес и задается он производителем сетевой карты.

Как только компьютер подключился к сети, независимо от того проводное это соединение или беспроводное, он может общаться только с компьютерами в локальной сети и только по физическим адресам. Для того, чтобы получить доступ в Глобальную сеть, машине в ней нужно получить IP-адрес. Для этого используется протокол DHCP. Если кратко: наш компьютер спрашивает все компьютеры в локальной сети, кто здесь DHCP-сервер, DHCP ему отвечает и выдаёт IP-адрес. Таким же образом компьютер узнаёт IP маршрутизатора, через который он может получить доступ к Сети, а затем пытается найти DNS-серверы или узнать стандартные у маршрутизатора. С теорией разобрались, перейдем к практике.

Порядок выполнения работы:

Настройка сети через терминал в UBUNTU

Рассмотрим автоматическую настройку сети для Ubuntu 16.04 без Network Manager с помощью стандартных скриптов системы, которые остались от Upstart и пока всё ещё используются. Сначала определим, какие шаги нам нужно предпринять, чтобы всё заработало:

- Включаем сетевой интерфейс и подключаемся к сети;
- Устанавливаем IP-адрес;
- Получаем адреса DNS-серверов.

Эти шаги очень символичны, потом что система всё сделает за нас сама. Нам нужно только выполнить нужные настройки. Посмотрим, какие сетевые интерфейсы подключены к системе. Команда:

ip link show

Но если хотите, можете использовать ifconfig:

ifconfig

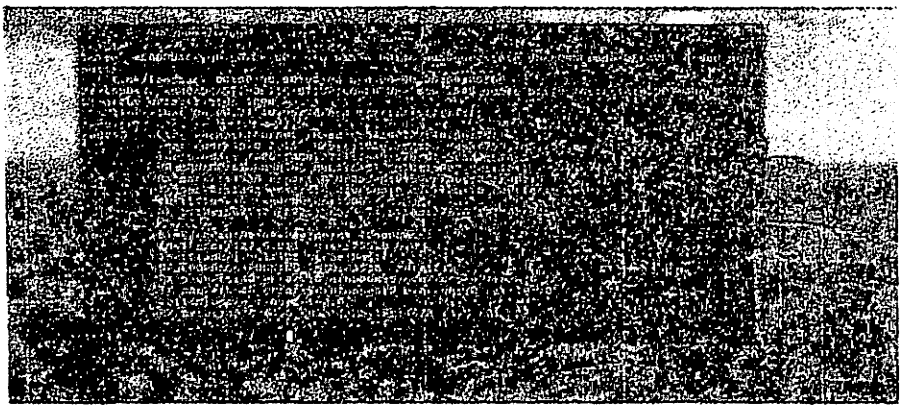

Рисунок 6.1. Просмотр сетевых интерфейсов

В нашей системе только один интерфейс — это enp0s3, есть еще lo, но он виртуальный и указывает на эту машину.

Настройки сети находятся в файле /etc/network/interfaces.

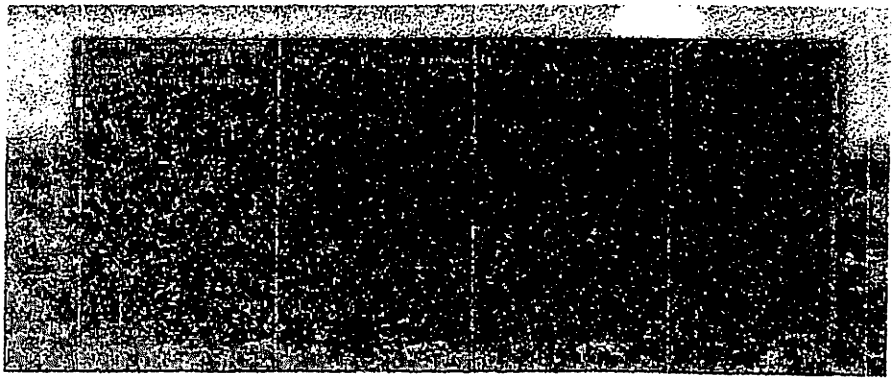

Рисунок 6.2. Виртуальный интерфейс «lo»

В этом файле строчки auto и iface. Первая указывает, что нужно активировать интерфейс при загрузке, вторая же определяет настройки самого интерфейса.

Настройка динамического получения ip-адреса

Добавим в этот файл такие строки, чтобы запускать интерфейс при загрузке и получать IP-адрес автоматически по DHCP:

auto enp0s3

iface enp0s3 inet dhcp

Синтаксис строки auto прост. Он состоит из самой команды и имени сетевого интерфейса. Рассмотрим подробнее:

$ iface интерфейс inet тип

Тип получения IP-адреса может иметь несколько значений, но нас в этой статье будут интересовать только два: dhcp и static.

После завершения настройки сохраните файл и перезапустите сетевой сервис:

sudo service networking restart

Настройка статического адреса ubuntu

При настройке статического IP-адреса компьютер не будет связываться с DHCP-сервером, поэтому здесь придётся указать намного больше параметров.

Содержимое нашего конфигурационного файла будет выглядеть вот так:

auto eth0

iface eth0 inet static

address 192.168.1.7

gateway 192.168.1.1

netmask 255.255.255.0

network 192.168.1.0

broadcast 192.168.1.255

С первыми двумя строчками все понятно, а следующие задают параметры настройки интерфейса:

- address - наш IP-адрес;
- gateway - шлюз, через который будем получать доступ в интернет;
- netmask - маска сети;
- network - адрес сети, имеет тот же адрес, что и шлюз, только с нулем вместо единицы;
- broadcast - широковещательный адрес сети, отправленный на него пакет придет всем компьютерам локальной сети.

Как видим, network и broadcast - это первый и последний IP-адреса сети.

Теперь сохраним файл и перезапустим сеть:

sudo service networking restart

Это была автоматическая настройка локальной сети Ubuntu, но я ещё расскажу, как всё сделать вручную, без конфигурационных файлов.

Ручная настройка сети в UBUNTU

Как и в предыдущем примере, смотрим сетевые интерфейсы:

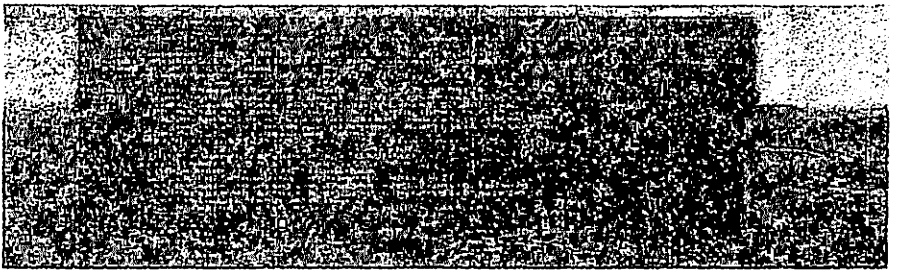

Рисунок 6.3. Просмотр сетевых интерфейсов

После того, как мы узнали интерфейс, можно переходить к настройке.

Получение ip-адреса по dhcp

Сначала включаем интерфейс:

sudo ip link set enp0s3 up

Затем с помощью команды dhclient запрашиваем ip:

sudo dhclient enp0s3

Настройка сети Ubuntu 16.04 завершена, у нас есть IP-адрес, и осталось только настроить DNS, но это мы рассмотрим ниже.

Настройка статического ip

Включаем интерфейс:

sudo ip link set enp0s3 up

Устанавливаем IP-адрес, маску сети и broadcast-адрес для нашего интерфейса:

sudo ip addr add 192.168.1.7/255.255.255.0 broadcast 192.168.1.255 dev enp0s3

Указываем IP-адрес шлюза:

sudo ip route add default via 192.168.1.1

Здесь 192.168.1.7 - наш IP-адрес, 255.255.255.0 - маска сети, 192.168.1.255 - широковещательный адрес.

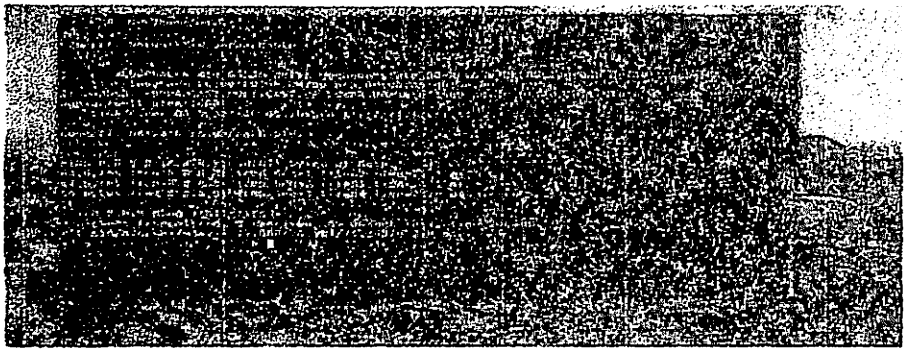

Рисунок 6.4. Настройка статического IP

Настройка DNS

Служба DNS используется для преобразования доменных имен сайтов в IP-адреса. При получении IP-адреса автоматически через DHCP мы используем правильные DNS-серверы, но если мы выбрали статический IP, то DNS можно и не получить, поэтому придётся сделать всё вручную.

Если вам нужно настроить DNS так, чтобы он не сбивался после перезагрузки, необходимо использовать систему настройки сети Ubuntu. Для этого откройте файл /etc/network/interfaces и добавьте в него строчку после директив для нужного интерфейса:

dns-nameservers 8.8.8.8 4.4.4.4

Здесь 8.8.8.8 и 4.4.4.4 это IP-адреса DNS серверов, можете заменить их на свои. И можно использовать один, а не два. Дальше сохраните файл и перезапустите сеть:

sudo service networking restart

Контрольные вопросы:

1. Как настроить сеть через терминал в Ubuntu?
2. Как выполнить настройку статического адреса Ubuntu?
3. Как получить ip-адреса по DHCP?
4. Как выполнить ручную настройку сети в Ubuntu?
5. Для чего используются настройка DNS?

ЛАБОРАТОРНАЯ РАБОТА № 7

Тема: Управление пользователями и группами в Linux

Цель работы:

Формирование умения работать осовными утилитами для добавления пользователя и утилитами управления группами.

Задание:

Создать новые файлы и каталоги, а также разобрать назначение прав доступа к файлам и папкам.

Теоретическая часть:

Добавление пользователя осуществляется при помощи команды *useradd.*

Пример использования:

sudo useradd vasyapupkin

Эта команда создаст в системе нового пользователя vasyapupkin. Чтобы изменить настройки создаваемого пользователя, вы можете использовать следующие ключи:

Таблица 7.1

Ключ	*Описание*
-b	Базовый каталог. Это каталог, в котором будет создана домашняя папка пользователя. По умолчанию /home
-c	Комментарий. В нем вы можете напечатать любой текст.
-d	Название домашнего каталога. По умолчанию название совпадает с именем создаваемого пользователя.
-e	Дата, после которой пользователь будет отключен. Задается в формате ГГГГ-ММ-ДД. По умолчанию отключено.
-f	Количество дней, которые должны пройти после устаревания пароля до блокировки пользователя, если пароль не будет изменен (период неактивности). Если значение равно 0, то запись блокируется сразу после устаревания пароля, при -1 - не блокируется. По умолчанию -1.
-g	Первичная группа пользователя. Можно указывать как GID, так и имя группы. Если параметр не задан будет создана новая группа название которой совпадает с именем пользователя.
-G	Список вторичных групп в которых будет находится создаваемый пользователь

-k	Каталог шаблонов. Файлы и папки из этого каталога будут помещены в домашнюю папку пользователя. По умолчанию /etc/skel.
-m	Ключ, указывающий, что необходимо создать домашнюю папку. По умолчанию домашняя папка не создается.
-p	Зашифрованный пароль пользователя. По умолчанию пароль не задается, но учетная пользователь будет заблокирован до установки пароля
-s	Оболочка, используемая пользователем. По умолчанию /bin/sh.
-u	Вручную задать UID пользователю.

Параметры создания пользователя по умолчанию

Если при создании пользователя не указываются дополнительные ключи, то берутся настройки по умолчанию. Эти настройки вы можете посмотреть выполнив

useradd -D

Результат будет примерно следующий:

GROUP=100

HOME=/home

INACTIVE=-1

EXPIRE=

SHELL=/bin/sh

SKEL=/etc/skel

CREATE_MAIL_SPOOL=no

Если вас не устраивают такие настройки, вы можете поменять их выполнив

sudo useradd -D -s /bin/bash

где,
-s это ключ из таблицы 7.1

Таким образом могут быть заданы параметры, определяемые только ключами: *-b -e -f -g -s*

Изменение пользователя

Изменение параметров пользователя происходит с помощью утилиты usermod. Пример использования:

sudo usermod -c "Эта команда поменяет комментарий пользователю" vasyapupkin

usermod использует те же опции, что и useradd.

Изменение пароля

Изменить пароль пользователю можно при помощи утилиты passwd.

sudo passwd vasyapupkin

passwd может использоваться и обычным пользователем для смены пароля. Для этого пользователю надо ввести

passwd

и ввести старый и новый пароли.

Основные ключи passwd:

Таблица 7.2

Ключ	***Описание***
-d	Удалить пароль пользователю. После этого пароль будет пустым, и пользователь сможет входить в систему без предъявления пароля.
-e	Сделать пароль устаревшим. Это заставит пользователя изменить пароль при следующем входе в систему.
-i	Заблокировать учетную запись пользователя по прошествии указанного количества дней после устаревания пароля.
-n	Минимальное количество дней между сменами пароля.
-x	Максимальное количество дней, после которого необходимо обязательно сменить пароль.
-l	Заблокировать учетную запись пользователя.
-u	Разблокировать учетную запись пользователя.

Установка пустого пароля пользователя

Супер пользователь с помощью утилит командной строки passwd и usermod или путем редактирования файла /etc/shadow может удалить пароль пользователь, дав возможность входить в систему без указания пароля.

sudo passwd -d vasyapupkin

или

sudo usermod -p "" vasyapupkin

Если учетная запись пользователя в этот момент была заблокирована командой passwd -l, то указанные выше команды так же снимут эту блокировку.

Установка пустого пароля может быть полезна как временное решение проблемы в ситуации, когда пользователь забыл свой пароль или не может его ввести из-за проблем с раскладкой клавиатуры. После этого имеет смысл принудить пользователя установить себе новый пароль при следующем входе в систему

sudo passwd -e vasyapupkin

Получение информации о пользователях

- w – вывод информации (имя пользователя, рабочий терминал, время входа в систему, информацию о потребленных ресурсах CPU и имя запущенной программы) о всех вошедших в систему пользователях.
- who – вывод информации (имя пользователя, рабочий терминал, время входа в систему) о всех вошедших в систему пользователях.
- who am i или whoami или id – вывод вашего имени пользователя.
- users – вывод имен пользователей, работающих в системе.
- id имя_пользователя – вывод о идентификаторах пользователя: его uid, имя_пользователя, gid и имя первичной группы и список групп в которых состоит пользователь.
- groups имя_пользователя – вывод списка групп в которых состоит пользователь.

Удаление пользователя

Для того, чтобы удалить пользователя воспользуйтесь утилитой userdel. Пример использования:

sudo userdel vasyapupkin

userdel имеет всего два основных ключа:

Таблица 7.3

Ключ	Описание
-f	Принудительно удалить пользователя, даже если он сейчас работает в системе.
-r	Удалить домашний каталог пользователя.

Управление группами

Создание группы

Программа groupadd создаёт новую группу согласно указанным значениям командной строки и системным значениям по умолчанию. Пример использования:

sudo groupadd testgroup

Основные ключи:

- ***-g*** -установить собственный GID.
- ***-p*** -пароль группы.
- ***-r*** -создать системную группу.

Изменение группы

Сменить название группы, ее GID или пароль можно при помощи groupmod. Пример:
sudo groupmod -n newtestgroup testgroup *#Имя группы изменено с testgroup на newtestgroup*

Опции groupmod:

- ***-g*** -установить другой GID.
- ***-n*** -новое имя группы.
- ***-p*** -изменить пароль группы.

Удаление группы
Удаление группы происходит так:
sudo groupdel testgroup
groupdel не имеет никаких дополнительных параметров.

Файлы конфигурации

Изменять параметры пользователей и групп можно не только при помощи специальных утилит, но и вручную. Все настройки хранятся в текстовых файлах. Описание каждого из них приведено ниже.
/etc/passwd

В файле */etc/passwd* хранится вся информация о пользователях кроме пароля. Одна строка из этого файла соответствует описанию одного пользователя. Примерное содержание строки таково:
vasyapupkin:x:1000:1000:Vasya Pupkin:/home/vpupkin:/bin/bash

Строка состоит из нескольких полей, каждое из которых отделено от другого двоеточием. Значение каждого поля приведено в таблице 7.4

Таблица 7.4

№	Поле	Описание
1	vasyapupkin	Имя пользователя для входа в систему.
2	x	Необязательный зашифрованный пароль.
3	1000	Числовой идентификатор пользователя (UID).
4	1000	Числовой идентификатор группы (GID).
5	Vasya Pupkin	Поле комментария
6	/home/vpupkin	Домашний каталог пользователя.
7	/bin/bash	Оболочка пользователя.

Второе и последнее поля необязательные и могут не иметь значения.

/etc/group

В */etc/group*, как очевидно из названия хранится информация о группах. Она записана в аналогичном */etc/passwd* виде:

vasyapupkin:x:1000:vasyapupkin,petya

Таблица 7.5

№	Поле	Описание
1	vasyapupkin	Название группы
2	x	Необязательный зашифрованный пароль.
3	1000	Числовой идентификатор группы (GID).
4	vasyapupkin,petya	Список пользователей, находящихся в группе.

В этом файле второе и четвертое поля могут быть пустыми.

/etc/shadow

Файл */etc/shadow* хранит в себе пароли, поэтому права, установленные на этот файл, не дают считать его простому пользователю. Пример одной из записей из этого файла:

vasyapupkin:*6Yvp9VO2s$Vfl0t.o754QB3HcvVbz5hlOafmO.LaHXwfavJHniHNzq/bCI3AEo562hhiWLoBSqxLy7RJJNm3fwz.sdhEhHL0:15803:0:99999:7:::*

Здесь:

	Поле	Описание
1	*vasyapupkin*	Имя пользователя для входа в систему.
2	*6Yvp9VO2s$Vfl0t.o754QB3HcvVbz5hlOafmO.LaHXwfavJHniHNzq/bCI3AEo562hhiWLoBSqxLy7RJJNm3fwz.sdhEhHL0*	Необязательный зашифрованный пароль.
3	*15803*	Дата последней смены пароля.
4	*0*	Минимальный срок действия пароля.
5	*99999*	Максимальный срок действия пароля.
6	*7*	Период предупреждения о пароле.
7	-	Период неактивности пароля.
8	-	Дата истечения срока действия учётной записи.

Контрольные вопросы:

1. Перечислите параметры создания пользователя по умолчанию;
2. Перечислите параметры создания пользователя по умолчанию;
3. Как получить информацию о пользователях?
4. Перечислите файлы конфигурации;
5. Как установить пустого пароля пользователя?

СОДЕРЖАНИЕ

СПИСОК ЛИТЕРАТУРЫ

1. Указ Президента Республики Узбекистан «О мерах по дальнейшему развитию высшего образования» № ПП-2909. 20 апреля 2017 г.
2. Указ Президента Республики Узбекистан «О мерах по дальнейшему расширению участия секторов и секторов экономики в повышении качества специалистов высшего образования» 27 июля 2017 г.
3. Р.П.Абдурахманов, Ф.К.Тожиева " Микропроцессоры". Учебник -М: Aloqachi, 2021.-264 с.
4. Р.П.Абдурахманов, Ф.К.Тожиева " Операционные системы". Учебное пособие -М: Aloqachi, 2021.-235 с.
5. Колисниченко, Д.Н. Linux. Полное руководство / Д.Н. Колисниченко, Аллен, Питер В.. - М.: СПб: Наука и Техника, 2017. - 784 с.
6. Колисниченко, Д.Н. Ubuntu Linux 7.04. Руководство пользователя (+DVD) / Д.Н. Колисниченко. - М.: СПб: Питер, 2016. - 189 с.
7. Матросов, В.Л. Операционные системы, сети и интернет-технологии: Учебник / В.Л. Матросов. - М.: Academia, 2017. - 1040 с.
8. Назаров, С.В. Операционные системы. Практикум (для бакалавров) / С.В. Назаров, Л.П. Гудыно, А.А. Кириченко. - М.: КноРус, 2017. - 480 с.
9. Партыка, Т.Л. Операционные системы, среды и оболочки: Учебное пособие / Т.Л. Партыка, И.И. Попов. - М.: Форум, 2015. - 256 с.
10. Партыка, Т.Л. Операционные системы, среды и оболочки: Учебное пособие / Т.Л. Партыка, И.И. Попов. - М.: Форум, 2018. - 256 с.
11. Девис, Т. OpenGL. Руководство по программированию / Т. Девис, Д. Шрайнер, Дж. Нейдер, и др.. - М.: СПб: Питер, 2018. - 624 с.
12. Синицын, С.В. Операционные системы / С.В. Синицын. - М.: Academia, 2016.
13. Спиридонов, Э.С. Операционные системы / Э.С. Спиридонов, М.С. Клыков, М.Д. Рукин и др. - М.: КД Либроком, 2017. - 350 с.
14. Спиридонов, Э.С. Операционные системы / Э.С. Спиридонов, М.С. Клыков, М.Д. Рукин. - М.: КД Либроком, 2015. - 350 с.
15. Таненбаум, Э. Современные операционные системы / Э. Таненбаум. - СПб.: Питер, 2019. - 1120 с.
16. Ўқув адабиётларни ишлаб чиқиш ва нашр этишга тайёрлаш бўйича услубий кўрсатмалар. Агзамов Ф.С., Эргашев А.Қ., Туляганов А.А., Гультураев Н.Х. - Тошкент: Муҳаммад ал-Хоразмий номидаги ТАТУ. 2018. - 52 б.

Методическое пособие разработано для лабораторных работ по предмету " Операционные системы" предназначено для бакалавров направления 5350100 – Телекоммуникационные технологии ("Телекоммуникации", "Телерадиовещание", "Мобильные системы").

Методическое пособие обсуждено на совещании заседании кафедры "АПОСУТ" №___ от ___ _____ 2021 года и рекомендовано для изучения в научно-методическом совете факультета.

Методическое пособие обсуждено на совещании факультета "Телекоммуникационные технологии" №___ от ____ _____ 2021 года и рекомендовано для изучения в научно-методическом совете университета.

Методическое пособие обсуждено на совещании научно-методического совета ТУИТ №___от __ _____ 2021 года и рекомендовано для печати.

Авторы: Х.Х.Ахмедова

Ф.К.Тожиева

Б.У.Акмурадов

Формат 60x84 1/16. Печ.лист 5,5.
Заказ № 96. Тираж 20.
Отпечатано в «Редакционно издательском»
отделе при ТУИТ.
Ташкент ул. Амир Темур, 108.