



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ДГТУ)**

Кафедра Кибербезопасность информационных систем

Разработал: Язвинская Н.Н.

## **ОПЕРАЦИОННЫЕ СИСТЕМЫ Лабораторный практикум – Блок 2**

## **СОДЕРЖАНИЕ**

### **ВВЕДЕНИЕ**

### **ТРЕБОВАНИЯ К ОТЧЕТАМ ПО ЛАБОРАТОРНЫМ РАБОТАМ**

Лабораторная работа №4 Установка и настройка открытой операционной системы на примере Ubuntu Linux

Лабораторная работа №5 Файловые системы ОС Linux

Лабораторная работа №6 Резервное копирование и восстановление системы на примере ОС Ubuntu Linux

Лабораторная работа №7 Администрирование ос ubuntu. Настройка загрузчика

Лабораторная работа №8 Создание дистрибутива сборки специализированной ОС

Лабораторная работа №9 Дебианизация, сертификация и интеграция в хранилище репозитория

Лабораторная работа №10 Конфигурирование и компиляция ядра linux

ПРИЛОЖЕНИЕ А Образец оформления лабораторной работы

ПРИЛОЖЕНИЕ Б Образец оформления титульного листа Журнала лабораторных работ

## ВВЕДЕНИЕ

В данном пособии представлен лабораторный практикум по дисциплине Операционные системы (ОС). Цели лабораторных занятий – это формирование у будущих специалистов 10.05.01 – «Компьютерная безопасность» систематического и целостного представления о значении и месте операционных систем в системном программном обеспечении вычислительных систем, об основных способах инсталляции, настроек и поддержки системных программных продуктов. Задачи лабораторных занятий: практическое освоение пользовательского интерфейса современных операционных систем; изучение взаимодействия аппаратных и программных средств на различных уровнях; изучение различных функциональных компонент современных операционных систем; изучение принципов управления различными ресурсами вычислительной системы и структурами данных.

Для полного освоения курса ОС необходимо последовательно выполнить все задания каждой работы, предварительно ознакомившись с теоретическим материалом курса лекций. Каждая лабораторная работа в данном пособии представляет собой решение отдельной проблемы для операционных систем семейства Unix (в частности, Ubuntu – учебная, свободно распространяемая ОС)

В результате выполнения лабораторных работ по дисциплине ОС у студентов формируются следующие знания и навыки: классификация операционных систем; версии ОС, их преимущества и недостатки; место ОС в составе информационной системы; основные функциональные компоненты ОС; средства мониторинга ОС; способы выбора ОС; способы реализации информационных систем и устройств в ОС; навыки по инсталляции и отладки ОС и ее компонентов, эксплуатации современных ОС и решения поставленных задач в ОС; принципы работы основных подсистем ОС и способы защиты от несанкционированного доступа; принципы построения и разработки ОС, а также методы расширения уже существующих систем; интерфейс прикладного программирования; принципы взаимодействия аппаратных и программных средств на различных уровнях; пользовательский интерфейс современных ОС; навыки по разработке программного обеспечения на базе ОС; принципы анализа и оценки эффективности функционирования ОС и ее компонентов; навыки инсталляции, настройки и администрирования параметров программного обеспечения ОС.

Лабораторные работы, представленные в пособии можно выполнять не только на базе лабораторий университета, но и дома при наличии соответствующей операционной системы на персональном компьютере. По результатам каждой лабораторной работы должен быть сформирован отчет, содержащий все команды и файлы, а также снимок экрана их выполнения. Каждая лабораторная работа содержит краткие теоретические сведения, которые являются дополнительным материалом к курсу лекций. В конце каждой работы есть вопросы для самоконтроля студента.

## **ТРЕБОВАНИЯ ПО ОФОРМЛЕНИЮ ОТЧЕТОВ ЛАБОРАТОРНЫХ РАБОТ**

Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям.

Требования по форматированию:

- шрифт TimesNewRoman
- интервал – полуторный
- поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см
- абзацный отступ – 1,25
- текст должен быть выровнен по ширине.

Отчет должен содержать: номер, тему лабораторной работы, кто выполнил (ФИО, группа) и проверил (ФИО, подпись, дата), цель работы и далее описанный процесс выполнения работы (ХОД РАБОТЫ), образец в Приложении А.

В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним.

Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Окно Windows) и ссылку на него в тексте.

Если в Методике выполнения приводится таблица с заданиями по номерам, то номер задания соответствует списочному номеру студента в группе.

В конце отчета приводятся выводы о проделанной работе (если требуется) и ответ на контрольный вопрос письменно, номер вопроса соответствует номеру студента в списке группы. Ответы на остальные контрольные вопросы готовятся устно.

**Отчет защищается выполнением практических заданий согласно Методике выполнения и устным ответом на контрольные вопросы по выбору преподавателя.**

## **ЛАБОРАТОРНАЯ РАБОТА № 4 (2 часа)**

### **Установка и настройка открытой операционной системы на примере Ubuntu Linux**

**Цель работы** – получение навыков установка и первоначальной настройки открытой операционной системы на примере Ubuntu Linux.

### **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Ubuntu - это современная полнофункциональная операционная система, основанная на ядре Linux. Ubuntu распространяется абсолютно бесплатно. Но при этом устанавливая Ubuntu на свой компьютер вы получаете полный набор всех необходимых для работы приложений, а всё недостающее в стандартной поставке вы сможете легко скачать из интернета. Вы можете использовать Ubuntu и всё доступное в этой системе ПО (программное обеспечение) безо всяких ограничений абсолютно бесплатно и на совершенно законных основаниях. Конечно для Ubuntu существуют и платные программы. Однако в интернет-архивах программного обеспечения Ubuntu, называемых репозиториями, доступно большое количество бесплатных открытых программ на все случаи жизни. Вы даже можете скачать исходный код всех компонент системы и сделать на его основе свой продукт.

Ubuntu поддерживается и спонсируется фирмой Canonical, однако огромный вклад в развитие этой ОС вносит сообщество - обычные люди, которые стремятся улучшить используемые ими приложения и инструменты.

Ubuntu - это один из многих дистрибутивов GNU/Linux, то есть операционных систем, построенных на ядре Linux и на основном программном обеспечении проекта GNU.

Нам потребуется версия Ubuntu Desktop, т.е. базовая версия Ubuntu для домашних компьютеров. Кроме самой Ubuntu существуют: Kubuntu, основанная на рабочем столе KDE; Xubuntu, основанная на рабочем столе XFCE; Ubuntu Netbook Remix, лучше настроенная для работы на нетбуках, и другие. Мало того, кроме Desktop существует ещё несколько различных по предназначению сборок: DVD, Alternate и Server.

Long Term Support - релиз с долгосрочной поддержкой.

Скачать текущую версию ОС Ubuntu можно по ссылке <https://ubuntu.ru/get>

### **1 Разметка диска для Ubuntu**

Для установки Ubuntu, достаточно 25 Гб свободного места. Однако рекомендуется выделить еще хотя бы 15Гб, чтобы было место для сохранения различных документов и других пользовательских файлов.

Надо сразу заметить, что Ubuntu абсолютно без проблем поддерживает диски Windows, поэтому если вы решили оставить Windows на своём компьютере, то вы сможете обращаться из Ubuntu к файлам, хранящимся на разделах Windows.

## 1.2. Разметка винчестера

Разделы винчестера бывают трёх типов: основные, расширенные и логические. Связаны они так: непосредственно винчестер делится на основные разделы, один из основных разделов может быть назначен расширенным и разделён на логические. При этом основных разделов может быть максимум четыре (с учётом расширенного), расширенный, если есть, то всегда один, а логических может быть сколько угодно. Вы можете разрезать винчестер максимум на 4 части, но одну из них вы можете поделить на сколько угодно кусков.

Некоторые программы, например, позволяют вам создать не один расширенный раздел, а несколько. Однако ни Ubuntu, ни уж тем более Windows не увидят логические диски на таких разделах.

## 1.3. Linux и разделы винчестера

Linux очень специфично работает с различными устройствами и источниками данных. Для каждого такого объекта создаётся специальный файл, через который происходит «общение» этого объекта с системой. В частности, подобные файлы есть для винчестеров и разделов на них. И обычно при описании работы с винчестерами и разделами в качестве названий используются как раз имена этих файлов.

Винчестеры называются `sda`, `sdb`, `sdc` и т.д. (`sda` - первый винчестер, `sdb` - второй и далее по аналогии). Подключаемые флешки и другие USB устройства так же идентифицируются, как винчестеры и тоже получают имена вида `sd*`.

Разделы на винчестерах называются так: `sda1`, `sda2`, `sda3` и т.д. Т.е. название раздела состоит из названия винчестера и цифры-номера раздела после него. Первые четыре цифры зарезервированы для основных разделов, а нумерация логических начинается всегда с пяти. Например, рассмотрим такое разбиение винчестера:

- **sda1** - основной
- **sda2** - расширенный
  - **sda5** - логический
  - **sda6** - логический
  - **sda7** - логический
- **sda3** – основной.

Как видно, у нас имеется 2 основных и 3 логических раздела, то есть в операционной системе будет доступно 5 дисков на этом винчестере. При этом четвёртого основного раздела нет, соответственно, нет и специального файла `sda4` в системе.

Обратите внимание, расширенный раздел - это всего лишь контейнер для логических, поэтому из ОС он недоступен и никакие данные на него записать нельзя.

Зачастую существующего объёма оперативной памяти для нормальной работы всех приложений не хватает. В этом случае включается так называемый механизм подкачки, использующий свободное место на винчестере для увеличения

объёма доступной оперативной памяти. Windows для этих целей использует обычные файлы, которые она размещает на доступных ей разделах. Linux тоже умеет так делать, однако из-за неэффективности подобного подхода обычно в Linux всё организовано немного по-другому. Для целей подкачки в Linux используется отдельный раздел со специальной файловой системой, называемый свопом (swap).

Вы ничего не сможете записать на этот раздел, собственно, из системы вы его вообще не увидите, Linux сам управляет работой с ним. Обычно размер свопа выбирается равным объёму оперативной памяти или чуть больше, поскольку swap используется для сохранения состояния компьютера (то есть содержимого оперативной памяти) при использовании спящего режима (он же hibernate).

В принципе, если у вас много оперативной памяти и вам не нужно использовать спящий режим, вы можете отказаться от использования свопа, однако рекомендуется не жалеть лишнего гигабайта-двух на винчестере и создать своп раздел. Правда увлекаться тоже не стоит, выделять под своп слишком много места абсолютно бесполезно.

Но кроме свопа понадобится как минимум раздел для файлов самой системы. Кроме системного раздела рекомендуется создать ещё и раздел для пользовательских документов и настроек. Дело в том, что Ubuntu устроена так, что все пользовательские данные, включая все настройки, полностью отделены от системных файлов и могут быть вынесены на отдельный раздел. Смысл так делать весьма прост: если вы что-то испортите, то всегда сможете переустановить Ubuntu просто отформатировав системный раздел и заново поставив туда систему, при этом вам не придётся особенно мучаться с сохранением настроек и данных, поскольку все они останутся на отдельном разделе.

При этом для системного раздела нам потребуется 15-20 гигабайт, для свопа - столько, сколько у вас оперативной памяти, а для оставшегося раздела под пользовательские данные всё оставшееся свободное место. Программа установки умеет автоматически выполнять разметку, но делает это она не оптимальным образом.

## **1.4 Файловая система Ubuntu**

Все современные операционные системы используют древовидную систему организации файлов. Отличия есть только в местоположении корня этого дерева. Если вы использовали операционные системы Windows, то вы привыкли к так называемым логическим дискам: C:, D:, A: и т.д. В Windows именно они и являются корнями, и все пути к файлам отсчитываются от одного из этих дисков. В Linux по-другому, в нём всегда есть только один корень, который так и называется - root (то есть корень по-английски), а обозначается «/», и путь к любому файлу на компьютере отсчитывается относительно этого корня. Например, /etc/passwd - это путь до файла, в котором хранятся данные обо всех пользователях компьютера.

## **2 Репозитории**

Программы и обновления в Ubuntu устанавливаются преимущественно из

репозиториях. Большая часть ПО в Ubuntu запакована в специальные .deb файлы, в которых содержатся программы и необходимые библиотеки. Репозитории – это специальные сервера-хранилища таких файлов. Пользовательские компьютеры подключаются к репозиториям по сети или через интернет и при помощи специальных утилит (таких как Synaptic) позволяют увидеть, какие пакеты у вас установлены, какие доступны для установки. Большинство утилит поддерживают простой поиск по ключевым словам и способны разбивать группы пакетов по категориям. Использование связки репозиторий-утилита позволяет использовать простой, централизованный метод установки/удаления программ, а также предоставляет удобный способ выкладывания обновлений.

Итак, программы для Ubuntu поставляются в виде так называемых deb-пакетов. Deb-пакет - это обычный архив, содержащий файлы устанавливаемого приложения и различную вспомогательную информацию. Поставить программу из deb-пакета кликнув по нему два раза левой кнопкой мыши. Появится окно установки с описанием программы, рисунок 1.

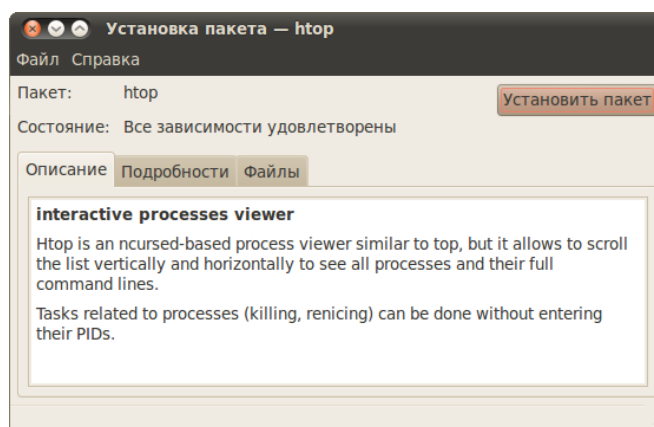


Рисунок 1 - Окно установки программы

Дальше вам надо будет нажать на кнопку «Установить пакет», ввести свой пароль и дождаться окончания процесса установки. Для установки новых приложений в Ubuntu вы должны обладать правами администратора. Однако это только самый простейший случай, который используется крайне редко. Правильней устанавливать программы через репозитории. Прямой установки deb-пакетов стоит избегать. Вместо этого гораздо предпочтительней использовать репозитории. Если же нужного вам приложения в репозиториях нет, то в любом случае устанавливайте пакеты, скачанные только с официальных сайтов разработчиков, иначе рискуете получить вредоносную программу, вместо желаемой или же вместе с ней.

Основное отличие deb-пакетов от программ-установщиков приложений из Windows заключается в так называемых зависимостях. Зависимость это то, что должно стоять в системе для обеспечения работы устанавливаемого приложения. Если система не может разрешить зависимости, то новый пакет не будет установлен. На практике такого никогда не случается, потому как Ubuntu всегда автоматически разрешает все зависимости и скачивает недостающие пакеты из интернета без участия пользователя. Если у вас нет интернета, то зависимости



придётся разрешать вручную.

Репозиторий в интернете - это специальный сайт с архивом пакетов и вспомогательной информацией. Каждый репозиторий имеет так называемый *индекс* - список всех доступных в нём пакетов с указанием их версий, зависимостей и прочей полезной информации. Ubuntu периодически или же по запросу скачивает со всех прописанных в системе репозиториях их индексы. И когда нужно выполнять какие-либо операции с пакетами Ubuntu ориентируется именно на эти сохранённые в ней описания репозиториях.

Например, когда вы просите Ubuntu установить какую-либо программу, она просматривает все сохранённые индексы и ищет из какого репозитория можно скачать последнюю версию запрашиваемого приложения, а также все его зависимости. И только после успешного завершения этого процесса начинается непосредственно загрузка пакетов из репозиториях на компьютер и последующая их установка. Кроме того, весь поиск пакетов и любая информация о доступных приложениях так же основываются на сохранённых индексах. То есть механизм очень простой: система сначала скачивает всю доступную информацию обо всех доступных приложениях, а потом по запросу показывает нужные данные пользователю или же сама использует их для некоторых операций.

Существуют:

- `getdeb` - репозиторий с новыми версиями программ для Ubuntu
- `playdeb` - репозиторий с новыми версиями игр для Ubuntu
- `launchpad` - сайт с множеством персональных репозиториях для Ubuntu.

Установочные файлы:

- `deb` (`deb` пакет) - архив `ar` с бинарными файлами, если изменить расширение на `ar` и открыть архиватором, то можно увидеть его структуру:

- `debian-binary`: версия формата `deb`-пакета. Она равняется «2.0» для текущих версий Ubuntu

- `control.tar.gz`: информация о пакете (описание, зависимости, электронная почта сопровождающего пакет), контрольные суммы устанавливаемых файлов, скрипты выполняющиеся после установки или после удаления

- `data.tar` или `data.tar.gz` или `data.tar.bz2` или `data.tar.lzma`: устанавливаемые файлы

- `run`, `sh` - исполняемый файл (в свойствах файла будет указано, что это исполняемый файл), содержит исполняемый код и архив с файлами, используется для установки программ и драйверов.

- `sh` - текстовый файл с командами для установки (в свойствах файла будет указано, что это текстовый файл).

- `tar.gz`, `tar.bz2`, `tar.xz` - архив, в котором могут лежать бинарные файлы или исходный код.

В Ubuntu всё программное обеспечение делится на четыре секции, называемые компонентами, чтобы отразить разницу в лицензии и уровне доступной поддержки.

Пакеты распределяются по компонентам таким образом:

- **Main** – свободное ПО, официально поддерживаемое компанией Canonical.
- **Restricted** – проприетарное ПО (в основном — драйверы устройств), официально поддерживаемое компанией Canonical.
- **Universe** – свободное ПО, официально не поддерживаемое компанией Canonical (но поддерживаемое сообществом пользователей).
- **Multiverse** – проприетарное ПО, не поддерживаемое компанией Canonical.

Существует четыре основных репозитория Ubuntu:

- **\$release** – это пакеты на момент выхода релиза.
- **\$release-security** – пакеты критических обновлений безопасности.
- **\$release-updates** – пакеты обновления системы (т.е. более поздние версии ПО, вышедшие уже после релиза).
- **\$release-backports** – бэкпорты более новых версий некоторого ПО, которое доступно только в нестабильных версиях Ubuntu.
- **partner** – репозиторий содержащий ПО компаний-партнеров Canonical.

Кроме официальных, существует множество репозиториях от авторов программ и от тех, кто не поленился собрать из исходников пакет и поделиться им с другими. Launchpad предлагает создавать РРА-репозитории — Personal Package Archive, обычно небольшой репозиторий, в который его хозяин складывает исходники, а пользователи на выходе получают уже готовый deb-пакет.

В Ubuntu существуют две основные графические утилиты управления программами: «Центр приложений Ubuntu» и «Менеджер пакетов Synaptic».

## 2.1 Работа репозитория

Пакет (например \*.deb файл) размещается на общедоступном интернет-ресурсе (например, [archive.ubuntu.com](http://archive.ubuntu.com)). Затем информация о пакете заносится в файл Packages, который, в свою очередь, для удобства работы пакуется в Packages.gz

Файлов Packages.gz может быть несколько (например, по одному для каждой архитектуры). Файл Release содержит описание репозитория в целом и ссылки на различные Packages.gz

Общая же схема работы:

1. Пользовательский компьютер подключается к репозиторию, и при наличии защиты, проверяет его истинность.
2. Читает файл Release, находит и скачивает необходимые Packages.gz
3. На основе скачанных Packages.gz обновляет локальную базу данных пакетов.
4. Теперь пользовательский компьютер «знает» где находится тот или иной пакет и при необходимости легко может его скачать и установить.

## 2.2 Подключение репозитория с помощью графического интерфейса

Некоторые репозитории помимо нужных пакетов могут содержать экспериментальные сборки различного системного ПО, в том числе и ядер linux.

Так как версия этих экспериментальных пакетов как правило выше, чем установленная, *Менеджер обновлений* может попытаться «обновить» систему с этих репозиториев, что в свою очередь может повредить вашу систему. Поэтому внимательно читайте описание подключаемого репозитория и информацию в *Менеджере обновлений*.

Для подключения репозитория установите утилиту Synaptic.

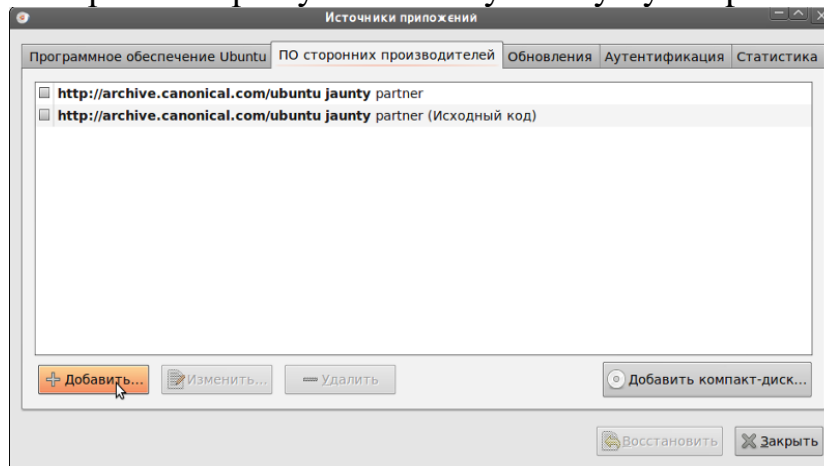


Рисунок 2 – Окно подключения репозитория

1. В появившемся окне заполните поле «*Строка APT:*» и нажмите кнопку «*Добавить источник*», рисунок 3.

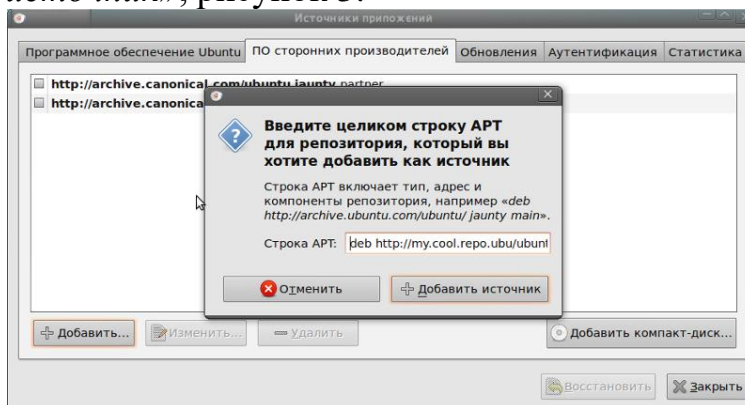


Рисунок 3 – Добавление источника

2. Источник будет добавлен и включен, нажмите кнопку «*Закрыть*».

Так как был подключен новый источник программного обеспечения, необходимо обновить информацию о пакетах. Появится окно, с предложением это сделать, нажать «*Обновить*», рисунок 4.

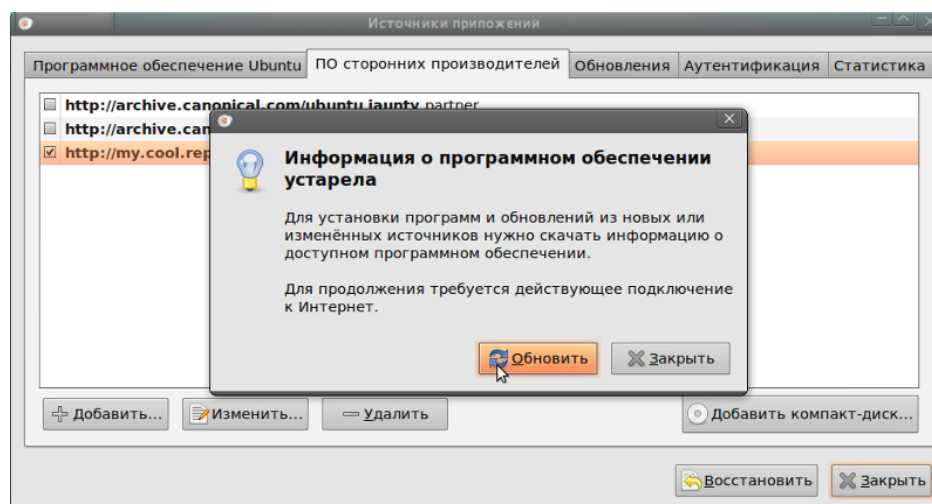


Рисунок 4 – Обновление ПО

После обновления информации о пакетах окно «*Источники приложений*» закроется, и скорее всего вы получите ошибку о неподписанном источнике приложений, тем не менее, вы сможете устанавливать пакеты, содержащиеся в подключенном репозитории стандартными средствами. Для устранения ошибки неподписанного репозитория см. пункт про защиту репозитория ниже.

Обратите также внимание на кнопку «Подробнее», появляющуюся при выборе каждой программы из списка. Нажав на неё, вы попадете на вкладку с описанием приложения, на которой кроме всего прочего есть кнопка Установки/Удаления и скриншот, а также в самой нижней строчке версия программы и основной пакет.

Дело в том, что программа  $\neq$  пакету, в состав одного приложения может входить несколько пакетов. При этом всегда есть основной, который отвечает за установку всех других.

## 2.3 Защита репозитория

Поскольку репозитории большей частью расположены в интернете, существует вероятность подмены репозитория злоумышленником на свой, содержащий модифицированные пакеты. Таким образом, пользователь может установить себе модифицированный пакет и тем самым поставить безопасность своей системы под угрозу. Многие репозитории имеют защиту от подмены. Такая защита реализована при помощи сверки цифровых подписей репозитория и клиента. В случае, когда репозиторий имеет цифровую подпись, а пользовательский компьютер содержит открытый ключ для этого репозитория — такой репозиторий считается доверенным.

В Ubuntu по умолчанию доверенными являются репозитории на установочных дисках и основные интернет репозитории — `archive.ubuntu.com`. При наличии на пользовательском компьютере нескольких подключенных репозитория, предпочтение отдается доверенным.

При подключении репозитория, защищенного цифровой подписью Вам нужно скачать (обычно с ресурса, рассказывающего про этот репозиторий, или с сервера ключей, что является более предпочтительным в любом случае) открытый ключ и добавить его в систему. Иногда для скачивания предоставляется доступный для установки пакет, который в свою очередь при своей установке сам прописывает ключ репозитория. Если вы скачиваете ключ с сайта репозитория, то вы получите обычный файл с расширением .key, .gpg или другим. Добавить его в систему можно так:

```
sudo apt-key add repo.key
```

где repo.key — полученный вами ключ репозитория.

Или при помощи графического интерфейса — запустите «*Источники приложений*» (Система→Администрирование→Источники приложений), перейдите на вкладку «*Аутентификация*» и нажмите на кнопку «*Импортировать файл ключа...*» - откроется диалог выбора файла. Выберите файл ключа и нажмите ОК, рисунок 5.

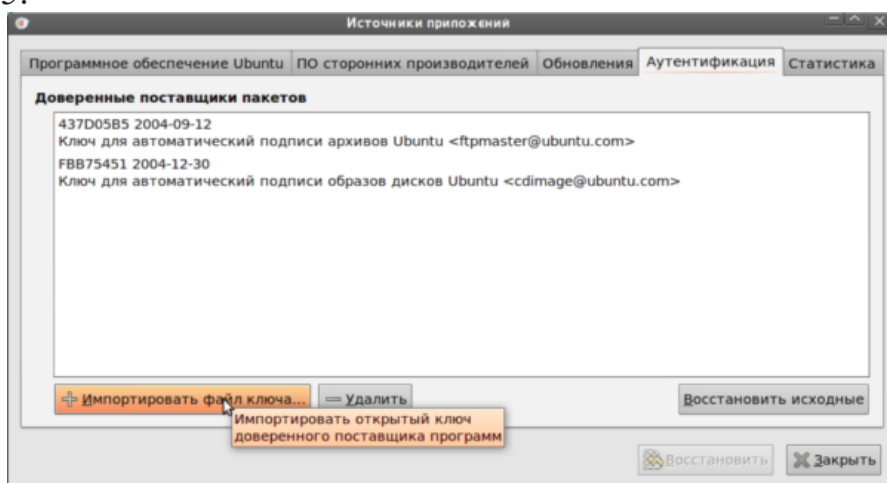


Рисунок 5 – Источники приложений

Однако гораздо более предпочтительным является добавление ключа со специального защищённого сервера. Обычно, когда заходит речь о ключе, даётся его непонятный с первого взгляда буквенно-цифровой идентификатор вида 123ABCDEF456 (строка из произвольных цифр и букв латинского алфавита в верхнем регистре). Это - уникальное имя (идентификатор) ключа. Иногда ключ описывается строкой вида 1024R/123ABCD, тогда идентификатором является часть после слеша. Так вот, ключи преимущественно хранятся на специальных серверах, откуда любой может их получить. Ключи для репозитория Ubuntu принято хранить на [keyserver.ubuntu.com](http://keyserver.ubuntu.com). Для получения и импортирования в систему ключа с сервера необходимо выполнить команду:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 0x12345678
```

Где вместо [keyserver.ubuntu.com](http://keyserver.ubuntu.com) можно подставить адрес другого сервера ключей, а вместо 12345678 необходимо написать идентификатор нужного вам ключа.

Для того, чтобы разом попытаться импортировать все недостающие ключи

репозиториях, выполните в консоли:

```
sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com `sudo aptitude  
update 2>&1 | grep -o '[0-9A-Z]\{16\}' | xargs`
```

## 2.4 Работа менеджера

Менеджер пакетов Synaptic позволяет полностью управлять отдельными пакетами в системе. Основное его отличие от Центра приложений, кроме более богатого функционала, в том, что он работает на уровне пакетов, а не приложений. С помощью Synaptic вы можете устанавливать, удалять, настраивать и обновлять пакеты в вашей системе, просматривать списки доступных и установленных пакетов, управлять репозиториями и обновлять систему до новой версии. Каждое приложение состоит из одного или более пакетов.

Установить его можно нажав на ссылку слева или введя в терминале команду:

```
sudo apt-get install synaptic
```

Перед запуском программы вы увидите окно, в которое вам нужно будет ввести свой пароль, для дальнейшей работы с приложением.

Для запуска Synaptic откройте Главное меню и наберите в поиске synaptic. Также Synaptic можно запустить введя в терминале команду:

```
sudo synaptic
```

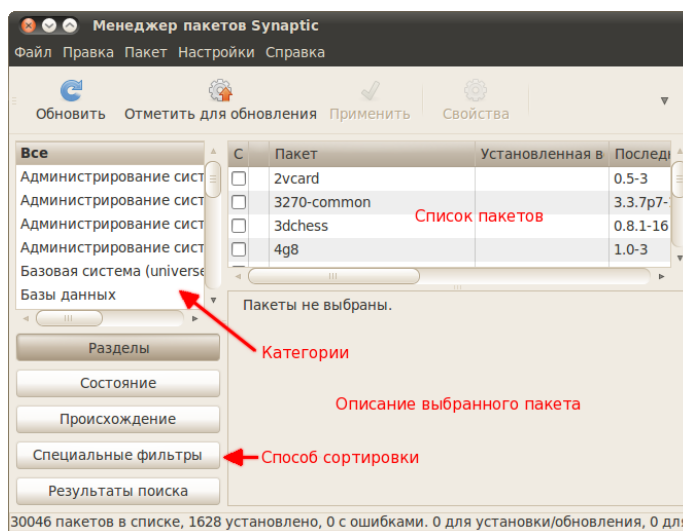


Рисунок 6 - Synaptic

Также на верхней панели есть строка поиска, а кроме неё кнопки, позволяющие совершать некоторые операции. При нажатии на кнопку «Обновить» будет произведено обновление индексов всех репозиториях, при нажатии на кнопку «Отметить для обновления» собственно будут отмечены для обновления все пакеты, для которых доступны новые версии, а кнопка «Применить» нужна для применения всех внесённых изменений.

Установленные пакеты помечаются зелёными квадратиками, а неустановленные - белыми. Изменить состояние того или иного пакета можно

нажав правой кнопкой мыши на его названии в списке и выбрав нужное действие, рисунок 7.

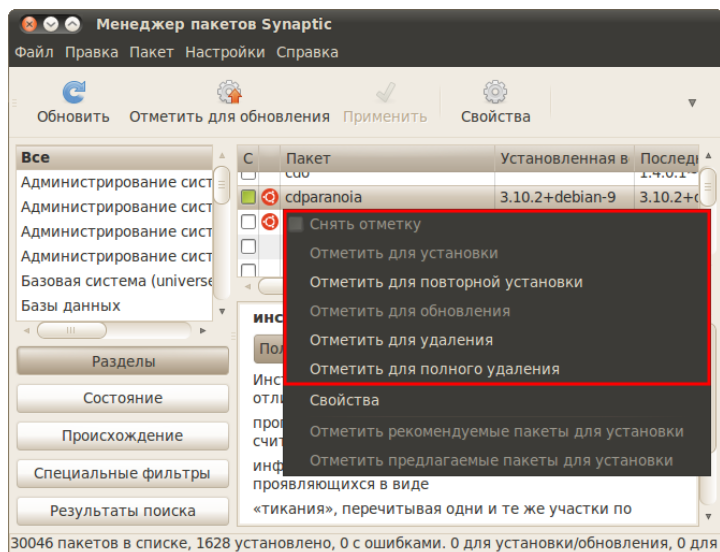


Рисунок 7 – Состояние пакета

Внесённые через Synaptic изменения вступают в силу только после нажатия на кнопку «Применить» на панели инструментов.

Удалить пакет можно одним из двух способов: либо просто удалить файлы пакета, либо удалить их вместе со всеми пользовательскими настройками, относящимися к удаляемому пакету. Отличаются эти способы следующим: многие программы создают в домашних папках пользователей файлы со своими настройками, так вот, при простом удалении эти программы удалятся без пользовательских настроек, а при полном - с ними.

Synaptic, как и остальные инструменты управления пакетами, автоматически следит за разрешением всех зависимостей и ликвидацией различных конфликтов. При совершении любых действий Synaptic выдаст вам окно с подробным описанием вносимых изменений.

От пакета могут зависеть два других, поэтому Synaptic выдаёт предупреждение, что удалить их можно только вместе, рисунок 8.

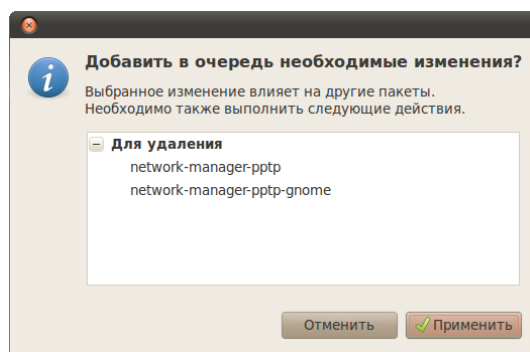


Рисунок 8 – Окно предупреждения

После нажатия на неё Synaptic выдаст вам всю сводку планируемых действий, и вы сможете проверить, что всё будет сделано именно так, как вы хотите. И только после этого собственно запустить процесс внесения изменений нажатием кнопки «Применить», рисунок 9.



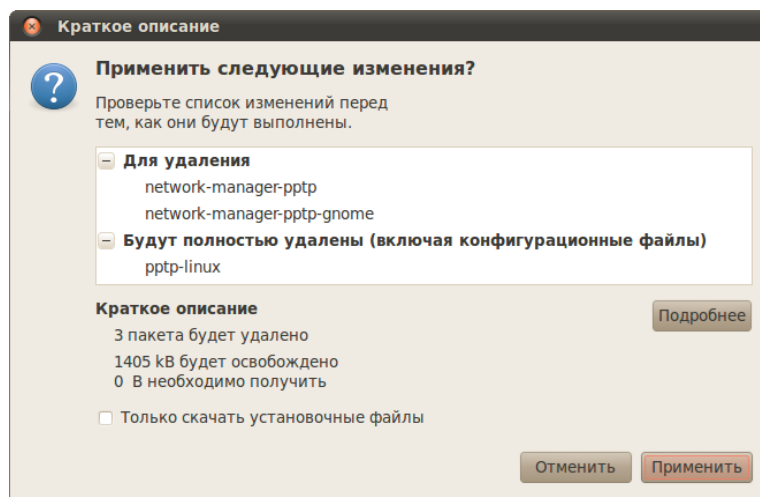


Рисунок 9 – Сводка планируемых действий Synaptic

Вся система состоит из пакетов. Так вот, через Synaptic вы можете удалить любой из них, например, ядро Ubuntu. Подобные действия приведут к полной неработоспособности системы, поэтому никогда не удаляйте пакеты, назначение которых вы не знаете.

Обновление:

- Щелкните по кнопке *Обновить* или нажмите **Ctrl+R** для того чтобы скачать список самых последних версий ПО.
- Правый клик на нужном пакете и выберите в появившемся меню *Отметить для обновления*.
- Для обновления, нажмите кнопку *Применить* на главной панели Менеджера пакетов Synaptic.

Synaptic предоставляет два варианта обновления системы:

#### 1. Умное обновления (рекомендуется)

Умное обновление попытается разрешить конфликты пакетов перед обновлением системы. Действие умного обновления аналогично действию команды `apt-get dist-upgrade`.

#### 2. Стандартное обновление

Стандартное обновление обновит только те пакеты, которые не требуют установки дополнительных зависимостей.

По умолчанию unaptic использует умное обновление. Для того чтобы изменить метод обновления системы откройте *Настройки→Параметры→Основные* и выберите требуемый способ в *Обновление системы*.

- Щелкните по кнопке *Обновить* или нажмите **Ctrl+R** для того чтобы скачать список самых последних версий ПО.
- Нажмите на кнопку *Отметить для обновления* или нажмите **Ctrl+G** для того, чтобы Synaptic отметил для обновления все пакеты.
- Для обновления, нажмите кнопку *Применить* на главной панели Менеджера пакетов Synaptic.



Восстановление «сломанных пакетов».

«Сломанные пакеты» - это пакеты которые имеют неудовлетворённые зависимости. Если сломанные пакеты обнаружены, то Synaptic не позволит проводить никаких изменений в системе с пакетами до тех пор, пока все сломанные пакеты не будут исправлены.

Для исправления сломанных пакетов:

1. Выберите *Правка→Исправить пакеты с ошибками* в главном меню.
2. Выберите *Внести отмеченные изменения* в меню *Правка* или нажмите **Ctrl+P**
3. Подтвердите изменения, щелкнув по кнопке *Применить*.

## 2.5 Настройка кэширования пакетов

Системный список репозиториев содержится в файле `/etc/apt/sources.list`. Для того, чтобы добавить репозиторий - отредактируйте этот файл, например так:

**sudo nano /etc/apt/sources.list**

и добавьте туда АРТ строку. Чем «выше» (т.е. ближе к началу файла) стоит строка, тем больший приоритет получит добавленный репозиторий.

Сохраните файл и закройте редактор. Для *nano* нужно нажать **Ctrl+X**, подтвердить сохранение изменений - **Y** и убедившись, что имя сохраняемого файла `/etc/apt/sources.list` нажать **Enter**.

Далее следует обновить список пакетов. Для этого выполните:

**sudo apt-get update**

Теперь можно устанавливать пакеты из нового репозитория, правда, для комфортной работы вам может быть необходимо импортировать в систему ключ репозитория, т.к. у вас постоянно будет появляться такое предупреждение.

Кроме того, при установке пакетов вам будут сообщать о том, что они являются ненадёжными.

Настройки кэширования пакетов находятся в двух местах. Первое – в самом synaptic, окно с настройками кэширования вызывается через меню *Настройки→Параметры→Файлы*, рисунок 10.

Кэш хранится в папке `/var/cache/apt/archives/`

- *Хранить все загруженные файлы в кэше* - система будет хранить все файлы, скачанные из репозиториев в кэше, согласно правилам хранения.

- *Удалять загруженные файлы после установки* - система будет удалять файлы, скачанные из репозиториев сразу после их установки.

- *Удалять только те пакеты, которые более недоступны в репозитории* - система будет хранить все пакеты, скачанные из репозиториев, удаляя лишь те, которые будут удалены в репозитории согласно правилам.

- *Очистить кэш* - полностью очищает кэш.

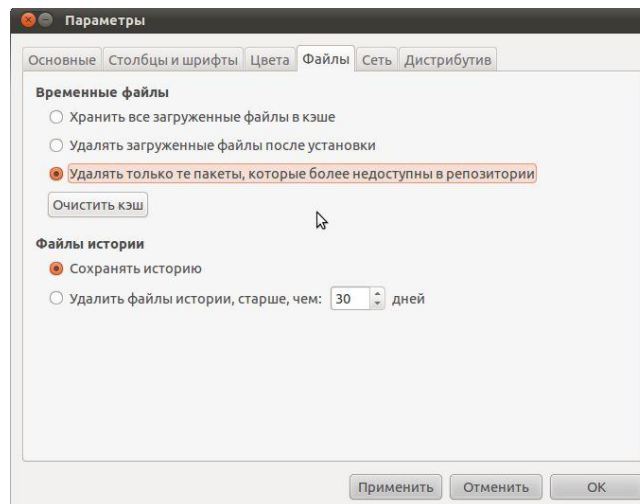


Рисунок 10 - Настройки кэширования пакетов

В разделе *Файлы истории* устанавливаются настройки времени хранения истории установок, либо отключается удаление этой истории вообще.

Второе место, в котором хранятся настройки кэширования устанавливает правила хранения пакетов в кэше. Для того, чтобы получить доступ к этим настройкам необходимо в терминале выполнить следующую команду:

```
sudo nano /etc/apt/apt.conf.d/20archive
```

Эти настройки имеют следующий вид:

```
APT::Archives::MaxAge "30";
```

```
APT::Archives::MinAge "2";
```

```
APT::Archives::MaxSize "500";
```

Здесь отображаются максимальный, минимальный срок хранения пакета в днях и максимальный размер кэша в мегабайтах. В данном случае все пакеты старше 30 дней будут удалены, даже если размер кэша менее 500 мб, и ни один пакет младше 2 дней удален не будет, даже если размер кэша превысит 500 мб.

### 3 Установка приложений

Установка из командной строки позволяет получить больше информации о процессе установки и позволяет гибко его настраивать, хотя и может показаться неудобной начинающему пользователю.

Запустите терминал. Обновить данные о доступных в репозиториях программах можно командой:

```
sudo apt-get update
```

По запросу введите пароль. Учтите, что при вводе в терминале пароль не отображается, ни звёздочками, ни кружками, никак. Это нормально. Для установки нужной программы введите команду:

```
sudo apt-get install имя-программы
```

Например:

```
sudo apt-get install nmap
```

Если нужно установить несколько программ, то их можно перечислить через пробел, например:

```
sudo apt-get install nmap
```

Если потребуется - ответьте на задаваемые вопросы (для положительного ответа нужно ввести Y или N). Программа будет установлена, если она уже установлена - она будет обновлена.

Не все программы входят в основные репозитории Ubuntu. Поэтому вам придется вручную подключать необходимые репозитории с нужными вам программами или пакетами. Для поиска программы в списке доступных пакетов воспользуйтесь командами:

```
sudo apt-cache search keyword
```

где keyword - название программы, часть названия программы или слово из её описания.

Удалить приложение

```
sudo apt remove nmap
```

### **3.1 Установка определенной версии пакета**

Ищем нужную версию пакета:

```
apt-cache showpkg имя_пакета
```

Устанавливаем нужную версию пакета:

```
apt-cache showpkg имя_пакета
```

```
apt-get install имя_пакета=версия
```

### **3.2 Установка из deb-пакета**

Если нужной программы нет в основном репозитории, и у автора программы нет своего репозитория, либо если репозитории недоступны (например, нет интернета), то программу можно установить из deb-пакета (скачанного заранее/принесённого на USB накопителе/...). Если deb-пакет есть в официальном репозитории, то его можно скачать с сайта <http://packages.ubuntu.com>. Часто deb-пакет можно скачать с сайта самой программы. Можно также воспользоваться поиском на сайте <http://getdeb.net>. Минус такого подхода - менеджер обновлений не будет отслеживать появление новых версий установленной программы.

#### **С использованием графического интерфейса**

Перейдите при помощи Nautilus в папку, где находится deb-пакет, откройте свойства файла (правая клавиша → Свойства), во вкладке «Права» разрешите выполнение файла (галочка у «Разрешить исполнение файла как программы»). Далее закрываем свойства файла, и по двойному щелчку Nautilus предложит нам открыть код или выполнить файл. Запускаем. Либо возможно это сделать специальным установщиком GDebi (установить можно из Центра приложений, вписав в поиск GDebi, либо вписав в командную строку:

```
sudo apt-get install GDebi
```

После установки запускаем deb-пакет с помощью установщика программ GDebi все, что от вас потребуется - это просто нажать кнопку «Установить пакет».

Возможные ошибки:

1. Пакет не может быть установлен. Например, он предназначен для другой архитектуры.

2. В системе отсутствуют необходимые устанавливаемому приложению пакеты. В таком случае «Установщик программ GDebi» автоматически попытается получить нужные пакеты из репозитория. Или же вы можете самостоятельно скачать требуемые пакеты и установить их.

#### **С использованием командной строки**

Запустите терминал (Меню: Приложения - Стандартные - Терминал). Установка выполняется с помощью программы dpkg

```
sudo dpkg -i /home/user/soft/ntlmaps_0.9.9.0.1-10_all.deb
```

Нужно будет ввести свой пароль. Не забывайте, что при вводе в терминале пароль не отображается. Заметьте, что при использовании dpkg нужно ввести полное имя файла (а не только название программы). Прочитайте, что dpkg выводит в терминал - там будет либо сообщение об успешной установке, либо описание ошибки (например, неудовлетворённые зависимости). Можно одной командой установить сразу несколько пакетов, например, следующая команда установит все deb-пакеты в директории:

```
sudo dpkg -i /home/user/soft/ntlmaps_*.deb
```

Это бывает полезно для установки пакета программы вместе с пакетами зависимостей.

#### **Запрет обновления пакета**

Бывает, когда ненужно ставить версию пакета новее установленной. К примеру отсутствует поддержка чего либо в новой версии либо, она не корректно работает на системе.

Через dpkg

В терминале:

```
sudo echo 'имя_пакета hold' | sudo dpkg --set-selections
```

имя пакета берется из команды, которой вы ставили этот пакет. Чтобы разрешить обновлять делаем так:

```
echo 'имя_пакета install' | sudo dpkg --set-selections
```

Смотрим статус пакета:

```
dpkg --get-selections | grep 'имя_пакета'
```

Через apt

Здесь все проще. Чтобы заблокировать пакет:

```
sudo apt-mark hold имя_пакета
```

Чтобы разблокировать пакет:

```
sudo apt-mark unhold имя_пакета
```

Через aptitude аналогично.

### **3.3 Установка программ с собственным инсталлятором из файлов sh, run**

Иногда программы могут распространяться с собственным инсталлятором. Это ничем не отличается от ситуации в Windows. Только здесь, распаковав tar.gz архив с дистрибутивом программы, вы вместо setup.exe увидите что-то наподобие install.sh. Это заранее собранный пакет ПО, который оформлен в виде скрипта или бинарника, он берёт на себя работу по размещению файлов в нужных местах и прописыванию нужных параметров. При этом пропадает возможность управлять

таким ПО с помощью пакетного менеджера. Пользоваться такими пакетами нежелательно, но если выбора нет, то переходим в директорию с файлом, например:

```
cd ~/soft
```

Разрешаем выполнять этот файл:

```
chmod +x install.sh
```

Запускаем его:

```
sudo ./install.sh
```

Иногда программу можно установить и без прав суперпользователя (без sudo), но это, скорее, исключение.

Иногда дистрибутив программы распространяется в виде самораспаковывающегося архива. В таком случае это будет просто один единственный файл .sh который и нужно запустить. Дальше вы просто получите мастер где нужно будет ответить на ряд вопросов, так же как это делается в Windows. Так устанавливаются официальные драйверы nVidia, ATI, среда разработчика NetBeans и т.п.

Есть программы, которые не нуждаются в инсталляции и распространяются в виде обычного архива tar.gz, который просто достаточно куда-то распаковать. В Windows также есть такие программы, их еще часто называют словом Portable. Устанавливать такие программы не требуется, достаточно распаковать в любое место, но стандартное место обычно - это каталог /opt. Конечно, пункты на запуск в меню вам придется добавлять вручную, для этого нужно щелкнуть правой кнопкой по заголовку меню Программы и выбрать Правка меню.

### **3.4 Установка из исходников**

Если для вашей системы нигде нет deb-пакетов, то программу можно собрать (скомпилировать) самому из исходных кодов, которые можно скачать на официальном сайте любой Open Source программы либо из source-репозитория дистрибутива.

Рекомендуется по возможности избегать этого способа установки программ. Основное, что вам понадобится - это средства для компиляции, для этого сначала нужно установить пакет build-essential. Дальше, нужно распаковать архив с кодами программы в какую-то временную папку. Потом нужно найти файл README или INSTALL, прочитать его и выполнить то, что там написано. Чаще, установка программ таким способом ограничивается последовательным выполнением следующих команд:

```
./configure
```

```
make
```

```
sudo make install
```

Но в некоторых случаях могут быть отличия. Кроме того, после выполнения скрипта ./configure вы можете получить сообщение о том, что в системе не установлено библиотек нужных для компиляции программы. В таком случае нужно будет установить их самому и повторить процесс. Обычно процесс

компиляции занимает определенное время и напрямую зависит от мощности вашего компьютера.

### 3.5 Автоматическая установка зависимостей при сборке из исходников

Такой тип установки лучше чем просто `./configure && make && make install`, и подходит для установки программ отсутствующих в репозиториях.

Ставим auto-apt:

```
sudo apt-get install auto-apt
```

Переходим в папку с распакованными исходниками и командуем:

```
sudo auto-apt update && auto-apt -y run ./configure
```

Команда auto-apt сама доставит необходимые пакеты для сборки, и позволит задавать меньше вопросов.

Создание deb-пакета для более простой работы в дальнейшем, (установка, удаление, и прочее):

```
checkinstall -D
```

## 4 Менеджер обновлений

Каждый репозиторий содержит так называемый индекс - список всех пакетов с указанием версий и прочей полезной информации. Ubuntu периодически скачивает новые версии этих индексов. Так вот, как только в репозитории появляется новая версия установленного в системе пакета, Ubuntu замечает это и сообщает об этом пользователю. Пользователю же нужно лишь согласиться на обновление, дальше система сделает всё сама.

В первую очередь новые версии пакетов исправляют найденные проблемы безопасности, поэтому настоятельно рекомендую вам всегда обновляться при первой же возможности. Ещё разок напомним только, что вся система состоит из пакетов, поэтому через механизм обновлений вы получаете исправления как системных компонентов, так и пользовательских программ.

Основным инструментом работы с обновлениями является Менеджер обновлений. Найти его можно в меню *Система→Администрирование*, рисунок 11.

В любой момент можно проверить наличие новых обновлений, нажав на соответствующую кнопку. Если обновления имеются, их конечно можно установить, для этого нужна вторая кнопка. Для выполнения обеих этих операций вам естественно понадобятся права администратора.

Система сама автоматически периодически проверяет наличие обновлений и в случае их доступности сообщает об этом пользователю. Делает это она с помощью автоматического запуска Менеджера обновлений при обнаружении новых версий установленных пакетов. Менеджер запускается в свёрнутом состоянии, однако система обращает ваше внимание на него выделяя его заголовок в списке окон на нижней панели.

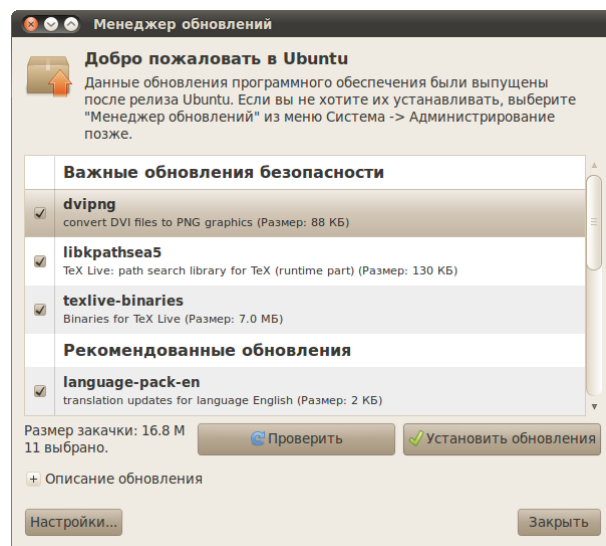


Рисунок 11- Окно менеджера обновлений

То есть фактически вам не нужно вручную запускать менеджер обновлений и что-то проверять, система это сделает за вас, вам останется только нажать на кнопку установки.

## МЕТОДИКА ВЫПОЛНЕНИЯ

1. Скачайте ISO образ Ubuntu <https://ubuntu.ru/get> или <https://ubuntu.com/download/desktop>
2. Подготовьте свободное место на жестком диске для установки ОС, например, на диске D (или другом диске достаточного объема).
3. Сделайте резервную копию важных данных!, чтобы в случае возникновения непредвиденных ситуаций избежать их потери.
4. Установочный ISO образ Linux Ubuntu необходимо записать на USB флешку, для того чтобы создать загрузочный установочный носитель, с которого и будет производиться установка. Например, в Windows – Rufus, Etcher; в Linux – Стандартное-Запись образа на USB накопитель, Etcher.
5. Загрузить ПК с носителя с установочным образом (предварительно изменить способ загрузки в UEFI).
6. В меню диска выбрать первый пункт -Ubuntu.  
В результате запустится меню программы установки, в котором надо выбрать язык, а также действие - «*Установить Ubuntu*».
- Однако если выбрать пункт «*Попробовать Ubuntu*», то система загрузится в Live режиме, где Вы можете посмотреть на систему, протестировать ее, при этом не устанавливая Ubuntu на свой компьютер.
7. Выбрать раскладку клавиатуры.
8. В разделе «Обновления и другие программы» есть два режима установки:
  - **Обычная установка** – в данном случае будет произведена установка в стандартном режиме с установкой всех входящих в дистрибутив приложений;
  - **Минимальная установка** – в этом случае установка Ubuntu будет произведена с минимальным набором приложений. Данный вариант подойдет тем,

кому нравится самостоятельно настраивать систему и устанавливать только те приложения, которые необходимы для работы.

**9. Выполнить разметку жесткого диска вручную!** Для этого выбрать пункт - Другой вариант.

После этого отобразится список существующих разделов, здесь будет и раздел для загрузки EFI (если у вас система с UEFI), и системный раздел для восстановления Windows, и другие разделы.

Необходимо найти диск D, например, или тот, на котором вы освободили место. Это можно сделать, ориентируясь на размер разделов, после того как нужный раздел вы найдете, выделите его и нажмите «Изменить» (диск C лучше не задеять в случае установленной на нем ОС Windows).

Затем надо сжать этот раздел для того, чтобы образовалось неразмеченное пространство, т.е. свободное место для Ubuntu.

В поле размер укажите тот размер, который вы хотите, чтобы у вас остался на диске D, например, общий размер D минус 50 гигабайт. Помните, что он не должен быть меньше уже занятого на диске пространства, иными словами, если на D около 100 гигабайт данных, не нужно пытаться сжать его до 50 гигабайт.

Чтобы диск D был доступен из Linux Ubuntu и все файлы на нем, то надо сразу примонтировать данный раздел. Для этого укажите:

- Использовать как – журналируемая файловая система NTFS (если диск с NTFS);
- Точка монтирования – /windows, именно так данный диск будет отображаться в файловой системе Ubuntu;

Галочку «Форматировать раздел» **не ставьте!** иначе отформатируете диск D, и все данные будут стерты. Далее «ОК», после подтверждения запустится процесс сжатия диска.

## 10. Создание разделов под ОС Ubuntu.

После сжатия в списке разделов отобразится «Свободное место», которое необходимо для создания разделов под Ubuntu.

Для создания разделов выделяем Свободное место и нажимаем на (+).

В данном случае необходимо создать точно такие же разделы, как если бы устанавливали Ubuntu на чистый диск, к таким относятся: **корневой раздел для системы** и **домашний раздел** для пользовательских данных. Раздел для подкачки SWAP создавать по желанию, так как теперь используется файл подкачки.

### **Корневой раздел:**

- Размер – для корневого раздела нужно указывать как минимум 15Гб, но лучше указывать больше;
- 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 «/». «ОК»

### **Домашний раздел.**



Выделяем Свободное место и нажимаем на (+).

- Размер – все оставшееся место;
- Тип нового раздела – указываем «Первичный»;
- Местоположение нового раздела – указываем «Начало этого пространства»;
- Использовать как – выбираем «Журналируемая файловая система Ext4»;
- Точка монтирования – для домашнего раздела указываем «/home». «ОК»

11. Указать устройство для загрузчика ОС Ubuntu.

Для этого выбираем загрузочный раздел с Windows в соответствующем пункте, размер такого раздела в районе +/- 100 мегабайт. Если у вас UEFI, то это будет раздел с файловой системой FAT32, а если BIOS, то с NTFS. Его название «Windows Boot Manager» отобразится в строке «Устройство для установки системного загрузчика». Далее «Установить сейчас», «Продолжить».

12. Выберите часовой пояс.

13. Создайте учетную запись.

14. Выполнить первоначальную настройку системы:

- установить русскую локализацию(при необходимости);
- раскладка клавиатуры ru/eng, настроить сочетание клавиш для переключения;
- сменить фон/тему рабочего стола;
- настроить захват мыши и клавиатуры (в случае с виртуальной машиной);
- освоить основные навыки работы (и показать их преподавателю) с файловым менеджером (копировать/перемещать/удалять файлы, изменять эмблемы, изменять способ отображения содержимого, разделять окно файлового менеджера на две части, автозапуск сменных носителей).

15. Установите утилиту GParted и сделайте скриншот ее работы: в итоге у вас должен быть загрузчик, корневой раздел и домашний, раздел подкачки по желанию.

16. Поключить репозитории, используя графические утилиты.

17.Поключить репозитории, используя консольные утилиты, проверить защиту (при необходимости импортировать в систему ключ репозитория)

18.Работа с программами с помощью графических утилит

- установить
- обновить
- удалить
- проверить зависимости

19.Работа с программами с помощью консольных утилит

- установить
- обновить
- удалить
- проверить зависимости

20. Установить минимум 5 программ одинаковой направленности, например, математические программы (вычислительная ОС), игры (геймерская ОС) и т.д.

21. В отчет добавить скриншот установленных программ.

### **3 КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Типы разделов винчестера, их отличие.
2. Что такое swar и для чего он нужен.
3. В какой раздел становится ОС?
4. Что такое репозитории и для чего они нужны?
5. Назовите доступные репозитории для ОС UBUNTU.
6. Какие существуют графические утилиты для управления программами?
7. Какие существуют консольные утилиты для управления программами?
8. Что такое зависимости?

## **ЛАБОРАТОРНАЯ РАБОТА №5 (2 - часа)**

### **ФАЙЛОВЫЕ СИСТЕМЫ ОС LINUX**

**Цель работы** – практическое знакомство с организацией данных основной файловой системы ОС Linux и используемыми утилитами.

### **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

При работе с Linux вы можете выбирать тип файловой системы, как и многие другие параметры. Вы будете работать с разделами Linux, на которых используется одна из расширенных файловых систем, поддерживаемых всеми дистрибутивами Linux и являющихся надежными готовыми решениями.

История расширенной файловой системы (ext) начинается с самых ранних дней Linux. В свое время эта файловая система позволила устранить ограничение на размер файла в 2 ГБ, но была чрезвычайно подвержена фрагментации. Поэтому вскоре после выпуска первой расширенной файловой системы была разработана ее вторая версия (ext2), устраняющая ряд дополнительных ограничений (например, максимальный размер файла был увеличен до 4 ТБ). Файловая система ext2 быстро стала общепринятым стандартом Linux, но продолжала развиваться вместе с развитием этой операционной системы. Таким образом, на сегодняшний день мы имеем еще две версии расширенной файловой системы – третью (ext3) и четвертую (ext4).

Однако среди прочих файловых систем Linux поддерживает и множество дисковых файловых систем, например, XFS, ReiserFS, Btrfs (B-tree File System) и JFS (IBM Journaled File System). В зависимости от задач, выполняемых на вашем компьютере и в вашей рабочей среде, какие-то из этих файловых систем могут оказаться более подходящими, чем расширенная файловая система. Тем не менее знакомство с расширенной файловой системой является хорошей отправной точкой, поскольку в большинстве дистрибутивов Linux по умолчанию используется файловая система ext3(редко) или ext4.

### **1 Особенности файловой системы ext4**

Последней версией расширенной файловой системы на сегодняшний день является файловая система ext4, обратно совместимая с ext2 и ext3. По сравнению с ext3 в ext4 реализован ряд улучшений, в основном касающихся скорости и надежности. Файловая система ext4 имеется в Linux с версией ядра 2.6.28 и выше.

В настоящее время один каталог Ext3 не может содержать более, чем 32000 подкаталогов. Ext4 снимает это ограничение и позволяет создавать неограниченное количество подкаталогов.

Файловая система Ext3, использует схему непрямого отображения блоков для отслеживания каждого блока, отвечающего за хранение данных файла. Такой подход неэффективен для больших файлов, особенно при операциях удаления и

усечения таких файлов, поскольку карта соответствия содержит по одной записи на каждый отдельный блок. В больших файлах блоков много, их карты соответствия большие, и обрабатываются они медленно.

В файловой системе применяется иной подход, основанный на так называемых экстентах. **Экстент** — это набор последовательных физических блоков, иными словами: «Эти данные находятся в следующих *n* блоках». Например, файл размером в 100 мегабайт может храниться в единственном экстенте такого же размера, вместо того, чтобы быть разбитым на 25600 4-килобайтных блоков, адресуемых путём непрямого отображения. Огромные файлы можно разделить на несколько экстентов.

Благодаря применению экстентов улучшается производительность, а также уменьшается фрагментированность, поскольку экстенты способствуют непрерывному размещению данных.

Ext4 использует механизм многоблочного распределения (multiblock allocator, mballoс) который позволяет распределить любое количество блоков с помощью единственного вызова и избежать огромных накладных расходов. Благодаря этому производительность существенно вырастает, что особенно заметно при отложенном распределении с использованием экстентов. Эта возможность никак не влияет на формат данных.

## 1.2 Организация системы файлов

Данные, хранящиеся на любом носителе, образуют файл Linux. Более того, многие устройства, подключенные к компьютеру (начиная с клавиатуры и заканчивая любыми внешними устройствами), Linux представляет как файлы (так называемые специальные файлы). **В Linux все является файлами или процессами, то есть отсутствует концепция системного реестра.** Вместо этого все объекты хранятся в виде одного из четырех типов файлов - обычными файлами, предназначенными для хранения данных, каталогами, файлами-ссылками и сокетами – это категория пользовательских данных.

Вторая категория данных, хранящихся в файловой системе – это метаданные, являющиеся индексными дескрипторами (index node) и обычно называемые inode. Индексные дескрипторы являются способом индексации атрибутов файлов в Linux. Каждый файл имеет свой inode, который обычно содержит следующую информацию:

- размер файла;
- владельцы файла (пользователь и группа);
- файловые разрешения;
- количество жестких и мягких ссылок;
- время последнего доступа и изменения файла;
- информацию о списке контроля доступа (ACL);
- любые дополнительные атрибуты, определенные для файла (например, признак неизменяемости).

Файловая система имеет иерархическую структуру. Linux может работать с различными типами файловых систем.

Каждый каталог - это отдельный файл особого типа ("d", от англ. "directory"), отличающийся от обычного файла с данными: в нем могут содержаться только ссылки на другие файлы и каталоги. Допустимые имена файлов и каталогов Linux всегда различает заглавные и строчные буквы в именах файлов и каталогов, поэтому **"student", "Student" и "STUDENT" будут тремя разными именами!**

Есть несколько символов, допустимых в именах файлов и каталогов, которые нужно использовать с осторожностью. Это спецсимволы "\*", "\", "&", "<", ">", ";", "(", ")", "|", а также символы пробела и табуляции. Кодировки и расширения В Linux в именах файлов и каталогов допустимо использовать не только символы латинского алфавита, но и любые символы любого языка. В файловой системе Linux нет никаких предписаний по поводу расширения: в имени файла может быть любое количество точек (в том числе ни одной), а после последней точки может стоять любое количество символов. Хотя расширения не обязательны, они широко используются: расширение позволяет программе, не открывая файл, только по его имени определить, какого типа данные в нем содержатся. Определить тип содержимого файла можно и на основании самих данных (сигнатур). Многие форматы предусматривают указание в начале файла, как следует интерпретировать дальнейшую информацию.

### 1.3 Кодировки и расширения

В Linux в именах файлов и каталогов допустимо использовать не только символы латинского алфавита, но и любые символы любого языка. В файловой системе Linux нет никаких предписаний по поводу расширения: в имени файла может быть любое количество точек (в том числе ни одной), а после последней точки может стоять любое количество символов. Хотя расширения не обязательны, они широко используются: расширение позволяет программе, не открывая файл, только по его имени определить, какого типа данные в нем содержатся. Определить тип содержимого файла можно и на основании самих данных (сигнатур). Многие форматы предусматривают указание в начале файла, как следует интерпретировать дальнейшую информацию.

В Linux есть утилита **file**, которая предназначена для определения типа содержащихся в файле данных. Эта утилита никогда не доверяет расширению файла (если оно присутствует), а анализирует сами данные. **file** различает не только разные данные, но и разные типы файлов.

### 1.4 Дерево каталогов

В большинстве современных файловых систем используется иерархическая модель организации данных: существует один каталог, объединяющий все данные в файловой системе - это "корень" всей файловой системы, корневой каталог. Корневой каталог может содержать любые объекты файловой системы, и в частности, подкаталоги. Подкаталоги также могут содержать любые объекты файловой системы и подкаталоги и т. д. Таким образом, все, что записано на диске - файлы, каталоги и специальные файлы -

обязательно "принадлежит" корневому каталогу: либо непосредственно (содержится в нем), либо на некотором уровне вложенности.

Структуру файловой системы можно представить наглядно в виде дерева, "корнем" которого является корневой каталог, а в вершинах расположены все остальные каталоги. На рис. 1 изображено дерево каталогов, курсивом обозначены имена файлов, прямым начертанием - имена каталогов.

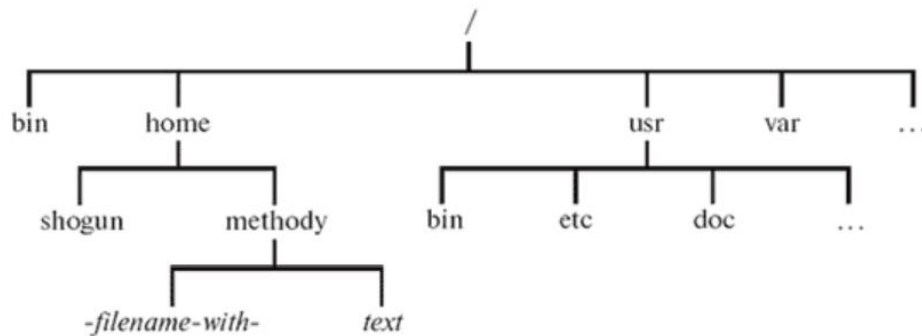


Рисунок 1 - Дерево каталогов

В любой файловой системе Linux всегда есть только один корневой каталог, который называется "/". Пользователь Linux всегда работает с единым деревом каталогов, даже если разные данные расположены на разных носителях: жестких или сетевых дисках, съемных дисках, CD-ROM и т. п. Такое представление отличается от технологии, применяемой в Windows, где для каждого устройства, на котором есть файловая система, используется свой корневой каталог, обозначенный литерой, например "a", "c", "d" и т. д.

Для того чтобы отключать и подключать файловые системы на разных устройствах в состав одного общего дерева, используются процедуры монтирования и размонтирования.

После того, как файловые системы на разных носителях подключены к общему дереву, содержащиеся на них данные доступны так, как если бы все они составляли единую файловую систему: пользователь может даже не знать, на каком устройстве какие файлы хранятся.

Положение любого каталога в дереве каталогов описывается при помощи полного пути. Полный путь всегда начинается от корневого каталога и состоит из перечисления всех вершин, встретившихся при движении по ветвям дерева до искомого каталога включительно. Названия соседних вершин разделяются символом "/" ("слэш"). В Linux полный путь, например, до каталога "methody" в файловой системе, приведенной на рис. 1, записывается следующим образом /home/methody.

### 1.5 Размещение компонентов системы: стандарт FHS

Фрагмент дерева каталогов типичной файловой системы Linux приведен на рис. 1. Утилита **ls** выведет список всего, что в этом каталоге содержится.

**Пример 1. Стандартные каталоги в /. Использование утилиты ls**

```
[student@localhost ~]$ ls /  
bin dev home lost+found misc net proc tmp var  
boot etc lib sbin usr
```

В примере 1 утилита `ls` вывела список подкаталогов корневого каталога. Этот список будет примерно таким же в любом дистрибутиве Linux. В корневом каталоге Linux-системы обычно находятся только подкаталоги со стандартными именами. Более того, не только имена, но и тип данных, которые могут попасть в тот или иной каталог, также регламентированы стандартом `Filesystem Hierarchy Standard` ("стандартная структура файловых систем"). Краткое описание стандартной иерархии каталогов Linux можно получить, выполнив команду **man hier**.

Содержимое подкаталогов корневого каталога.

**/bin** Название этого каталога происходит от слова "binaries" ("двоичные", "исполняемые"). В этом каталоге находятся исполняемые файлы самых необходимых утилит, которые могут понадобиться системному администратору или другим пользователям.

**/boot** "Boot" - загрузка системы. В этом каталоге находятся файлы, необходимые для загрузки ядра - и, обычно, само ядро. Пользователю практически никогда не требуется непосредственно работать с этими файлами.

**/dev** В этом каталоге находятся все имеющиеся в системе файлы особого типа, предназначенные для обращения к различным системным ресурсам и устройствам. Например, файлы `/dev/ttyN` соответствуют виртуальным консолям, где N - номер виртуальной консоли. Данные, введенные пользователем на первой виртуальной консоли, система считывает из файла `/dev/tty1`; в этот же файл записываются данные, которые нужно вывести пользователю на эту консоль. В специальных файлах в действительности не хранятся никакие данные, при их помощи данные передаются.

**/etc** Каталог для системных конфигурационных файлов. Здесь хранится информация о специфических настройках данной системы: информация о зарегистрированных пользователях, доступных ресурсах, настройках различных программ.

**/home** Здесь расположены каталоги, принадлежащие пользователям системы - домашние каталоги, отсюда и название "home". Отделение всех файлов, создаваемых пользователями, от прочих системных файлов дает очевидное преимущество: серьезное повреждение системы или необходимость обновления не затронет пользовательских файлов.

**/lib** Название этого каталога - сокращение от "libraries" (англ. "библиотеки"). Чтобы не включать эти функции в текст каждой программы, используются стандартные функции библиотек - это значительно экономит место на диске и упрощает написание программ. В этом каталоге содержатся библиотеки, необходимые для работы наиболее важных системных утилит, размещенных в `/bin` и `/sbin`.

**/mnt** Каталог для монтирования (от англ. "mount") - временного подключения файловых систем, например, на съемных носителях (CD-ROM и

др.).

**/proc** В этом каталоге все файлы "виртуальные" - они располагаются не на диске, а в оперативной памяти. В этих файлах содержится информация о программах (процессах), выполняемых в данный момент в системе.

**/root** Домашний каталог администратора системы - пользователя root. Смысл размещать его отдельно от домашних каталогов остальных пользователей состоит в том, что

**/home** может располагаться на отдельном устройстве, которое не всегда доступно (например, на сетевом диске), а домашний каталог root должен присутствовать в любой ситуации.

**/sbin** Каталог для важнейших системных утилит (название каталога - сокращение от "system binaries"): в дополнение к утилитам /bin здесь находятся программы, необходимые для загрузки, резервного копирования, восстановления системы. Полномочия на исполнение этих программ есть только у системного администратора.

**/tmp** Этот каталог предназначен для временных файлов: в таких файлах программы хранят необходимые для работы промежуточные данные. После завершения работы программы временные файлы теряют смысл и должны быть удалены. Обычно каталог /tmp очищается при каждой загрузке системы.

**/usr** Здесь можно найти такие же подкаталоги bin, etc, lib, sbin, как и в корневом каталоге. Однако в корневой каталог попадают только утилиты, необходимые для загрузки и восстановления системы в аварийной ситуации - все остальные программы и данные располагаются в подкаталогах /usr. Этот раздел файловой системы может быть очень большим.

**/var** Название этого каталога - сокращение от "variable" ("переменные" данные). Здесь размещаются те данные, которые создаются в процессе работы разными программами и предназначены для передачи другим программам и системам (очереди печати, электронной почты и др.) или для сведения системного администратора (системные журналы, содержащие протоколы работы системы). В отличие от каталога /tmp сюда попадают те данные, которые могут понадобиться после того, как создавшая их программа завершила работу.

Стандарт FHS регламентирует не только перечисленные каталоги, но и их подкаталоги, а иногда даже приводит список конкретных файлов, которые должны присутствовать в определенных каталогах.

Этот стандарт последовательно соблюдается во всех Linux-системах. Командная оболочка "знает", что исполняемые файлы располагаются в каталогах /bin, /usr/bin и т. д. - именно в этих каталогах она ищет исполняемый файл cat. Благодаря этому каждая вновь установленная в системе программа немедленно оказывается доступна пользователю из командной строки. Для этого не требуется ни перезагружать систему, ни запускать какие-либо процедуры - достаточно просто поместить исполняемый файл в один из соответствующих каталогов. Рекомендации стандарта по размещению файлов и каталогов основываются на принципе размещения файлов, которые по-разному используются в системе, в разных подкаталогах.

По типу использования файлы можно разделить на следующие группы:



## **пользовательские/системные файлы.**

**Пользовательские файлы** - это все файлы, созданные пользователем и не принадлежащие ни одному из компонентов системы.

### **Изменяющиеся/неизменные файлы**

К неизменным файлам относятся все статические компоненты программного обеспечения: библиотеки, исполняемые файлы и т. д. - все, что не изменяется само без вмешательства системного администратора.

Изменяющиеся файлы изменяются без вмешательства человека в процессе работы системы: системные журналы, очереди печати и пр. Выделение неизменных файлов в отдельную структуру (например, /usr) позволяет использовать соответствующую часть файловой системы в режиме "только чтение", что уменьшает вероятность случайного повреждения данных и позволяет применять для хранения этой части файловой системы CD-ROM и другие носители, доступные только для чтения. разделяемые/неразделяемые файлы Это разграничение становится полезным, если речь идет о сети, в которой работает несколько компьютеров.

Значительная часть информации при этом может храниться на одном из компьютеров и использоваться всеми остальными по сети (к такой информации относятся, например, многие программы и домашние каталоги пользователей). Однако часть файлов нельзя разделять между системами (например, файлы для начальной загрузки системы). Полный путь к каталогу формально ничем не отличается от пути к файлу, т. е. по полному пути нельзя сказать наверняка, является его последний элемент файлом или каталогом. Чтобы отличать путь к каталогу, иногда используют запись с символом "/" в конце пути, например "/home/student/".

## **1.6 Текущий и домашний каталог**

Каждая выполняемая программа "работает" в строго определенном каталоге файловой системы. Такой каталог называется текущим каталогом. Можно представлять, что программа во время работы "находится" именно в этом каталоге, это ее "рабочее место". В зависимости от текущего каталога поведение программы может меняться: зачастую программа будет по умолчанию работать с файлами, расположенными именно в текущем каталоге - до них она "дотянется" в первую очередь. Текущий каталог есть у любой программы, в том числе и у командной оболочки (shell) пользователя.

Поскольку пользователь взаимодействует с системой через командную оболочку, можно говорить о том, что пользователь "находится" в том каталоге, который в данный момент является текущим каталогом его командной оболочки. Все команды, выполняемые пользователем при помощи shell, наследуют текущий каталог shell, т. е. "работают" в том же каталоге. По этой причине пользователю важно знать текущий каталог shell.

Для этого служит утилита pwd: Команда pwd (print working directory) возвращает полный путь текущего каталога командной оболочки - естественно, именно той командной оболочки, при помощи которой была выполнена команда

pwd. Почти все утилиты по умолчанию читают и создают файлы в текущем каталоге.

Например, утилита cat (concatenation – конкатенация) - выводит на экран содержимое файла "text":

```
[student@localhost student]$ cat text
```

В действительности, командная оболочка, прежде чем передавать параметр "text" (имя файла) утилите cat, подставляет значение текущего каталога - получается полный путь к этому файлу в файловой системе:

```
"/home/student/text".
```

Содержимое данного файла утилита cat выведет на экран.

**Относительный путь**(relative path) - путь к объекту файловой системы, не начинающийся в корневом каталоге. Для каждого процесса Linux определен текущий каталог, с которого система начинает относительный путь при выполнении файловых операций.

Относительный путь строится точно так же, как и полный - перечислением через "/" всех названий каталогов, встретившихся при движении к искомому каталогу или файлу. Между полным и относительным путем есть только одно существенное различие: относительный путь начинается от текущего каталога, в то время как полный путь всегда начинается от корневого каталога. Относительный путь любого файла или каталога в файловой системе может иметь любую конфигурацию - чтобы добраться до искомого файла, можно двигаться как по направлению к корневому каталогу, так и от него. Linux различает **полный** и **относительный** пути очень просто: если имя объекта начинается на "/" - это полный путь, в любом другом случае - относительный.

Отделить путь к файлу от его имени можно с помощью команд **dirname** и **basename** соответственно.

В Linux у каждого пользователя обязательно есть собственный каталог, который и становится текущим сразу после регистрации в системе - домашний каталог. Домашний каталог (home directory) - это каталог, предназначенный для хранения собственных данных пользователя Linux. Как правило, является текущим непосредственно после регистрации пользователя в системе. Полный путь к домашнему каталогу хранится в переменной окружения

HOME.Имя домашнего каталога ~

Поскольку каждый пользователь располагает собственным каталогом и по умолчанию работает в нем, решается задача разделения файлов разных пользователей. Обычно доступ других пользователей к чужому домашнему каталогу ограничен: наиболее типична ситуация, когда пользователи могут читать содержимое файлов друг друга, но не имеют права их изменять или удалять.

## 2 Работа с файловой системой

### 2.1 Информация о содержимом каталога – утилита ls

Чтобы иметь возможность ориентироваться в файловой системе, нужно знать, что содержится в каждом каталоге. Просмотреть содержимое любого

каталога можно при помощи утилиты `ls` (сокращение от англ. "list" - "список"): Команда `ls` без параметров выводит список файлов и каталогов, содержащихся в текущем каталоге. Утилита `ls` принимает один параметр - имя каталога, содержимое которого нужно вывести. Имя может быть задано любым доступным способом: в виде полного или относительного пути.

Кроме параметра, утилита `ls` может использовать множество ключей, которые нужны для того, чтобы выводить дополнительную информацию о файлах в каталоге или выводить список файлов выборочно. Чтобы узнать обо всех возможностях `ls`, нужно прочесть руководство по этой утилите с помощью команды **`man ls`**.

Ключ **`-F`** используется, чтобы отличать файлы от каталогов. При наличии этого ключа `ls` в конце имени каждого каталога ставит символ `/`, чтобы показать, что в нем может содержаться что-то еще.

Утилита `ls` по умолчанию не выводит информацию об объектах, чье имя начинается с `.` - в том числе о `.` и `..`. Для того чтобы посмотреть полный список содержимого каталога, и используется ключ `-a` (`all`). Как правило, с `.` начинаются имена конфигурационных файлов и конфигурационных каталогов (вроде `.bashrc`), работа с которыми (т. е. настройка окружения, "рабочего места") не пересекается с работой над какой-нибудь прикладной задачей. Родительский каталог (`parent directory`) - это каталог, в котором содержится данный. Для корневого каталога родительским является он сам. Ссылки на текущий и на родительский каталог обязательно присутствуют в каждом каталоге в `Linux`.

Даже если каталог пуст, т. е. не содержит ни одного файла или подкаталога, команда `ls -a` выведет список из двух имен: `.` и `..`. За ссылками на текущий и родительский каталоги могут следовать несколько файлов и каталогов, имена которых начинаются с `.`.

В них содержатся настройки командной оболочки (файлы, начинающиеся с `.bash`) и других программ. В домашнем каталоге каждого пользователя `Linux` всегда присутствует несколько таких файлов. Использование этих файлов позволяет пользователям независимо друг от друга настраивать поведение командной оболочки и других программ - организовывать свое "рабочее место" в системе.

## 2.2 Перемещение по дереву каталогов – команда `cd`

Пользователь может работать с файлами не только в своем домашнем каталоге, но и в других каталогах. В этом случае будет удобно сменить текущий каталог. Для смены текущего каталога командной оболочки используется команда `cd` (от англ. "change directory" - "сменить каталог"). Команда `cd` принимает один параметр: имя каталога, в который нужно переместиться - сделать текущим. В качестве имени каталога можно использовать полный или относительный путь. В приглашении командной строки часто указывается текущий каталог `shell` - чтобы пользователю легче было ориентироваться, в каком каталоге он "находится" в данный момент. Командная оболочка умеет достраивать имена файлов и каталогов: пользователю достаточно набрать

несколько первых символов имени файла или каталога и нажать Tab.

Если есть только один вариант завершения имени - оболочка закончит его сама, и пользователю не придется набирать оставшиеся символы. Дистраивание - весьма существенное средство экономии усилий и повышения эффективности при работе с командной строкой. Современные командные оболочки умеют дистраивать имена файлов и каталогов, а также имена команд. Дистраивание наиболее развито в командном интерпретаторе zsh.

Оболочка PowerShell также умеет дистраивать имена. Для перемещения в родительский каталог ("/home") удобно воспользоваться ссылкой "..". Необходимость вернуться в домашний каталог из произвольной точки файловой системы возникает довольно часто, поэтому командная оболочка поддерживает обозначение домашнего каталога при помощи символа "~". Поэтому чтобы перейти в домашний каталог из любого другого, достаточно выполнить команду "cd ~".

При исполнении команды символ "~" будет заменен командной оболочкой на полный путь к домашнему каталогу пользователя. При помощи символа "~" можно ссылаться и на домашние каталоги других пользователей: "~имя пользователя". Команда cd, поданная без параметров, эквивалентна команде "cd ~" и делает текущим каталогом домашний каталог пользователя.

## 2.3 Создание каталогов – утилита **mkdir**

В домашнем каталоге, как и в любом другом, можно создавать сколько угодно подкаталогов, в них - свои подкаталоги и т. д. Иными словами, пользователю принадлежит фрагмент (поддерево) файловой системы, корнем которого является его домашний каталог. Чтобы организовать такое поддерево, потребуется создать каталоги внутри домашнего. Для этого используется утилита **mkdir**. Она применяется с одним обязательным параметром: именем создаваемого каталога. По умолчанию каталог будет создан в текущем каталоге.

## 2.4 Копирование и перемещение файлов

Для перемещения файлов и каталогов предназначена утилита **mv** (от англ. "move" - "перемещать").

У mv два обязательных параметра: первый - перемещаемый файл или каталог, второй - файл или каталог назначения. Имена файлов и каталогов могут быть заданы в любом допустимом виде: при помощи полного или относительного пути. Кроме того, mv позволяет перемещать не только один файл или каталог, а сразу несколько.

За подробностями о допустимых параметрах и ключах следует обратиться к руководству по mv.

Перемещение файла внутри одной файловой системы в действительности равнозначно его переименованию: данные самого файла при этом остаются на тех же секторах диска, а изменяются каталоги, в которых произошло перемещение. Перемещение предполагает удаление ссылки на файл из того

каталога, откуда он перемещен, и добавление ссылки на этот самый файл в тот каталог, куда он перемещен. В результате изменяется полное имя файла - полный путь, т. е. положение файла в файловой системе. Иногда требуется создать копию файла: для большей сохранности данных, для того, чтобы создать модифицированную версию файла и т. п.

В Linux для этого предназначена утилита **ср** (от англ. "copy" - "копировать"). Утилита **ср** требует использования двух обязательных параметров: первый - копируемый файл или каталог, второй - файл или каталог назначения. Как обычно, в именах файлов и каталогов можно использовать полные и относительные пути.

Существует несколько вариантов комбинации файлов и каталогов в параметрах **ср** - о них можно прочесть в руководстве. Нужно иметь в виду, что в Linux утилита **ср** нередко настроена таким образом, что при попытке скопировать файл поверх уже существующего файла никакого предупреждения не выводится. В этом случае файл будет просто перезаписан, а данные, которые содержались в старой версии файла, безвозвратно потеряны. Поэтому при использовании **ср** следует всегда быть внимательным и проверять имена файлов, которые нужно скопировать.

Созданная при помощи **ср** копия файла связана с оригиналом только в воспоминаниях пользователя, в файловой же системе исходный файл и его копия - две совершенно независимые и ничем не связанные единицы. Поэтому при наличии нескольких копий одного и того же файла в рамках одной файловой системы повышается вероятность запутаться в копиях или забыть о некоторых из них. Если задача состоит в том, чтобы обеспечить доступ к одному и тому же файлу из разных точек файловой системы, нужно использовать специально предназначенный для этого механизм файловой системы Linux - ссылки.

## 2.5 Файл и его имена: ссылки

### Жесткие ссылки – утилита **ln**

Каждый файл представляет собой область данных на жестком диске компьютера или на другом носителе информации, которую можно найти по имени. В файловой системе Linux содержимое файла связывается с его именем при помощи жестких ссылок. Создание файла с помощью любой программы означает, что будет создана жесткая ссылка - имя файла, и открыта новая область данных на диске. Причем количество ссылок на одну и ту же область данных (файл) не ограничено, то есть у файла может быть несколько имен. Пользователь Linux может добавить файлу еще одно имя (создать еще одну жесткую ссылку на файл) при помощи утилиты **ln** (от англ. "link" - "соединять, связывать").

Первый параметр - это имя файла, на который нужно создать ссылку, второй - имя новой ссылки. По умолчанию ссылка будет создана в текущем каталоге:

Пример 2. Создание жестких ссылок

```
[student@localhost ~]$ ln text text-hardlink
```

В примере 2 в домашнем каталоге пользователя `student` создана жесткая ссылка с именем `"text-hardlink"` на файл `"text"`.

Если вывести подробный список файлов текущего каталога и его подкаталогов (`"ls -lR"`), то у файлов `"text"` и `"text-hardlink"` совпадут и размер, и время создания. Теперь `"text-hardlink"` и `"text"` - это два имени одного и того же файла.

Доступ к одному и тому же файлу при помощи нескольких имен может понадобиться в следующих случаях: Одна и та же программа известна под несколькими именами. Доступ пользователей к некоторым каталогам в системе может быть ограничен из соображений безопасности. Однако если все же нужно организовать доступ пользователей к файлу, который находится в таком каталоге, можно создать жесткую ссылку на этот файл в другом каталоге. Современные файловые системы даже на домашних персональных компьютерах могут насчитывать до нескольких десятков тысяч файлов и тысячи каталогов. Обычно у таких файловых систем сложная многоуровневая иерархическая организация - в результате пути ко многим файлам становятся очень длинными. Чтобы организовать более удобный доступ к файлу, который находится очень "глубоко" в иерархии каталогов, также можно использовать жесткую ссылку в более доступном каталоге.

Полное имя некоторых программ может быть весьма длинным (например, `i586-alt-linux-gcc-3.3`), к таким программам удобнее обращаться при помощи сокращенного имени (жесткой ссылки) - `gcc-3.3`.

### **Индексные дескрипторы**

Поскольку благодаря жестким ссылкам у файла может быть несколько имен, понятно, что вся существенная информация о файле в файловой системе привязана не к имени.

В файловых системах Linux вся информация, необходимая для работы с файлом, хранится в индексном дескрипторе. Для каждого файла существует индексный дескриптор: не только для обычных файлов, но и для каталогов и т. д.

Каждому файлу соответствует один индексный дескриптор. **Индексный дескриптор** - это описание файла, в котором содержится: тип файла (обычный файл, каталог, специальный файл и т. д.); права доступа к файлу; информация о том, кому принадлежит файл; отметки о времени создания, модификации, последнего доступа к файлу; размер файла; указатели на физические блоки на диске, принадлежащие этому файлу - в этих блоках хранится "содержимое" файла.

Все индексные дескрипторы пронумерованы, поэтому номер индексного дескриптора - это уникальный идентификатор файла в файловой системе - в отличие от имени файла (жесткой ссылки на него), которых может быть несколько. Узнать номер индексного дескриптора любого файла можно при помощи утилиты **`ls` с ключом `-i`**

Если вывести номера индексных дескрипторов файла `"text"` и жесткой ссылки на него `"text-hardlink"` - можно увидеть, что эти номера совпадают, то есть этим двум именам соответствует один индексный дескриптор, т. е. один и

тот же файл. Все операции с файловой системой - создание, удаление и перемещение файлов - производятся на самом деле над индексными дескрипторами, а имена нужны только для того, чтобы пользователь мог легко ориентироваться в файловой системе. Более того, имя (или имена) файла в его индексном дескрипторе не указаны.

**Жесткую ссылку** (имя файла, хранящееся в каталоге) можно представлять как каталожную карточку, на которой указан номер индексного дескриптора - идентификатор файла. **Жесткая ссылка** (hard link) - запись вида имя файла+номер индексного дескриптора в каталоге. Жесткие ссылки в Linux - основной способ обратиться к файлу по имени.

У жестких ссылок есть два существенных ограничения:

-Жесткая ссылка может указывать только на файл, но не на каталог, потому что в противном случае в файловой системе могут возникнуть циклы - бесконечные пути.

-Жесткая ссылка не может указывать на файл в другой файловой системе. Например, невозможно создать на жестком диске жесткую ссылку на файл, расположенный на дискете. Чтобы избежать этих ограничений, были разработаны символичные ссылки.

### **Символичные ссылки.**

**Символическая ссылка** - это просто файл, в котором содержится имя другого файла. Символичные ссылки, как и жесткие, предоставляют возможность обращаться к одному и тому же файлу по разным именам. Кроме того, символичные ссылки могут указывать и на каталог, чего не позволяют жесткие ссылки. Символичные ссылки называются так потому, что содержат символы - путь к файлу или каталогу.

Символическая ссылка (symbolic link, файл-ссылка) - это файл особого типа ("l"), в котором содержится путь к другому файлу. Если на пути к файлу встречается символическая ссылка, система выполняет подстановку: исходный путь заменяется тем, что содержится в ссылке.

Символическую ссылку можно создать при помощи команды **ln** с ключом **"-s"** (сокращение от "symbolic"). Если выполнить команду `cat имя_файла-ссылки`, то на экран будет выведено содержимое файла, на который указывает ссылка. Символическая ссылка вполне может содержать имя несуществующего файла. В этом случае ссылка будет существовать, но не будет "работать": например, если попробовать вывести содержимое такой "битой" ссылки при помощи команды `cat`, будет выдано сообщение об ошибке. Узнать, куда указывает символическая ссылка, можно при помощи утилиты `realpath`.

## **2.6 Удаление файлов и каталогов – утилиты rm и rmdir**

В ОС Linux для удаления файлов предназначена утилита `rm` (сокращение от англ. "remove" - "удалять"): Если удалить файл `text` в домашнем каталоге пользователя `student`, файл `text-hardlink`, который является жесткой ссылкой на удаленный файл `text`, сохранится, количество жестких ссылок на этот файл

уменьшится с "2" до "1" - действительно, text-hardlink - теперь единственное имя этого файла. Однако если удалить и жесткую ссылку text-hardlink, у этого файла больше не останется ни одного имени, он станет недоступным пользователю и будет уничтожен. Утилита `rm` предназначена именно для удаления жестких ссылок, а не самих файлов. В Linux, чтобы полностью удалить файл, требуется последовательно удалить все жесткие ссылки на него. При этом все жесткие ссылки на файл (его имена) равноправны - среди них нет "главной", с исчезновением которой исчезнет файл. Пока есть хоть одна ссылка, файл продолжает существовать. Впрочем, у большинства файлов в Linux есть только одно имя (одна жесткая ссылка на файл), поэтому команда `rm` имя файла в большинстве случаев успешно удаляет файл. Как уже говорилось, символичные ссылки - это отдельные файлы, поэтому после удаления файла `text`, `text-symlink`, который ссылался на этот файл, продолжает существовать, однако теперь это - "битая ссылка", поэтому его также можно удалить командой `rm`. Для удаления каталогов предназначена другая утилита - `rmdir` (от англ. "remove directory"). Впрочем, `rmdir` согласится удалить каталог только в том случае, если он пуст - в нем нет никаких файлов и подкаталогов. Удалить каталог вместе со всем его содержимым можно командой `rm` с ключом "-r" (recursive). Команда `rm -r` каталог - очень удобный способ потерять в одночасье все файлы: она рекурсивно обходит весь каталог, удаляя все, что попадется: файлы, подкаталоги, символичные ссылки... а ключ "-f" (force) делает ее работу еще неотвратимее, так как подавляет запросы вида "удалить защищенный от записи файл", так что `rm` работает безмолвно и безостановочно.

**ПОМНИТЕ: если вы удалили файл, значит, он уже не нужен, и не подлежит восстановлению!** В Linux не предусмотрено процедуры восстановления удаленных файлов и каталогов. Поэтому стоит быть очень внимательным, отдавая команду `rm` и, тем более, `rm -r`: нет никакой гарантии, что случайно удаленные данные удастся восстановить.

## 2.7 Права доступа в файловой системе

Говоря о правах доступа пользователя к файлам, заметим, что в действительности манипулирует файлами не сам пользователь, а запущенный им процесс (например, утилита `rm` или `cat`). Поскольку и файл, и процесс создаются и управляются системой, ей нетрудно организовать какую угодно политику доступа одних к другим, основываясь на любых свойствах процессов как субъектов и файлов как объектов системы.

В Linux, однако, используются не какие угодно свойства, а результат идентификации пользователя – его UID. Каждый процесс системы обязательно принадлежит какому-нибудь пользователю, и идентификатор пользователя (UID) – обязательное свойство любого процесса Linux.

Когда программа `login` запускает стартовый командный интерпретатор, она приписывает ему UID, полученный в результате диалога.

Обычный запуск программы (`exec()`) или порождение нового процесса (`fork()`) не изменяют UID процесса, поэтому все процессы, запущенные



пользователем во время терминальной сессии, будут иметь его идентификатор. Поскольку UID однозначно определяется входным именем, оно нередко используется вместо идентификатора – для наглядности. Например, вместо выражения "идентификатор пользователя, соответствующий входному имени student", говорят "UID student" (в приведенном ниже примере этот идентификатор равен 500):

Пример 3. Как узнать идентификаторы пользователя и членство в группах [student@localhost student]\$ id uid=500 (student) gid=500(student) группы=500 (student) Утилита id выводит входное имя пользователя и соответствующий ему UID, а также группу по умолчанию и полный список групп, членом которых он является.

Пользователь может быть членом нескольких групп, равно как и несколько пользователей могут быть членами одной и той же группы. Исторически сложилось так, что одна из групп – группа по умолчанию – является для пользователя основной - когда говорят о "GID пользователя", имеют в виду именно идентификатор группы по умолчанию. GID пользователя вписан в учетную запись и хранится в /etc/passwd, а информация о соответствии имен групп их идентификаторам, равно как и о том, в какие еще группы входит пользователь – в файле /etc/group.

Из этого следует, что пользователь не может не быть членом как минимум одной группы.

При создании объектов файловой системы – файлов, каталогов и т. п. – каждому приписывается ярлык. Ярлык включает в себя UID – идентификатор пользователя-хозяина файла, GID – идентификатор группы, которой принадлежит файл, тип объекта и набор так называемых атрибутов (код доступа), а также некоторую дополнительную информацию. Атрибуты (или код доступа) определяют, кто и что имеет право делать с файлом.

Ключ **"-l"** утилиты **ls** определяет длинный (long) формат выдачи (справа налево): имя файла, время последнего изменения файла, размер в байтах, группа, хозяин, количество жестких ссылок и строка атрибутов. Первый символ в строке атрибутов определяет тип файла. Тип **"-"** отвечает "обычному" файлу, а тип **"d"** – каталогу (directory).

Несмотря на то, что создание жестких ссылок на каталог невозможно, значение поля "количество жестких ссылок" (второй столбец) для всех каталогов примера равно двум, а не одному.

На самом деле этого и следовало ожидать, потому что любой каталог файловой системы Linux всегда имеет не менее двух имен: собственное (например, tmp) и имя **"."** в самом этом каталоге (tmp/.).

Если же в каталоге создать подкаталог, количество жестких ссылок на этот каталог увеличится на 1 за счет имени **".."** в подкаталоге (например, tmp/subdir1/..)

### 2.7.1 Иерархия прав доступа

Рассмотрим более подробно, чему соответствуют девять символов в

строке атрибутов, выдаваемой ls. Эти девять символов имеют вид "rwxrwxrwx", где некоторые "r", "w" и "x" могут заменяться на "-".

Очевидно, буквы отражают принятые в Linux три вида доступа – чтение, запись и использование – однако в ярлыке они присутствуют в трех экземплярах! Дело в том, что любой пользователь (процесс) Linux по отношению к любому файлу может выступать в трех ролях: как хозяин (user), как член группы, которой принадлежит файл (group), и как посторонний (other), никаких отношений собственности на этот файл не имеющий. Строка атрибутов – это три тройки "rwx", описывающие права доступа к файлу хозяина этого файла (первая тройка, "u"), группы, которой принадлежит файл (вторая тройка, "g") и посторонних (третья тройка, "o"). Если в какой-либо тройке не хватает буквы, а вместо нее стоит "-", значит, пользователю в соответствующей роли будет в соответствующем виде доступа отказано.

При выяснении отношений между файлом и пользователем, запустившим процесс, роль определяется так: Если UID файла совпадает с UID процесса, пользователь – хозяин файла. Если GID файла совпадает с GID любой группы, в которую входит пользователь, он – член группы, которой принадлежит файл. Если ни UID, ни GID файла не пересекаются с UID процесса и списком групп, в которые входит запустивший его пользователь, этот пользователь – посторонний. Именно в роли хозяина пользователь (процесс) может изменять ярлык файла. Единственное, чего не может делать хозяин со своим файлом – менять ему хозяина.

## 2.8 Использование прав доступа в Linux

В Linux определено несколько системных групп, задача которых – обеспечивать доступ членов этих групп к разнообразным ресурсам системы. Часто такие группы носят говорящие названия: "disk", "audio", "cdwriter" и т. п. Если обычным пользователям доступ к некоторому файлу, каталогу или специальному файлу Linux закрыт, он открыт членам группы, которой этот объект принадлежит.

Например, в Linux почти всегда используется виртуальная файловая система /proc – каталог, в котором в виде подкаталогов и файлов представлена информация из таблицы процессов. Имя подкаталога /proc совпадает с PID соответствующего процесса, а содержимое этого подкаталога отражает свойства процесса. Хозяином такого подкаталога будет хозяин процесса (с правами на чтение и использование), поэтому любой пользователь сможет посмотреть информацию о своих процессах. Именно каталогом /proc пользуется утилита ps. Использование утилиты ps для просмотра выполняющихся процессов Linux рассматривается в работе “Процессы ОС Linux”.

## 2.9 Суперпользователь

Суперпользователь - единственный пользователь в Linux, на которого не распространяются ограничения прав доступа. Имеет нулевой идентификатор

пользователя. Суперпользователь в Linux – это выделенный пользователь системы, на которого не распространяются ограничения прав доступа. UID суперпользовательских процессов равен 0: так система отличает их от процессов других пользователей.

Именно суперпользователь имеет возможность произвольно изменять владельца и группу файла. Ему открыт доступ на чтение и запись к любому файлу системы и доступ на чтение, запись и использование к любому каталогу. Наконец, суперпользовательский процесс может на время сменить свой собственный UID с нулевого на любой другой.

Именно так и поступает программа login, когда, проведя процедуру идентификации пользователя, запускает стартовый командный интерпретатор. Среди учетных записей Linux всегда есть запись по имени root ("корень"), соответствующая нулевому идентификатору, поэтому вместо "суперпользователь" часто говорят "root".

Множество системных файлов принадлежат root, множество файлов только ему доступны на чтение или запись. Пароль этой учетной записи – одна из самых больших драгоценностей системы. Именно с ее помощью системные администраторы выполняют самую ответственную работу.

Существует два различных способа получить права суперпользователя. Первый – это зарегистрироваться в системе под этим именем, ввести пароль и получить стартовую оболочку, имеющую нулевой UID. Это – самый неправильный способ, пользоваться которым стоит, только если нельзя применить другие. В ОС Ubuntu описанный способ не используется, вместо него используется второй способ.

Второй способ — воспользоваться специальной утилитой sudo, которая позволяет выполнить одну или несколько команд от лица другого пользователя. По умолчанию эта утилита выполняет команду от лица пользователя root, то есть запускает командный интерпретатор с нулевым UID. Отличие от предыдущего способа в том, что всегда известно, кто именно запускал sudo, а значит, ясно, с кого спрашивать за последствия.

## 2.10 Поиск файлов

Для поиска файла по имени или его части используется утилита locate. Параметр задает имя файла. Для поиска без учета регистра служит ключ `-i`. Для ограничения объема выводимой информации используется ключ `-n` число. Построчный вывод получается, если результаты поиска направить по конвейеру в программу less, например `locate mp3 | less`

Утилита locate ведет поиск в базе данных, которая должна периодически обновляться утилитой updatedb, выполняемой с правами администратора. Другой способ найти и файл предоставляет утилита **find**. Достоинствами утилиты find являются независимость от базы данных и широкие функциональные возможности, недостаток – меньшая скорость поиска по сравнению с locate

Ключи утилиты find

Ключ	Назначение
-name	Задаёт имя файла или его часть
-size	Задаёт размер файла, например 12k
-type	Задаёт тип объекта для поиска: f-обычный файл d-каталог l-символьная ссылка
-a	Логическая связка and
-o	Логическая связка or
-user	Задаёт имя пользователя

## 2.11 Суперблок

На самом верхнем уровне вся информация о самой файловой системе хранится в т. н. суперблоке. Хотя работа с суперблоком может не представлять особого интереса, понимание концепции использования команды `dump2fs` может помочь вам получить полное представление о концепциях хранения данных в файловой системе.

В листинге приведен пример, в котором мы получаем информацию о разделе, расположенном на устройстве `/dev/sda1` (в нашем случае это раздел `/boot`). В конструкции `grep -i superbloc` мы используем команду `grep` без учета регистра для вывода информации, содержащей строку `superblock`.

**Листинг. Использование `dumpe2fs` для получения информации суперблока**

```
# dumpe2fs /dev/sda1 | grep -i superbloc
Primary superblock at 1, Group descriptors at 2-2
Backup superblock at 8193, Group descriptors at 8194-8194
Backup superblock at 24577, Group descriptors at 24578-24578
Backup superblock at 40961, Group descriptors at 40962-40962
Backup superblock at 57345, Group descriptors at 57346-57346
Backup superblock at 73729, Group descriptors at 73730-73730
```

## 3 Справка по работе с терминалом

**Терминал** - графическая программа эмулирующая консоль. Такие программы позволяют не выходя из графического режима выполнять команды.

Терминал по сравнению с консолью имеет дополнительный функционал (различные настройки, вкладки, можно запускать много окон, управление мышью в некоторых программах, контекстное меню, главное меню, полоса прокрутки).

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

**dstu@Zotac-Zbox-Nano:~\$**

dstu - имя учетной записи пользователя

@ - разделитель

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)

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 (автовыполнение команд при входе в оболочку, внутренние команды оболочки, ведение истории, можно назначать сокращенные команды - **алиасы**).

**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

Команды для управления программами строятся по такой схеме:

название\_программы -ключ значение

**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)

## Горячие клавиши. Копирование команд

Часто вам придётся следовать каким-либо инструкциям, которые требуют копирования команд в терминал. Вставить текст в терминал можно тремя способами: `Ctrl+Shift+V`, нажатием средней кнопки мыши или правой кнопки мыши и выбором строки «Вставить».

<code>↑</code> или <code>Ctrl+P</code>	прокрутка недавно использованных команд вверх
<code>↓</code> или <code>Ctrl+N</code>	прокрутка недавно использованных команд вниз
<code>Enter</code>	выполнение выбранной команды
<code>Tab</code>	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 <code>tab</code> выведется список всех возможных вариантов.
<code>Ctrl+R</code>	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.
<b>History</b>	Команда <b>history</b> выводит список всех команд, которые вы вводили. Каждой команде будет присвоен номер. Чтобы выполнить команду под номером <code>x</code> , просто введите <code>!x</code> . Если у вас получилась слишком длинная история, можно попробовать <code>history I less</code> , это сделает список прокручиваемым.

## Изменение текста

Используйте стрелки влево/вправо, чтобы перемещаться по строке. Ввод с клавиатуры будет добавлять символы. Существующий текст удаляться не будет.

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

## Просмотр справки и руководства по программе

Для получения краткого руководства по программе, нужно выполнить:

## **man** программа

Перемещаться можно клавишами со стрелками, а выйти из него, нажав клавишу `q`. Подробнее в смотрите в `man man`

Для получения справки, в которой указаны ключи, нужно выполнить:

```
программа -h  
программа --help
```

Оба варианта должны работать одинаково, но некоторые авторы программ оставляют один из этих ключей, так что пробуйте оба.

**Если вы не уверены, какая команда вам нужна, попробуйте поискать по текстам мануалов:**

```
man -k something-you-need
```

будет искать то, что вам нужно во всех мануалах. Попробуйте:

```
man -k nautilus
```

чтобы увидеть, как это работает.

Пользователи рабочего стола Gnome/Unity могут воспользоваться утилитой `help` предоставляющей простой GUI, выполнив команду в консоли

```
help man:<команда>
```

## **Выполнение команд с неограниченными привилегиями**

Перед большинством команд, перечисленных ниже, необходимо писать команду:

```
sudo
```

Это временно даёт права суперпользователя, которые необходимы для работы с файлами и каталогами, которые не принадлежат вашему аккаунту. Для использования `sudo` требуется ввести пароль. Только пользователи с такими административными привилегиями могут использовать эту команду.

## **Команды для работы с файлами и папками.**

- Символ тильда (`~`) обозначает вашу домашнюю папку. Если вы `user`, то тильда (`~`) — это `/home/user`.
- Команда `pwd` («print working directory») позволяет вам узнать, в какой директории вы находитесь в данный момент. Помните, однако, что эту же информацию Gnome Terminal всегда показывает в названии окна.

- `ls` покажет вам все файлы в текущей директории. Если использовать эту команду с определёнными опциями, можно также включить отображение размера файлов, времени последнего изменения и прав на файлы. Например:

```
ls ~
```

покажет всё, что у вас есть в домашней папке.

- Команда `cd` меняет рабочую директорию. Когда вы только запускаете терминал, вы будете находиться в вашей домашней папке. Чтобы перемещаться по файловой системе, нужно использовать `cd`. Примеры:

1. Чтобы попасть в корневой каталог, выполните:

```
cd /
```

2. Чтобы попасть в домашнюю папку, выполните:

```
cd ~
```

3. Чтобы переместиться на один каталог вверх, используйте:

```
cd ..
```

4. Для возврата в предыдущую посещённую директорию, используйте:

```
cd -
```

5. Для перемещения через несколько директорий за раз, нужно написать полный путь папки, в которую вы хотите переместиться. Например, команда:

```
cd /var/www
```

переместит вас в подкаталог `/www` каталога `/var/`. А команда:

```
cd ~/Рабочий\ стол
```

переместит вас на рабочий стол. В данном случае «`\`» обозначает экранирование пробела.

- Команда `cp` копирует файл. Например, команда:

```
cp text new
```

создаст точную копию файла «`text`» и назовёт её «`new`», при этом «`file`» никуда не исчезнет. Для копирования директории необходимо воспользоваться командой:



```
cp -r directory new
```

где опция `-r` обозначает рекурсивное копирование.

- Команда `mv` перемещает файл в другое место или просто переименовывает файл. Например, команда:

```
mv file text
```

переименует «file» в «text».

```
mv text ~/Рабочий\ стол
```

переместит «text» вам на рабочий стол, но не переименуют его. Чтобы всё-таки переименовать файл, нужно специально указывать ему новое имя. Вы можете заменять путь к вашей домашней папке на '~', чтобы ускорять работу. Помните, что при использовании `mv` вместе с `sudo`, терминал так же привязывает '~' к вашей домашней папке. Однако если вы включаете сессию суперпользователя в консоли с помощью `sudo -i` или `sudo -s`, то '~' будет ссылаться на корень диска, а не на ваш домашний каталог.

- Команда `rm` удаляет файл.
- Команда `rmdir` удаляет пустую папку. Чтобы удалить папку вместе со всеми вложениями, используйте `rm -r` вместо этого.
- `mkdir` Эта команда создаёт директорию. Команда `mkdir music` создаст вам папку, которая будет называться «music».
- `man` попробуйте эту команду, когда вы хотите прочитать встроенное описание другой команды. Например:

```
man man
```

покажет вам описание самой этой команды.

### Команды информации о системе:

- Команда **df** показывает объём занятого дискового пространства на всех смонтированных разделах. Наиболее полезна:

```
df -h
```

Поскольку использует для отображения Мегабайты (M) и Гигабайты (G), а не блоки. `-h` означает «human readable» («читаемый для человека»).

- **du** отображает объём дискового пространства, занятого конкретной папкой. Она может показывать информацию для всех поддиректорий отдельно или для папки в целом.

```
user@users-desktop:~$ du /media/floppy
```

- 1032 /media/floppy/files
- 1036 /media/floppy/

```
user@users-desktop:~$ du -sh /media/floppy
```

```
1.1M /media/floppy/
```

.. -s означает «summary» (в целом), а -h означает «human readable» («читаемый для человека»).

- **free** отображает объём свободной и занятой оперативной памяти. Команда:

```
free -m
```

показывает информацию в Мегабайтах.

- **top** отображает информацию о вашей системе, запущенных процессах и системных ресурсах, включая загрузку CPU, использование RAM и swap и количество запущенных процессов. Чтобы остановить top, нажмите **q**.

Также существует еще несколько замечательных и очень нужных утилит, оформленных в стиле top:

- **htop** аналог top, намного превосходящий по возможностям
- **iftop** информация об активных сетевых соединениях, скорость сетевой загрузки/отдачи
- **iotop** информация о процессах выполняющих активные дисковые операции

**htop**, **iftop**, **iotop** не входят в стандартный комплект дистрибутива Ubuntu, но легко устанавливаются из стандартных репозиториях.

- **uname** с опцией -a выводит всю системную информацию, включая имя машины, версию ядра и другие детали. Самое полезное из этого — проверка версии ядра.

- **lsb\_release** с опцией -a выводит информацию о версии Linux, которую вы используете. Например:

```
user@computer:~$ lsb_release -a
```

- **ifconfig** выводит отчёт о сетевых интерфейсах системы.
- **hwinfo** без ключей выводит очень длинный список всего оборудования, но существует ключи для получения конкретной информации о части оборудования --cpu --disk --memory и другие, см man по этой утилите.
- **lsusb** и **lspci** информация о USB- и PCI- устройствах

### Команда добавления нового пользователя:

- Команда создаёт нового пользователя в вашей системе с именем «newuser»:

```
adduser newuser
```

Чтобы назначить ему пароль, используйте:

```
passwd newuser
```

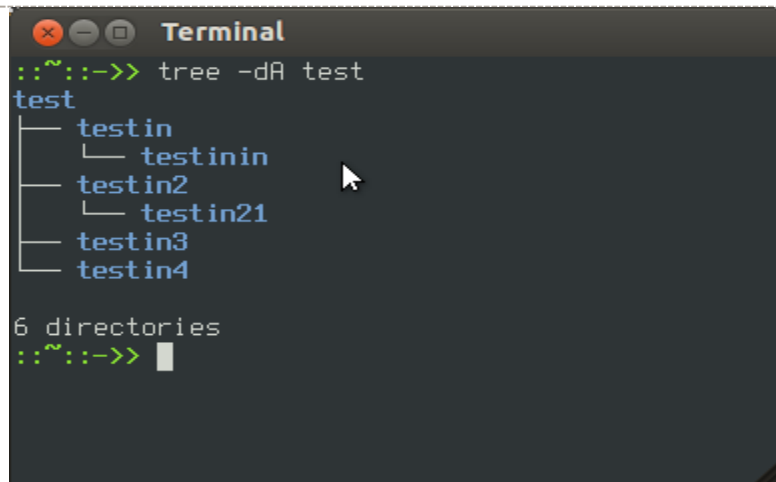
### Построение дерева каталогов (teree):

Утилита выводит дерево каталогов в красиво отформатированном виде. Пакет tree находится в репозиториях Ubuntu, поэтому установка выполняется одной строкой:

```
sudo apt-get install tree
```

Для вывода дерева каталогов команда должна иметь следующий вид:

```
tree -dA test
```



```
Terminal
::~::->> tree -dA test
test
├── testin
│   └── testinin
├── testin2
│   └── testin21
├── testin3
└── testin4

6 directories
::~::->> █
```

### Сохранение информации общего плана

Иногда необходимо коротко представить информацию «общего плана» о системе. Приведенными ниже командами формируется вывод такой информации в файл about\_system.txt в вашей «Домашней папке». Команды преобразованы к виду «для использования в терминале».

**Можно выполнить как сразу весь блок команд, так и по отдельности.**

Что выводит каждая из команд, указано в комментарии к каждой строке, справа.

```

echo "System: "`lsb_release -d --short` `uname -m`                > ~/about_system.txt # версия ОС и разрядность
echo "Kernel: "`uname -r`" DE: $XDG_CURRENT_DESKTOP Session: $GDMSSESSION" >> ~/about_system.txt # ядро, DE и вид сессии
echo "-----"                                                  >> ~/about_system.txt # линия, разделитель
echo "Processor: "`cat /proc/cpuinfo | grep "model name" -m1 | cut -c14-` >> ~/about_system.txt # Процессор
echo "Memory (Gb): "`free | grep Mem | awk '{print int($2/10485.76)/100}`" >> ~/about_system.txt # Размер ОЗУ
echo "Video: "`lspci -k | egrep 'VGA|3D' -A2`                    >> ~/about_system.txt # Видеокарты
echo "-----"                                                  >> ~/about_system.txt # разделитель

```

Полученный файл (/home/user/about\_system.txt) или его можно открыть через терминал:

```
gedit ~/about_system.txt
```

### Информация о дисках

Следующая команда отличается. Для вывода информации о дисках потребуется **ввод пароля администратора** последует запрос [sudo] password for user:

```
sudo parted -l | grep /dev/sd -B1 -A2                >> ~/about_system.txt # Hard,flash
```

Вывод содержит название устройства (модель), как устройство представлено в системе (/dev/sdX) и его размер, далее размер сектора и вид таблицы разделов. Будут отмечены не только жесткие диски, но и USB-флешки, если они в этот момент подключены (/dev/sdc в следующем примере):

Вывод информации о разделах на диске зависит от вида таблицы разделов. Если диски размечены, то можно воспользоваться командами fdisk и parted.

Далее к этим командам можно «дописать» и этот вывод в файл «краткой информации».

```
>> ~/about_system.txt
```

### МЕТОДИКА ВЫПОЛНЕНИЯ

1. Ознакомиться с теоретическими сведениями.
2. После загрузки ОС Linux и запроса имени ввести имя и пароль пользователя.

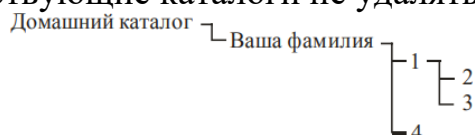
3. По окончании загрузки ОС запустить терминал.

**Все задания работы следует выполнить в режиме командной строки с помощью терминала.**

3. Привести пример работы команд, выбранных преподавателем (3 действия).

4. Создать файл с информацией общего плана о вашей системе, включая информацию о дисках и их разметки.

5. Создать в домашнем каталоге следующую структуру подкаталогов (существующие каталоги не удалять!):



6. Скопировать файл /etc/group в каталоги 1, 2, 3 и 4 используя абсолютные имена копируемого файла и каталога назначения.

7. С помощью утилиты file вывести на экран сведения о 3 - 4 различных файлах (в том числе из каталогов /bin и /dev).

8. Выполнить команду `ls -l /dev` используя таблицу обозначений типов файлов, перечислить типы файлов, хранящихся в каталоге /dev

Обозначения типов файлов

Символ	Тип файла
d	Каталог
l	Символьная ссылка
s	Сокет
b	Блочное устройство
c	Символьное устройство
p	Именованный канал

9. Используя справочную систему, ознакомиться с ключами утилиты `ls -R`, `-l` (единица), `-m`, `--color`, ключи, определяющие порядок вывода.

10. Создать жесткую и символическую ссылки для одного из созданных в п.2 файлов

Индивидуальные задания для бригад

Номер бригады	Задание
1	Вывести список имен файлов из /var, используя ключ <code>-l</code> Список упорядочить по размерам файлов. 2. Найти файлы, имена которых оканчиваются на pdf
2	Вывести список имен файлов из /bin, используя ключ <code>-l</code> Список упорядочить по датам создания

	2. Найти файлы, имена которых оканчиваются на jpg
3	Вывести список имен файлов из /sbin, используя ключ -l Список упорядочить по именам 2. Найти файлы, размеры которых превышают 25к (запись +25k)
4	Вывести список имен файлов из /tmp, используя ключ -l Список упорядочить по именам 2. Найти файлы, имена которых оканчиваются на text
5	Вывести список имен файлов из /usr, используя ключ -l Список упорядочить по размерам файлов. 2. Найти файлы, имена которых оканчиваются на jpg и размеры более 1к
6	Вывести список имен файлов из /bin, используя ключ -l Список упорядочить по датам создания 2. Найти файлы, размеры которых превышают 15к (запись +15k)
7	Вывести список имен файлов из /usr, используя ключ -l Список упорядочить по размерам файлов. 2. Найти файлы, размеры которых превышают 25к (запись +25k) и имена начинаются на s
8	Вывести список имен файлов из /var, используя ключ -l Список упорядочить по датам создания 2. Найти файлы, размеры которых превышают 25к (запись +25k) и имена начинаются на s, а заканчиваются на jpg
9	Вывести список имен файлов из /sbin, используя ключ -l Список упорядочить по размерам файлов 2. Найти файлы, размеры которых превышают 1М (запись +1m)
10	Вывести список имен файлов из /bin, используя ключ -l Список упорядочить по именам 2. Найти файлы, размеры которых превышают 5к (запись +5k)

11. Получить информацию о разделе, расположенном на устройстве /dev/sda1 с использованием **dumpe2fs**

12. Вывести информацию об операциях чтения/записи, а также общей статистики с помощью команды **iostat**.

Обратите внимание на то, что по умолчанию эта команда выводит информацию об операциях чтения/записи для всех устройств, а в верхней строке отображает общую статистику использования.

13. Вывести значения нескольких системных счетчиков через каждые 3 секунды с помощью команды **sar**:

```
$ sar 4 5
```

14. Показать параметры ядра с помощью команды **sysctl**.

Просмотр параметров ядра, относящихся к файловой системе  
`# sysctl -a | grep fs. | less`

15. Отобразить информацию, хранящуюся в inode с помощью команды `stat`  
`$ stat /etc/services`

16. Проверить с помощью команды `filefrag` фрагментацию файла. Опция `-v` позволяет получить более подробную информацию.

## **КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Типы файлов ОС Linux
2. Назначение утилиты `file`.
3. Структура дерева каталогов ОС Linux.
4. Отличия структуры файловых систем ОС Windows и Linux.
5. В чем отличие каталогов `/var` и `/tmp`.
6. Назначение утилиты `pwd`.
7. Назначение утилиты `cat`.
8. Назначение утилиты `ls`. Использование ключей `-F`, `-a`.
9. Утилита `mkdir`.
10. Утилиты копирования и перемещения файлов.
11. Жесткие ссылки: назначение и создание.
12. Создание файлов.
13. Символьные ссылки.
14. Удаление файлов и каталогов. Как восстановить ошибочно удаленный файл?
15. Назначение утилиты `id`.
16. Ярлыки объектов файловой системы.
17. Права доступа к файлу.
18. Суперпользователь и его права.
19. Назначение утилиты `sudo`.
20. Утилиты поиска файлов `locate` и `find`, их достоинства и недостатки.



## **ЛАБОРАТОРНАЯ РАБОТА №6 (2 часа)**

### **РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ СИСТЕМЫ НА ПРИМЕРЕ ОС UBUNTU LINUX**

**Цель работы** – получить навыки резервного копирования операционной системы, что позволит быстро восстановить работоспособность компьютера и всех установленных программ в случае сбоя в работе ОС или заражения компьютера вирусами.

#### **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

##### **1 Применение Snapshots**

Главным преимуществом резервного копирования считается возможность быстрого восстановления любых данных. Но бэкап далеко не всегда позволяет полностью восстановить работу диска или ОС. Такая проблема чаще всего обусловлена тем, что сразу же после бэкапа может происходить существенное изменение копируемых данных. Следовательно, многие файлы и приложения ОС изменяются и уже не могут быть связаны с копией. Поэтому перед тем как сделать бэкап, следует воспользоваться снимком. Под этим понятием подразумевается снимок или фотография. Snapshot позволяет «откатить» ОС до того момента, когда был сделан снимок.

Снимки применяются, когда требуется вернуться к одному состоянию виртуальной машины без необходимости создания новых.

Снимки содержат следующую информацию:

- настройки виртуальной машины;
- состояние дисков виртуальной машины;
- содержимое памяти виртуальной машины.

Снимок виртуальной машины может быть выполнен даже на выключенном устройстве. Выполнение снимка на несколько секунд приостанавливает работу компьютера. Но потом машина продолжает работать в своем прежнем режиме.

Snapshot можно сделать двумя способами:

- на странице с дисками;
- на странице с виртуальными машинами (кнопка «откатиться на предыдущий снимок»).

Все созданные снимки файловой системы можно найти на странице с дисками. Особенностью «снимков» является их размер. Снимки «вешат» на порядок меньше, чем диски. Еще одна особенность заключается в том, что сделанные снимки представляют собой своеобразную цепочку или дерево. Поэтому если удалить один из снимков, его «соседи» объединяются между собой. Процесс удаления ненужных снимков не занимает много времени.

Деревья снимков на Windows и Linux представлены на рисунке 1.



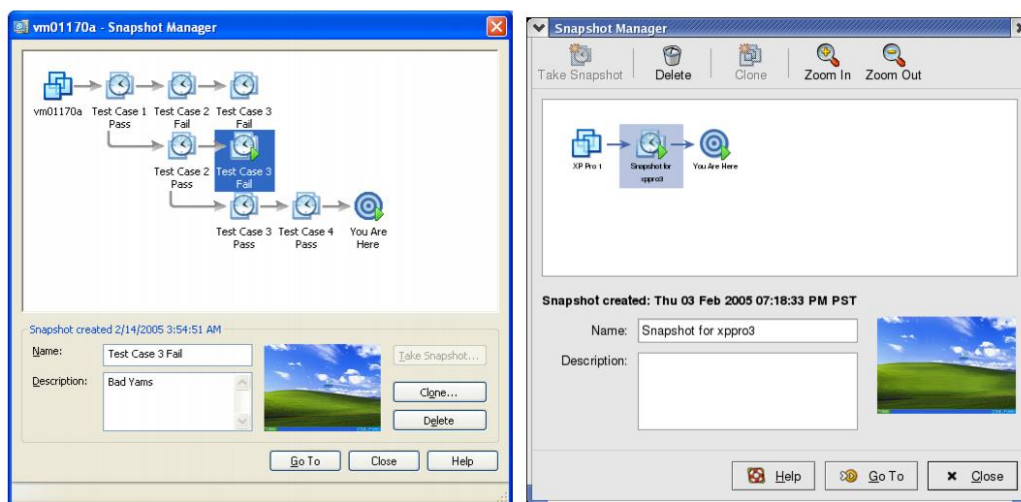


Рисунок 1 - Деревья снимков на Windows и Linux

Существенным преимуществом снимков является их многочисленность. Разработчики могут делать до 60 снимков, тем самым страхуя себя от случайной потери важной информации. Но снимкоты тоже «подгружают» ОЗУ, а потому существует вероятность возникновения определенных ограничений в функционировании виртуальной инфраструктуры. Поэтому любой разработчик должен знать, что длительное хранение snapshot не является целесообразным.

## 2 Виды резервного копирования

Резервное копирование подразделяется на различные виды в зависимости от задач, которые ставятся перед реализующим его программным обеспечением. В одних случаях пользователям необходимо лишь создавать копии важных файлов, хранящихся на диске, в других — создавать полноценные образы операционной системы с возможностью отката всех предыдущих изменений. При этом для системных администраторов предоставляются возможности централизованного хранения резервных копий данных, что упрощает контроль за версиями резервных копий и восстановление систем по мере необходимости. Естественно, в зависимости от выбранного типа резервного копирования задействуется тот или иной алгоритм сравнения и сохранения, когда информация в точности записывается на носитель с бекапом. Для восстановления файлов и данных также могут использоваться функции файловых систем, поддерживающих журналирование и протоколирование изменений, — вначале делается полный слепок файловой системы, а данные в резервную копию сохраняются по мере необходимости, если отдельные файлы помечены как измененные. Файловые системы с расширенной поддержкой контроля версии подходят для такого случая лучшего всего, поскольку существенно экономят место на резервном носителе. Кроме традиционного создания резервных копий файлов, которые не используются в данный момент, существуют алгоритмы резервирования в реальном времени. В этом случае резервное копирование происходит даже тогда, когда файл открыт в какой-либо программе. Такая возможность достигается благодаря использованию

снапшотов (snapshot) файловых систем и активно применяется, например, в системах виртуализации для работы с виртуальными дисковыми накопителями. Процесс резервирования данных может происходить несколькими путями. Рассмотрим наиболее распространенные из них.

## **2.1 Клонирование разделов и создание образов**

Клонирование подразумевает копирование раздела или разделов диска со всеми файлами и директориями, а также файловыми системами на резервный носитель, то есть создание полной копии данных на другом носителе. Это требует большого количества пространства на резервном носителе, но в то же время позволяет добиться наиболее полного резервирования отдельного ПК или диска с данными. Также особо следует упомянуть о клонировании системы в виде специального образа — виртуального накопителя, то есть отдельного файла, который может содержать в себе несколько разделов диска. Такой образ может быть создан средствами самой операционной системы. Он позволяет сократить объем данных, а также предоставляет возможность впоследствии работать с ним, как с обычным диском, либо подключать его к виртуальным машинам, что упрощает перенос операционных систем с одного сервера или компьютера на другой. Сегодня виртуальные образы набирают популярность за счет гибкости подключения, а также кроссплатформенности и легкого переноса с одного компьютера на другой. Как правило, клонирование или создание образа для резервного копирования происходит достаточно редко, поскольку объем, занимаемый резервной копией, очень большой. Подобные процедуры применяются в большинстве случаев именно для создания копии операционной системы со всеми файлами, а не для резервирования отдельных данных на диске. Для резервирования пользовательских данных, которые часто меняются или задействуются в работе, повсеместно используется другой тип резервного копирования — полное файловое резервирование.

## **2.2 Полное файловое резервирование**

Такой тип резервного копирования подразумевает создание дубликатов всех файлов на носителе простым методом — копированием из одного места в другое. Полное файловое резервирование вследствие длительности процесса обычно проводится в нерабочее время, что объясняется слишком большими объемами данных. Такой тип резервирования позволяет сохранить важную информацию, но из-за больших сроков резервирования он не очень подходит для восстановления быстро меняющихся данных. Полное файловое копирование рекомендуется проводить не реже раза в неделю, а еще лучше чередовать его с другими типами файлового копирования: дифференциальным и инкрементным.

## **2.3 Дифференциальное резервирование**

Дифференциальное резервирование предполагает копирование только тех файлов, что были изменены с последнего полного резервного копирования. Это позволяет уменьшить объем данных на резервном носителе и при необходимости ускорить процесс восстановления данных. Поскольку дифференциальное

копирование обычно производится гораздо чаще, чем полное резервное копирование, оно очень эффективно, так как позволяет восстанавливать те данные, которые подверглись изменению совсем недавно, и отслеживать историю изменения файлов с момента полного копирования.

## **2.4 Инкрементное резервирование (Incremental backup)**

Инкрементное резервирование несколько отличается от дифференциального. Оно подразумевает, что при первом запуске происходит резервное копирование только тех файлов, которые были изменены с тех пор, как в последний раз выполнялось полное или дифференциальное резервное копирование. Последующие процессы инкрементного резервирования добавляют только те файлы, которые подверглись изменению с момента предыдущей процедуры резервирования. При этом изменившиеся или новые файлы не замещают старые, а добавляются на носитель независимо. Конечно, в этом случае история изменения файлов увеличивается с каждым этапом резервирования, а процесс восстановления данных для этого типа резервирования происходит гораздо дольше, поскольку необходимо восстановить всю историю изменений файлов, шаг за шагом. Однако при дифференциальном резервировании процесс восстановления более прост: восстанавливается основная копия и в нее добавляются последние данные дифференциального резервирования.

Многие программные пакеты для резервирования используют различные виды резервирования, а зачастую совмещают их с целью большей эффективности и экономии места.

## **3 Резервное копирование и восстановление системы Ubuntu**

<https://help.ubuntu.ru/wiki/backup>

### **МЕТОДИКА ВЫПОЛНЕНИЯ**

1. Создать снапшот системы Ubuntu.
2. Настроить автоматическое создание снапшотов системы:

(по вариантам)

- раз в неделю,
- раз в месяц,
- раз в день,
- каждый час,
- после перезагрузки.

Причем снимки должны удаляться, если они старше:

(по вариантам)

- месяца,
- трех месяцев,
- недели,
- суток,
- на усмотрение админа.

3. Восстановить систему из снапшота.

4. Создать резервную копию с помощью системных команд.
5. Восстановить систему из резервной копии с помощью системных команд.

## **КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Какие различия между Backup и Snapshot?
2. Опишите преимущества и недостатки Snapshot.
3. Опишите преимущества и недостатки Backup.
4. Преимущества восстановления из Snapshot.
5. Преимущества восстановления из Backup.
6. Какие утилиты позволяют сделать снапшот системы?
7. Какие утилиты позволяют сделать резервную копию системы?
8. Что такое репликация?

## ЛАБОРАТОРНАЯ РАБОТА №7 (2 часа) АДМИНИСТРИРОВАНИЕ ОС UBUNTU. Настройка загрузчика

**Цель работы** – получить начальные навыки администрирования ОС Linux

### ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

GRUB (GRand Unified Bootloader) — программа-загрузчик операционных систем. GRUB является эталонной реализацией загрузчика, соответствующего спецификации Multiboot и может загрузить любую совместимую с ней операционную систему. Среди них: Linux, FreeBSD, Solaris и многие другие. Кроме того, GRUB умеет по цепочке передавать управление другому загрузчику, что позволяет ему загружать Windows и другие системы.

**Multiboot Specification** (Спецификация мультизагрузки) — открытый стандарт, созданный Фондом свободного программного обеспечения. Спецификация описывает интерфейс между ядром операционной системы и загрузчиком, следуя которому один и тот же универсальный загрузчик может применяться для загрузки нескольких разных операционных систем, установленных на одном и том же компьютере, если ядра этих ОС поддерживают спецификацию Multiboot.

После настройки GRUB пользователь при включении компьютера видит список операционных систем, которые установлены на его компьютер и которые можно загрузить, выбрав подходящую. GRUB позволяет пользователю при загрузке задавать произвольные параметры и передавать их в ядро Multiboot-совместимой ОС для дальнейшей обработки.

В работе под GRUB подразумевается GRUB2, который используется в операционных системах семейства Ubuntu начиная с версии 9.10, до него использовался GRUB первой версии, сейчас известный как GRUB Legacy. GRUB 2 полностью переписан с нуля и не имеет ничего общего с GRUB Legacy.

GRUB2 поддерживает все необходимые функции, в том числе и UEFI (Boot menu), а также очень настраиваемый.

### 1 Таблицы разделов диска

Загрузчик может быть установлен в различные таблицы разделов диска. Сейчас самые используемые это **GPT (GUID Partition Table)** и **MBR (Master Boot Record)**. Установка загрузчика Grub немного отличается для каждой из этих таблиц, учитывая их особенности. GPT - более новая и функциональная таблица разделов, MBR - уже устаревшая, но до сих пор часто используемая.

**MBR** находится в самом начале диска, точнее занимает первые 512 байт. Она содержит информацию, о том, какие логические и расширенные разделы есть на этом устройстве. Кроме того, в MBR находится исполняемый код, который может сканировать разделы в поисках операционной системы, а также инициировать загрузку операционной системы. Для Windows - это загрузчик WIndows, в Linux там находится код инициализации Grub. Поскольку места там очень мало, обычно

этот код используется только для инициализации основного загрузчика, расположенного где-нибудь на диске.

Разделы начали называть первичными (primary), а также добавили расширенные (extended) и логические (logical). Один расширенный раздел, может содержать несколько логических, таким образом вы сможете создать необходимое количество разделов, рисунок 1.

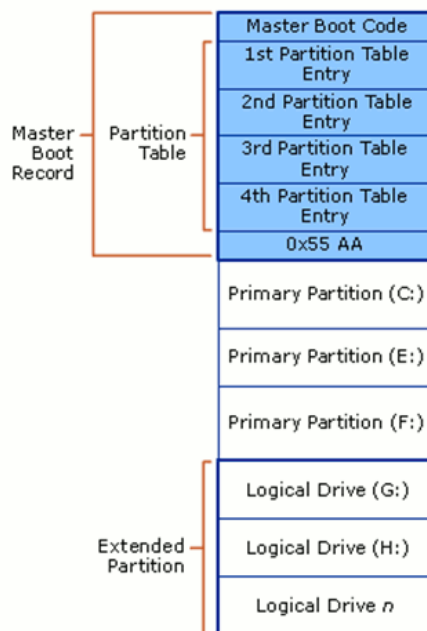


Рисунок 1 - Master Boot Record

**Достоинство.** MBR можно назвать полную совместимость со многими операционными системами в том числе Windows, включая старые версии, Linux и MacOS.

**Недостатки.** MBR использует 32-битную адресацию пространства, поэтому вы сможете работать только с дисками размером до двух терабайт. Конечно, со временем появились способы поддерживать и большие объемы, но работать с ними она будет не так хорошо. Еще один минус в том, что MBR расположена только в начале диска и если вы ее случайно затрете, то диск станет полностью нечитаемым.

**GPT** это современный стандарт управления разделами на жестком диске. Это часть стандарта EFI (Extensible Firmware Interface). В GPT используется адресация LBA. Это блочная адресация, каждый блок имеет свой номер, например, LBA1, LBA2, LBA3, и так далее, при чем адреса MBR автоматически транслируются в LBA, например, LBA1 будет иметь адрес 0,0,1 и так далее.

GPT не содержит кода загрузчика, она рассчитывает, что этим будет заниматься EFI, здесь размещена только таблица разделов. В блоке LBA0 находится MBR, это сделано для защиты от затирания GPT старыми утилитами работы с дисками, а уже с блока (LBA1) начинается сама GPT. Под таблицу разделов резервируется 16 384 байт памяти, по 512 на блок, а это 32 блока, таким образом первые разделы начнутся с блока LBA34 (32+1MBR+1GPT), рисунок 2.

## GUID Partition Table Scheme

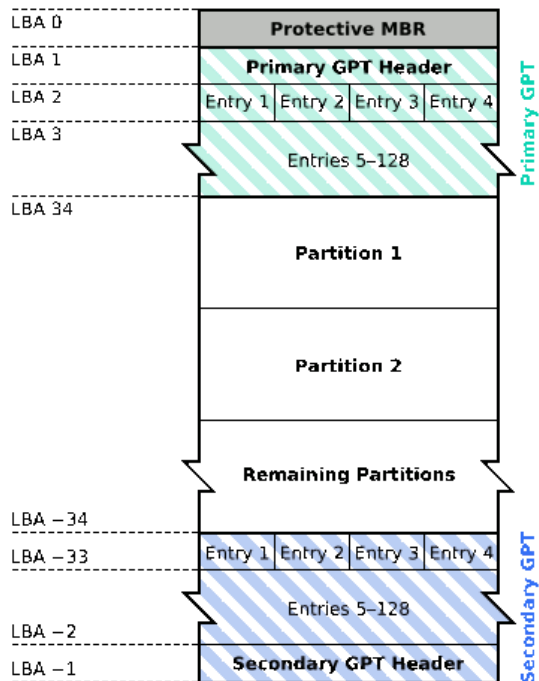


Рисунок 2 - GUID Partition Table

**Достоинства.** Количество разделов ограничено только операционной системой. Ядро Linux поддерживает аж до 256 разделов. Благодаря адресации LBA, GPT в отличие от MBR может создавать разделы до 9,4 ЗБ.

Кроме того, служебная информация GPT дублирована, она размещается не только в начале диска, но и в конце, таким образом во многих случаях при повреждении GPT может сработать автоматическое восстановление.

GPT поддерживает юникод, поэтому можно задавать имена и атрибуты разделам. Имена могут быть заданы на любом поддерживаемом языке, и вы сможете обращаться к дискам по этим именам. Для дисков используются глобальные уникальные идентификаторы GUID (Globally Unique Identifier), это одна из вариаций UUID с большей вероятностью уникальных значений, может также использоваться для идентификации дисков вместо имен.

**Поддержка операционных систем.** MacOS и новые версии Windows, начиная от Windows 8 используют GPT по умолчанию. Вы не сможете установить MacOS в системе с MBR, она будет работать на этом диске, но вы не сможете ее туда установить. Windows поддерживает как MBR так и GPT, начиная с версии 8, более ранние версии установить на GPT не удастся, но работать с GPT можно начиная с XP.

Ядро linux включает поддержку как MBR так и GPT, только для установки на GPT придется использовать загрузчик GRUB2.

### Какая таблица разделов используется на ПК.

Если у вас предустановлена Windows 10 на ноутбуке, то используется GPT, но в других случаях необходимо узнать.



В Linux использовать для этого утилиту fdisk:

```
sudo fdisk -l
```

Disk /dev/sda: 465,8 GiB, 500107862016 bytes, 976773168 sectors

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: dos

Disk identifier: 0x1c50df99

Disklabel type: dos - значит, что у вас используется MBR, в GPT так будет и написано - GPT. Также узнать GPT или MBR используется можно с помощью программы GPARTED.

## 2 Восстановление GRUB

Разумеется, с ним при настройке могут возникнуть проблемы, вплоть до полного отказа операционной системы.

Одна из самых распространенных причин - это неправильный порядок установки двух операционных систем (Linux и Windows). Стоит соблюдать правильную последовательность: сначала Windows, потом уже Linux.

Если, например, сделать наоборот, то как раз-таки Grub будет поврежден; система будет грузиться напрямую в Windows, а дистрибутив Linux останется недоступным.

Grub может сломаться и по другим причинам. Например, из-за попыток ручного изменения параметров запуска (при недостатке опыта), в таком случае нужно будет либо вручную убирать лишнее, либо полностью переустанавливать Grub.

В случае, если система есть, но поврежден загрузчик. Восстановить GRUB можно с помощью LiveCD/USB или при его отсутствии с помощью консоли

[https://help.ubuntu.ru/wiki/%D0%B2%D0%BE%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5\\_grub](https://help.ubuntu.ru/wiki/%D0%B2%D0%BE%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_grub)

Сначала вставьте носитель с LiveCD системой в дисковод или порт USB, если это флешка. Для загрузки с носителя, сначала надо зайти в меню BIOS и выставить приоритет загрузки с внешнего устройства.

Для запуска BIOS нажмите Del, F2, F8 или Shift +F2. В открывшемся меню найдите раздел Boot, и в пункте Boot Device Priority или 1st Boot Device или Boot Option #1 выберите нужное устройство. Дальше перейдите на вкладку Exit и выберите Exit & Save settings. Дальше начнется загрузка образа.

[https://help.ubuntu.ru/wiki/%D0%B2%D0%BE%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5\\_grub#%D0%B2%D0%BE%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5\\_%D1%81\\_%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E\\_livecdusb](https://help.ubuntu.ru/wiki/%D0%B2%D0%BE%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_grub#%D0%B2%D0%BE%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81_%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E_livecdusb)

В случае, если одна система удалена (например, Ubuntu), осталась другая



(например, Windows).

В подобных случаях все напрямую зависит от способа установки ОС на машину. В случае с EFI - достаточно выйти в системное загрузочное меню UEFI (Boot menu), найти там загрузчик Windows, и стартовать с него. Если запускается - то переместить на первое место в загрузочном разделе настроек UEFI/BIOS. Если же доступа к загрузочному меню или настройкам BIOS нет, то надо восстанавливать Grub из его консоли. Определить какой диск с какой разметкой GPT или MBR. В зависимости от выводов команд, выяснить как установлена загружаемая система.

Для UEFI. Нужно выяснить, где ESP раздел, найти на нем бинарный файл /EFI/Microsoft/Boot/bootmgfw.efi (регистр символов в полном имени файла может быть иным - смотрите вывод команды ls и пользуйтесь автодополнением по TAB). Найти раздел EFI. Ввести ls.

```
(hd0) (hd0,gpt1) (hd0,gpt2) (hd0,gpt3) ...
```

Далее перебрать выводы команд

```
ls (hd0,gpt1)
```

```
ls (hd0,gpt2)
```

```
ls (hd0,gpt3)
```

...

Найти на какой из вводов ответ будет содержать "Тип файловой системы fat, UUID XXXX-XXXX". С высокой долей вероятности это он и есть. Далее вывод команды ls, добавив слеш после имени раздела, например "**ls (hd0,gpt1)/**". Если в выводе EFI - то найден, и продолжаем далее, если нет - перебрать выводы команды ls без слешей. Найти следующий раздел fat. Допустим, на разделе gpt1 есть каталог EFI. Теперь ввести последовательно команды, выясняя путь к загрузочному конфигу (проверяем раздел (hd0,gpt1), применяя автодополнением по TAB вместо Enter - сам grub покажет какие файлы он видит). Далее последовательность команд, чтоб с этого файла загрузиться:

```
insmod part_gpt
```

```
insmod fat
```

```
chainloader (hd0,gpt1)/EFI/Microsoft/Boot/bootmgfw.efi
```

Должна пойти загрузка Windows.

**Boot Repair.** Самый простой способ восстановить загрузчик Grub - это использовать утилиту Boot Repair. Для её запуска вам понадобится LiveCD с Ubuntu или другим дистрибутивом. Сначала загрузитесь в Live среду, затем добавьте PPA к системе:

```
sudo add-apt-repository -y ppa:yannubuntu/boot-repair
```

После этого установите утилиту:

```
sudo apt install boot-repair
```

Для запуска выполните:

```
sudo boot-repair
```

Далее для восстановления загрузчика достаточно нажать кнопку **Рекомендуемый способ восстановления**. Затем просто дождитесь завершения процесса восстановления. Далее перезагрузка.

## 4 Настройка GRUB

### Настройка GRUB через файлы конфигурации

Настройка GRUB производится через основной файл конфигурации «/boot/grub/grub.cfg» это основной итоговый файл, коорый отображает все изменения GRUB. **Нумерация дисков идет с нуля, а нумерация разделов - с единицы!**

Кроме файла «grub.cfg», отвечающего за загрузочное меню, имеются файл «/etc/default/grub», «40\_custom» и папка «/etc/grub.d».

#### */etc/default/grub*

Данный файл содержит в себе основные настройки для GRUB. Через него, собственно, они и изменяются. Для наглядности ниже приводится примерное содержимое этого файла:

```
GRUB_DEFAULT=6
#GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT="2"
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""
# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console
# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480
# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to
Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entrys
#GRUB_DISABLE_LINUX_RECOVERY="true"
```

Файл представляет из себя набор опций в понятном формате **ОПЦИЯ=ЗНАЧЕНИЕ**.

Наиболее часто встречающаяся потребность при настройке GRUB - изменение стандартного пункта загрузки и/или времени показа меню. Рассмотрим же, как это делается.

#### */etc/grub.d*

Эта папка содержит в себе скрипты, которые используются для создания

файла «grub.cfg». При обновлении GRUB они находят все установленные на компьютере системы и ядра и формируют в файле «grub.cfg» меню загрузки, которое мы и видим. Два основных из них:

- «10\_linux» и «30\_os-prober» отвечают за поиск Linux ядер и остальных ОС на других разделах соответственно.
- Файл «40\_custom» позволяет добавлять свои пункты загрузки. Это может быть полезно, если вы, например, хотите добавить какие-то особые варианты загрузки системы.

Файл «40\_custom» должен заканчиваться пустой строкой, иначе последний пункт не будет отображаться в меню.

### **Команды консоли GRUB.**

Чтобы попасть в консоль, нужно нажать клавишу `C` во время отображения меню загрузки.

`ls` -универсальная команда при использовании в чистом виде выдает список жестких дисков и разделов. Также она может быть использована как одноименная команда в Linux - для вывода содержимого папки. например: `ls /boot/grub`

Еще одно полезное свойство командны «`ls`» - она позволяет получить информацию о любом разделе: `ls (hd0,5)`

Команда сообщит тип файловой системы на разделе, метку раздела (если таковая имеется), UUID и дату последнего изменения данных на разделе (в формате UTC).

`cat` -данная команда выводит содержимое заданного файла, используется в формате:

`cat /путь/имя_файла`

`linux` - аналог команды «`kernel`» в [GRUB Legacy](#). Загружает указанное Linux-ядро:

`linux файл_ядра опция1=значение опция2 опция3`

Например, так:

`linux /boot/vmlinuz-2.6.32-020632-generic root=/dev/sda5 single`

`initrd` - загружает указанный `initrd`-образ. Используется так:

`initrd /boot/initrd.img-2.6.32-020632-generic`

Версия `initrd` должна соответствовать версии загружаемого ядра.

`chainloader` - передает управление загрузкой по цепочке другому загрузчику (загрузчик ищется на заданном в качестве `root` разделе). В общем случае требует указания файла для загрузки:

`chainloader /путь/имя_файла`

Для (загрузчика Windows) можно использовать:

`chainloader +1`

`boot`

`root` - при использовании без параметров сообщает, какой раздел сейчас используется в качестве корневого и тип файловой системы на этом разделе, также

команда может быть использована для задания другого root-раздела. Раздел задается в «grub device» - формате »(hd\*,\*)». например:

```
root (hd0,5)
```

После задания раздела команда сообщит новый root-раздел и тип файловой системы. Примечание: «root hd(\*,\*)» не всегда корректно срабатывает. более предпочтительным вариантом является «set root»

set - весьма универсальная команда для изменения различных параметров. Служит для задания значений переменных и используется в формате:

```
set переменная=значение
```

Наиболее необходимое ее применение - для задания root-раздела, например:

```
set root=(hd0,5)
```

Также с ее помощью можно, например, «на лету» изменить цвет текста в меню и консоли, что позволяет опробовать цветовую схему перед установкой ее в качестве основной. Для этого изменяем переменные «color\_normal» - для обычного пункта (и текста в консоли) и «color\_highlight» для выделенного пункта соответственно. Например:

```
set color_normal=magenta/green
```

```
set color_highlight=light-blue/black
```

search - служит для поиска раздела по UUID, метке или заданному файлу. Имеет следующие ключи:

- -u (или -fs-uuid) - поиск раздела по UUID
- -l (или -label) - поиск по метке раздела
- -f (или -file) - поиск по указанному файлу
- -n (или -no-floppy) - не проверять флоппи-дисковод (чтоб не трещал)
- -s (или -set) - установить найденный раздел в качестве значения

заданной переменной.

lsfonts - команда отобразит список загруженных в настоящий момент шрифтов.

help - при использовании в чистом виде выведет список доступных команд. В формате:

```
help r - выведет справку по всем командам, начинающимся на «r».
```

```
help search - отобразит справку по команде «search»
```

```
halt - выключение компьютера.
```

```
reboot - перезагрузит компьютер.
```

background\_image - позволяет «на лету» изменить фоновое изображение. Используется в формате:

```
background_image /путь/имя_файла
```

Дает возможность посмотреть на выбранную картинку в действии, избегая лишних перезагрузок. В сочетании с заменой цветов через set позволит довольно быстро подобрать подходящий вариант оформления.

Данная команда не заменит ваши настройки оформления, фон будет изменен лишь на текущий сеанс.

При использовании без параметров сбросит текущее фоновое изображение. Однако, заданные цвета останутся, так что если у вас черный цвет текста - на черном фоне вы его не увидите.

terminal\_output.console - позволяет переключиться на обычную черно-белую цветовую гамму. Весьма полезно при работе с консолью в том случае, если у вас установлено фоновое изображение. Картинка на фоне - это, конечно, красиво, но на некоторых участках фона может быть не виден текст.

Полезные ссылки официального сообщества Ubuntu:

<https://help.ubuntu.ru/wiki/grub>

[https://help.ubuntu.ru/wiki/%D0%B2%D0%BD%D0%B5%D1%88%D0%BD%D0%B8%D0%B9\\_%D0%B2%D0%B8%D0%B4\\_grub](https://help.ubuntu.ru/wiki/%D0%B2%D0%BD%D0%B5%D1%88%D0%BD%D0%B8%D0%B9_%D0%B2%D0%B8%D0%B4_grub)

## МЕТОДИКА ВЫПОЛНЕНИЯ

1. Отобразите какая таблица разделов используется на ПК.
2. Создайте систему LiveCD с Ubuntu той же разрядности, что и установленная система. Обязательно всем! Это необходимо в случае восстановления Grub. LiveUSB - <https://help.ubuntu.ru/wiki/liveusb> или с помощью **UNetbootin** (Universal Netboot Installer) <https://help.ubuntu.ru/wiki/unetbootin>
3. Настроить GRUB через конфигурационные файлы: выбрать пункт, который нужно загружать по умолчанию, время, на протяжении которого будет показываться меню, а также параметры ядра для каждого пункта. Настроить внешний вид - выбрать одну из доступных тем, установить разрешение, или задать пользовательские параметры цвета, шрифтов и фоновый рисунок. Настроить возможность загрузки с другим ядром, создать дополнительное меню загрузки, сделать заставку во время загрузки системы. Включить текстовый режим загрузки системы.

**Примечание.** Некорректно примененная тема способна «свалить» загрузку!

4. Очистить загрузочное меню: удалить старые ядра.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие существуют таблицы раздела дисков?
2. В чем их преимущества и недостатки?
3. Сколько байт резервируется под таблицу разделов?
4. Что описывает Multiboot Specification?

## **ЛАБОРАТОРНАЯ РАБОТА №8 (2 часа)**

### **СОЗДАНИЕ ДИСТРИБУТИВА СБОРКИ СПЕЦИАЛИЗИРОВАННОЙ ОС**

**Цель работы** – создать свободно распространяемую специализированную под определенные задачи ОС.

#### **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Дистрибутив (distribute — распространять) — форма распространения программного обеспечения. В данном случае, форма распространения операционной системы Linux. Дистрибутив Linux состоит из ядра операционной системы и набора программ, настроенных специальным образом.

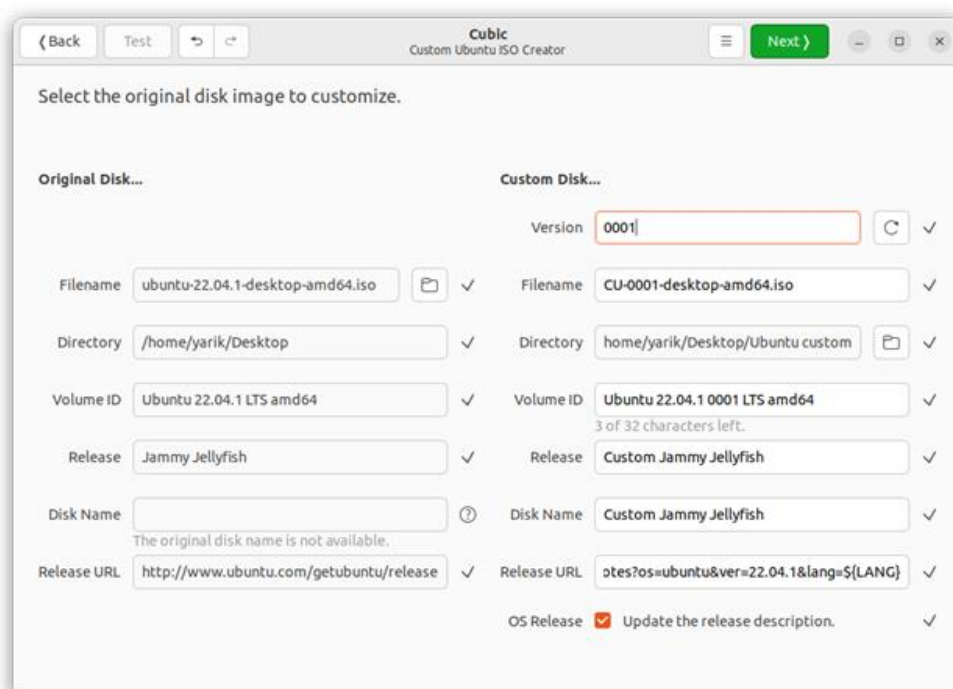
Дистрибутив Linux — установочный пакет для развёртывания операционной системы, состоящей из ядра Linux, утилит GNU, дополнительного ПО и диспетчера пакетов. Он также может включать в себя пакет для установки дисплейного сервера и развёртывания среды рабочего стола.

Компании Debian и Ubuntu занимаются именно таким распространением ядра Linux со всем необходимым программным обеспечением (таким, как сетевой менеджер, диспетчер пакетов, среда рабочего стола и т.д.) в качестве полнофункциональной операционной системы.

Дистрибутив также осуществляет установку необходимых обновлений в процессе установки, а развёрнутая на базе определённого дистрибутива ОС — в дальнейшем.

В предыдущих лабораторных работах вы определили цели и задачи своей ОС. Теперь, когда всё настроено и установлено, необходимо создавать свою сборку. Для этого необходимо свободное место на разделе с папкой */home* не менее 25 ГБ.

Для создания сборки можно использовать утилиту Cubic, позволяет создать образ с предустановленными программами, установить другое ядро, исключить некоторые пакеты из Ubuntu для облегчения или оптимизации ОС, настроить авто установку и т.д.



## МЕТОДИКА ВЫПОЛНЕНИЯ

Создать сборку ОС любым способом. В созданных сборках должны присутствовать программы, установленные в Лабораторной работе №5, допускается повторение не больше двух программ (из пяти) в пределах одной группы.

1. Установить программу Cubic, указать рабочую директорию проекта, название сборки.
2. Установить ПО и сделать необходимые настройки.
3. Настроить dash-панель через ~/.config/dconf и задать иконку учетной записи.
4. Определить состав пакетов сборки.
5. Выбрать ядро сборки
6. Выбрать уровень сжатия. Чем сильнее сжатие, тем медленнее работа и наоборот.
7. Протестировать запуск образа.

## ЛАБОРАТОРНАЯ РАБОТА №9 (4 часа) ДЕБИАНИЗАЦИЯ, СЕРТИФИКАЦИЯ И ИНТЕГРАЦИЯ В ХРАНИЛИЩЕ РЕПОЗИТОРИЕВ

**Цель работы** - научиться собирать пакеты Debian из исходников.

### ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Дистрибутив GNU/Linux — общее определение операционных систем, использующих ядро Linux, готовых для конечной установки на пользовательское оборудование. Поставляемая в рамках дистрибутива операционная система состоит из ядра Linux и, как правило, включает в себя набор библиотек и утилит, выпускаемых в рамках проекта, а также графическую подсистему и набор приложений, таких как редакторы документов и таблиц, проигрыватели мультимедиа, системы для работы с базами данных и т. д. Небольшие дистрибутивы могут не включать в состав графическую подсистему, и, в целях экономии, предлагать альтернативы.

В настоящее время существует большой список дистрибутивов Linux; более половины из них поддерживаются в актуальном состоянии, что обеспечивается регулярным выпуском обновлений разработчиками дистрибутива.

Так как ядро и сопутствующее ему программное обеспечение, в основном, являются открытыми, тот или иной дистрибутив может быть установлен на самый широкий спектр аппаратного обеспечения, например: домашний компьютер, сервер, ноутбук или нетбук, смартфон или планшетный компьютер. Кроме этого, некоторые из дистрибутивов специально предназначены для разворачивания в средах с ограниченными ресурсами.

Debian и системы на основе Debian, такие как Ubuntu, используют Advanced Package Tool, сокращенно APT, для установки, обновления, обновления и удаления программного обеспечения из командной строки. Обычно менеджер пакетов APT хранит список репозитория в файле с именем `/etc/apt/sources.list` и в любом файле с суффиксом `.list` в каталоге `/etc/apt/sources.list.d/`. Когда мы устанавливаем пакет, команда `apt` извлекает двоичную или предварительно скомпилированную версию данного пакета из этих репозитория. Помимо установки двоичных пакетов, APT также позволяет загружать исходный код пакета. Таким образом, вы можете затем добавить некоторые функции в исходный код, собрать пакет из исходного кода и, наконец, установить измененную версию пакета. В данной лабораторной работе вы попробуете создавать пакеты `debian` из исходного кода в Debian, Ubuntu и других системах на основе `apt`.

Для сборки программы из исходников, то необходимо изучить, как дебианизировать исходники. Стандартные методы

`./configure && make && make install`

Удалить программы, установленные командой [`make install`](#), можно только командой `make uninstall`.

Но не все исходники это поддерживают, а что ещё чаще - исходники удаляют



после установки, тогда удалить программу можно только вручную. Но чтобы это сделать, нужно точно знать, что и куда установилось. АРТ не знает ничего о программах, установленных вручную. Соответственно, могут быть конфликты, или просто непонятные ошибки. Очень часто исходники по умолчанию «рассчитаны» на определённый дистрибутив, или, наоборот, рассчитаны только на установку из исходников, при этом выполняются разного рода «удобные» настройки в конфигурационных файлах.

Полное Руководство начинающего разработчика Debian доступно <https://www.debian.org/doc/manuals/maint-guide/index.en.html>

## 1 Шифрование в Linux

В мире свободного программного обеспечения, для предотвращения «краж» или «подделок», принято подписывать свои «ценные» данные электронным ключом, открытая часть которого хранится на общедоступных серверах и позволяет другим пользователям легко выяснить подлинность и целостность тех или иных данных.

Один из способов защиты наиболее важной информации - шифрование файлов и каталогов. Во время шифрования содержимое файлов перемешивается с избыточными данными в соответствии с установленным алгоритмом, таким образом, что расшифровать его можно только имея специальный пароль или ключ.

### 1.1 Создание ключа шифрования

**GNU Privacy Guard** или просто **GPG** - инструмент с открытым исходным кодом для шифрования файлов, который может быть использован для шифрования любого файла из командной строки или в графическом режиме:

gpg опции файл параметры

Опции указывает что необходимо сделать с файлом, как это сделать и какие возможности использовать. Основные опции:

- h - вывести справку по утилите;
- s, --sign - создать цифровую подпись, эта опция используется вместе с другими опциями для шифрования;
- clearsign - подписать незашифрованный текст;
- e, --encrypt - зашифровать данные, с помощью ключа;
- c, --symmetric - зашифровать данные, с помощью пароля;
- d, --decrypt - расшифровать данные, зашифрованные с помощью ключа или пароля;
- verify - проверить подпись;
- k, --list-keys - вывести доступные ключи;
- list-sigs - вывести доступные подписи;
- fingerprint - вывести все ключи вместе с их отпечатками;
- delete-key - удалить ключ;
- delete-secret-key - удалить секретный ключ;
- export - экспортировать все ключи;
- export-secret-keys - экспортировать все секретные ключи;

- import - импортировать ключи;
- send-keys - отправить ключи на сервер, должен быть указан сервер ключей;
- recv-keys - получить ключи от сервера ключей;
- keyserver - указать сервер ключей;
- fetch-keys - скачать ключи;
- gen-key - создать ключ;
- sign-key - подписать ключ;
- passwd - изменить пароль для ключа.

**Симметричный шифр** - самый простой и в то же время надежный способ шифрования файлов linux. Расшифровать файл сможет любой у кого есть пароль. Для использования просто запустите терминал и выполните команду `gpg` с параметром `-c`:

```
gpg -c имя файла
```

Утилита создаст файл с расширением `gpg`. Для расшифровки используйте:

```
gpg имя_файла.gpg
```

**Асимметричный шифр** более надежный так как для шифрования используется два ключа - публичный, собственно для шифрования, которым может воспользоваться любой, и приватный - для расшифровки. Причем файл можно расшифровать только с помощью приватного ключа, даже если вы зашифровали файл, без приватного ключа вы его не расшифруете.

Сначала необходимо настроить `gpg`, создать пару ключей, для этого наберите:

```
gpg --gen-key
```

и далее ответить на вопросы, выбрав при этом требуемый тип ключа и размер, срок действия. Типы ключей: RSA and RSA (default), DSA and ElGAMAL, DSA(только для подписывания), RSA(только для подписывания).

В каталоге `~/.gnupg` появятся два файла. В файле `pubring.gpg` публичный ключ, а в `secring.gpg` приватный:

```
ls ~/.gnupg/
```

Список доступных ключей `gpg --list-keys`

Если необходимо шифровать файлы на другом компьютере необходимо экспортировать публичный ключ, для этого есть опция `-a`:

```
gpg -a -o gpgkey.asc --export имя_ключа
```

Далее передать файл на целевое устройство и импортируем ключ:

```
gpg --import gpgkey.asc
```

После импорта ключа уровень доверия к нему по умолчанию будет неизвестным поэтому при каждом шифровании `gpg` будет спрашивать действительно ли вы доверяете этому ключу. Чтобы этого избежать нужно указать уровень доверия. Для этого воспользуйтесь редактором ключей:

```
gpg --edit-key Username
```

Для выбора уровня доверия введите команду `trust`:

```
gpg> trust
```

Для своих ключей можно использовать пункт абсолютно доверяю с номером 5, вы же знаете, что это именно ваш ключ.

Теперь можно переходить к шифрованию. Для того чтобы зашифровать файл `linux` используйте команду:

```
gpg -e -r ID_пользователя имя_файла
```

ID пользователя нужно указывать тот что вы использовали при создании ключа. Для расшифровки используйте:

```
gpg -d имя_файла.gpg
```

Для каталогов действия аналогичны только сначала нужно создать архив с помощью `tar`:

```
tar -cf - каталог | gpg -e -r ид_пользователя
```

А для расшифровки:

```
gpg -d каталог.gpg | tar -xvf
```

## 1.2 Подписи и шифрование

Для проверки подлинности файлов может использоваться не шифрование, а подпись. Тогда на основе файла и ключа создается отпечаток, который записывается в файл. Если файл будет изменен, то отпечаток уже не совпадет.

Вы можете подписать файл с помощью опции `--sign`:

```
gpg --sign имя_файла
```

Если вы не хотите изменить исходный файл, то можно создать подпись в отдельном файле:

```
gpg -b имя_файла
```

Тогда в каталоге, рядом с файлом появиться файл `.sig` с подписью. Далее, чтобы проверить достаточно использовать команду `verify`:

```
gpg --verify textfile.sig textfile
```

Если файл был изменен, то вы увидите, что подпись не сходится.

## 1.2 Ключ для бэкапота

Бэкапорт - создание пакета из последней версии дистрибутива в более раннюю, но поддерживаемую.

Если вы бэкапотируете пакет, дебианизированный не вами, обязательно нужно изменить версию командой **`dch -i`** для того, чтобы в изменения вписался ваш e-mail. А для того, чтобы ваш открытый ключ попал на сервер, необходимо в настройках Пароли и ключи, настроить соединение с сервером публичных ключей. Для этого, в меню Параметры на закладке необходимо поставить галку Опубликовать ключи.... Теперь можно выбрать ключ и в меню по правой кнопке выбрать Синхронизировать и опубликовать ключи.

## 1.3 Создание ключа с помощью графического интерфейса

Пароли и ключи-Добавить-ключ GPG key.

Созданный ключ необходимо использовать при создании пакетов.

Для этого, в файл ~/.bashrc, или в другой стартовый скрипт, вашего шелла (для zsh ~/.zshrc), нужно вписать переменные

`export DEBEMAIL=ваш@имейл`

На основании e-mail будет искаться ключ в pgp, при подписи пакета. Нужно завершить сеанс и зайти заново, чтобы изменения вступили в силу.

## 2 Дебианизация и сборка паркетов

Необходим архив с исходным кодом программы \*.tar.gz (можно скачать, либо загрузить с репозитория). Далее создайте каталог, поместите и распакуйте там архив. Изменить файлы:

copyright – вписать ФИО студента, email;

control – вписать

- Source (название пакета),
- Section (название раздела в дистрибутиве, к которому относится пакет с исходным кодом),
- Priority (приоритет optional, обычно, назначается новым пакетам, которые не конфликтуют с другими, имеющими приоритет required, important или standard),
- Maintainer
- Build-Depends – установить зависимости (если у пакета есть зависимости, их надо предварительно установить: `sudo apt build-dep имя или dpkg-depcheck -d ./configure`),
- Homepage – адрес ПО
- Package – имя пакета должно совпадать с Source,
- Description- описание,

changelog,

rules

Далее выполнить дебианизацию и сборку пакета.

[https://help.ubuntu.ru/wiki/%D1%81%D0%B1%D0%BE%D1%80%D0%BA%D0%B0\\_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D0%BE%D0%B2%D0%B4%D0%B5%D0%B1%D0%B8%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F\\_%D0%B1%D0%B5%D1%80%D1%91%D1%82%D1%81%D1%8F\\_%D0%BD%D0%B5\\_%D0%B8%D0%B7\\_%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8F\\_%D1%82%D0%B5%D0%BA%D1%83%D1%89%D0%B5%D0%B3%D0%BE\\_%D0%B2%D1%8B%D0%BF%D1%83%D1%81%D0%BA%D0%B0\\_ubuntu](https://help.ubuntu.ru/wiki/%D1%81%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D0%BE%D0%B2%D0%B4%D0%B5%D0%B1%D0%B8%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%B1%D0%B5%D1%80%D1%91%D1%82%D1%81%D1%8F_%D0%BD%D0%B5_%D0%B8%D0%B7_%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8F_%D1%82%D0%B5%D0%BA%D1%83%D1%89%D0%B5%D0%B3%D0%BE_%D0%B2%D1%8B%D0%BF%D1%83%D1%81%D0%BA%D0%B0_ubuntu)

## МЕТОДИКА ВЫПОЛНЕНИЯ

1. Установить make, autoconf, automake, libtool, autotools-dev, dpkg-dev, devscripts, fakeroot.

2. Создать ключ шифрования с помощью системных команд и графического интерфейса, где необходимо выбрать требуемый тип ключа и размер, срок действия, имя, email и описание.
3. Собрать Debian-пакет (из исходных или бинарных файлов), представить преподавателю все исправленные файлы сборки.
4. Подписать пакет ключом.
5. Создать аутентифицированный локальный репозиторий, добавить этот репозиторий к программным источникам и разместить там собранный пакет. Показать возможность обновления из локального репозитория.
6. Добавьте репозиторий в систему и обновить.

## **КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Что такое АРТ, для чего предназначен, что хранит?
2. Что такое Deb-пакет?
3. Что такое сборка из исходников?
4. Что такое сборка из бинарных файлов?
5. Опишите варианты создания ключей шифрования.
6. В чем отличие предлагаемых GNU Privacy Guard типов ключей? Описать.
7. Отличие симметричного и ассиметричного шифров.
8. Что такое бэкпорт и для чего он нужен?

## **ЛАБОРАТОРНАЯ РАБОТА №10 (2 часа)**

### **КОНФИГУРИРОВАНИЕ И КОМПИЛЯЦИЯ ЯДРА LINUX.**

#### **САМОСТОЯТЕЛЬНАЯ РАБОТА**

### **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

В Debian есть несколько очень полезных инструментов, которые существенно упрощают работу по сборке и установке ядер в Ubuntu. Они упрощают процесс компиляции ядра и создания из него пакета *.deb*, который позволяет устанавливать новое ядро точно таким же образом, как и любой другой пакет. Можно создать ядро на одной машине, а затем просто установить пакет на других машинах без перекомпиляции или отладки — это удобно, если вам требуется обновить ряд аналогичных машин.

Чтобы собрать собственное ядро Ubuntu, вам нужно получить исходные тексты ядра и различные инструментальные средства, необходимые для настройки, компиляции и сборки пакета. Необходимо выполнить основные шаги по сборке ядра, применяя документацию ядра.

### **МЕТОДИКА ВЫПОЛНЕНИЯ**

- 1.1 Установить несколько пакетов, которые понадобятся для сборки и настройки ядра.
- 1.2 Получить исходный код ядра.
- 1.3 Распаковать исходный код ядра.
- 1.4 Конфигурировать ядро (с уточнением, знать основные параметры конфигурации и рассказать об измененных преподавателю. Показать, какое оборудование подключено)
- 1.5 Скомпилировать и собрать пакет ядра, в названии должна быть фамилия студента на англ. яз (вместо mykernel)
- 1.6 Инсталлировать собранный пакет ядра.
- 1.7 Перезагрузить и протестировать.
- 1.8 Написать отчет по параметрам ядра.

### **КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Что такое компиляция и для чего применяется компиляция ядра Linux?
2. Назовите основные параметры конфигурации ядра Linux?
3. Где физически располагаются драйверы ядра при его загрузке?

**ПРИЛОЖЕНИЕ А**  
**Образец оформления лабораторной работы**

**ЛАБОРАТОРНАЯ РАБОТА №1**  
**ИНТЕРПРЕТАТОР КОМАНДНОЙ СТРОКИ ОС MS WINDOWS**  
**Внешние и внутренние команды**

Выполнил: Иванов И.И., гр. ВКБ-31

Проверил:

---

*(ф.и.о, подпись, дата)*

**Цель работы** — знакомство с возможностями интерпретатора командной строки и командами MS Windows

**ХОД РАБОТЫ**

1. ...
2. ... и т.д.

**Вывод:**...

**ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. ...
2. ... и т.д.

**ПРИЛОЖЕНИЕ Б**  
**Образец оформления титульного листа Журнала лабораторных работ**



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ДГТУ)**

Факультет Информатика и вычислительная техника

Кафедра «Кибербезопасность информационных систем»

**ЖУРНАЛ ЛАБОРАТОРНЫХ РАБОТ**

по дисциплине \_\_\_\_\_

Обучающийся \_\_\_\_\_  
подпись, дата

Группа \_\_\_\_\_

Направление \_\_\_\_\_

Профиль \_\_\_\_\_

Проверил: \_\_\_\_\_ кт.н., доцент Язвинская Н.Н.  
отметка, подпись, дата



Ростов-на-Дону

202\_\_