*Модуль 1 (2 пары)*

**Знакомство с Linux**

1. Введение.
   1. **История Linux**

Что такое Linux или GNU/Linux. Общее название unix-подобных ОС на основе ядра Linux, собранных для него библиотек и системных программ, разработанных в рамках проекта GNU. Richard Stallman and GNU.

Linux = библиотеки и утилиты GNU + ядро, написаное Торвальдсом, или kernel и выпущенное в 1991г + различное прикладное ПО.

У Linux нет своего рода официальной комплектации, вместо этого существуют дистрибутивы. Самые популярные из них Ubuntu, Mint, Debian GNU/Linux, Fedora, openSUSE, Gentoo, CentOS.

* 1. **Понятие лицензии**. **Свободное программное обеспечение**(GNU)

Linux распространяется на условия GPL (general public license), то есть свободно.

Лицензия на ПО – это правовой «инструмент», который определяет использование и распространение ПО, защищенного авторским правом.

Существует ПО:

**коммерческое** (несвободное). Пользователь использует одну или несколько копий ПО, но правообладателем ПО по-прежнему является его издатель.

**и свободное или открытое**. В данном случае пользователь является полноценным правообладателем подобного ПО, имеет право на изменение/доработку этого ПО и распространять его (даже за деньги) с предоставлением исходного кода, но ТОЛЬКО под той же свободной лицензией.

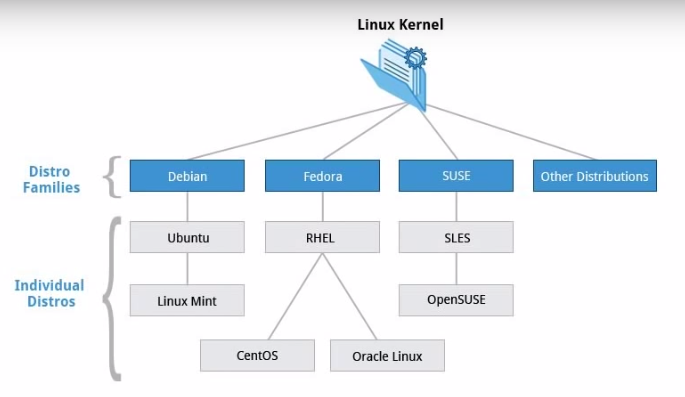
**GPL дает права**:

1. Свободно использовать программу с любой целью
2. Модификацию программы (есть доступ к сорцам)
3. Свободно публиковать модификации вместе с сорцами.
4. Свободно распространять её копии (скачал, расшарил)
   1. **Понятие дистрибутива Linux.**

Linux не имеет одной комплектации. Вместо это Linux поставляется в виде дистрибутивов, в которых ядро Linux объединено вместе с библиотеками и утилитами GNU и другими уже прикладными программами, делающими его полноценной ОС.

<https://en.wikipedia.org/wiki/Linux_distribution>

Все дистрибутивы можно разделить на семейства исходя из родительской линии разработки



Отмечу самые популярные:

* собственно, Debian и его дочерний Ubuntu, Kali-Linux.
* Fedora, и его дочерние RHEL, Centos, Oracle Enterprise, trixBox, Elastix (последние 2 для ip-телефонии)
* SlackWare -> openSUSE, SLES.

**Не всегда Linux является бесплатным**. Некоторые дистрибутивы могут иметь платную поддержку (Commercial Support and Services) – Oracle, RHEL, SLES.

Но большинство имеет и без того **огромную бесплатную базу знаний** и сообщество, готовое прийти на помощь (CentOS, Fedora, Ubuntu)

1. **Ядро Linux**

Очередность загрузки ОС (более развернуто <https://www.thegeekstuff.com/2011/02/linux-boot-process/>):

1. powerON
2. POST – инициализация оборудования и самопроверка его работоспособности.
3. Далее BIOS передает управление загрузчику (boot loader). Загрузка ОС может быть осуществлена как с локальных носителей (hdd, cd), так и с удаленных (по сети). Для этого BIOS должен уметь находить место загрузчика. В случае загрузки с локальных носителей, местонахождение загрузчика на диске указано в таблице разделов загрузочного носителя, хранящихся в 2 форматах: MBR или более новый GPT.

В случае с MBR таблица разделов хранится в первых 512 байтах загрузочного носителя.

В случае с GPT таблица загрузчик хранится на специальном служебном разделе и специальной файловой системой.

*Основные отличие MBR и GPT (количество разделов 32 и размер адресуемого пространства 2Тб в MBR) вы уже должны знать из курса EITE.*

GRUB, ISOlinux – используемые загрузчики в Linux.

1. В случае наличия 2 и более установленных образов ядра операционной системы загрузчик покажет окно выбора.
2. В загрузчике указаны пути к ядру и initRD
3. После загрузчик монтирует временную initramfs-файловую систему, чтобы найти местонахождение файлов образа ядра и специального initial RAM-диска, содержащего необходимые драйверы для инициализации дисков и другого оборудования.
4. Загрузчик выполняет загрузку initial RAM disk (initrd) - наиболее критичные драйвера необходимые для загрузки ядра и после загрузку ядра ОС.
5. Ядро осуществляет монтирование корневой файловой системы, выгружает из оперативной памяти initramfs, и загружает в нее драйверы всех устройств, короче говоря стартует ОС, после…
6. Запускает ядро корневой первый процесс ОС /sbin/init – процесс запускающий все основные службы пользовательского окружения (У данного процесса PID=1). И уже **init**  загружает всё остальное, в том числе **text-mode login** – приглашение пользователя залогиниться в систему в командной строке, отображаемое в терминале. *ps -ef | grep init*

Ядро – «клей» между железом и приложениями/программами. Различные версии ядра, как и самые последние, можно найти на <https://www.kernel.org/>

Текущая версия ядра

***$ uname -a***

Ядро решает следующие задачи:

1) обеспечивает абстракцию оборудования от приложений.

2) выделение **памяти** запущенным процессам.

3) обеспечение доступа запущенным приложениям к набору **системных вызовов**.

**Процессы**. Ядро запускает, останавливает, перезапускает процессы.

Ядро определяет, каким процессам и когда разрешен доступ к процессору (процессорному времени). Многозадачность – выделение тайм-слота (конечного значения процессорного времени) для каждого процесса. Передача доступа к процессору между процессами – *переключение контекста* (context switching), этим и занимается ядро. То есть ядро работает между работой процессов, т.е. в промежутки работы процессов.

**Память.** Распределение памяти между процессами.

* Есть участок памяти, доступ к которой имеет только ядро, остальные процессы (пользовательские) получить доступ к этой памяти не могут.
* Каждый процесс нуждается в своей области памяти
* Невозможно пересечение между областями памяти разных процессов.
* Все процессы могут использовать память совместно.
* Возможно наличие ReadOnly-участков памяти.
* Что делать процессам, когда памяти недостаточно (обращение к swap).

Доступ к памяти ядром реализуется при помощи MMU (memory management unit) и механизма «виртуальной памяти». Процесс считает, будто у него есть вся память компьютера. MMU, применяя карту адресов памяти, отводит в виртуальную память для отдельного процесса столько физической памяти сколько определено ядром.

**Драйверы.** Ядро является «интерфейсом» между железом и процессами. Все аппаратные ресурсы управляются ядром. Как правило, напрямую любой пользовательский процесс не может получить доступ к оборудованию, делается это для того, условно «пользователь не сломал» устройство. Подобный запрос доступа к оборудованию носит название **системного вызова.**

**Системные вызовы.** С точки зрения программиста, системный вызов обычно выглядит как вызов [подпрограммы](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) или функции из [системной библиотеки](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0) (из ядра). Категории системных вызовов:

А) **Управление процессами** – загрузить новый процесс в память, запустить, приостановить, завершить, выделить или освободить память и пр.

Б) **Работа с файлами** – создание, удаление, открытие, закрытие файла, установка его атрибутов.

В) **Управление устройствами**, включение выключение, к примеру сетевого интерфейса.

Г) **Работа с информацией**

Д) **Коммуникации** – создание новых соединений, их удаление, посылка и прием данных с удаленных узлов, добавление удаленных устройств по сети.

Системный вызов – это набор функций общих для нескольких процессов, выполняются эти вызовы ядром по запросу пользовательского процесса. Другим словом, системный вызов – это инструмент взаимодействия приложения с «внешним миром».

1. **Установка Ubuntu.**

Установка возможна с любых типов носителей, так и из сети. Ставим на ВМ в virtualBox.

Разбивка диска и точка монтирования (посмотреть partition table *sudo fdisk –*l):

/ (ext4 – работает быстрее)

swap = physical memory

home

1. **Архитектура и окружение Ubuntu.**

***$ uname –a***  - информация о системе (разрядность, версия ОС, ядра и пр.)

GUI environment GNOME/KDE (на винду больше похож)

1. **Основы работы в системе. Базовые операции с системой.**

Перезагрузка, выключение, гибернация.

*Модуль 2 (2 пары)*

**Использование Ubuntu**

1. **Графический интерфейс пользователя(GUI).**

**GNOME Desktop Environment**. Является окружением по умолчанию у многих дистрибутивов RHEL, Fedora, CentOS, SUSE Linux Enterprise, Debian**.**

Другие: **KDE, Xfce, LXE, Unity.**

Desktop Environment – это всего лишь служба (надстройка) запущенная поверх ядра. Когда мы устанавливаем GNOME, стартует процесс ***X display manager***, который управляет GUI, процессом залогинивания в графическую оболочку.

Иные системы графического окружения:

gdm (Gnome)

lightdm (используется в UBUNTU)

Kdm (kde)

**Разлогиниться. CTRL-ALT-L (или из трея в верхнем правом углу) –** заблокировать рабочий стол. Все горячие клавиши можно посмотреть здесь:

System Settings → Keyboard → Shortcuts.

**Reboot/Restart/Shut Down**. Иногда нужно перезагрузиться, например, после обновления **kernel** (ядра). Процедуры выключения и перезагрузки осуществляет процесс **init**.

**Suspend (sleep) mode.** Так же из трея.

**Запуск программы**. Клик по «Пуску». Навигация по категориям. Как видно, Ubuntu уже из коробки – готовая ОС для дома.

**Приложения по дефолту**. Можно изменить мозиллу на другой браузер.

**Папки по дефолту**. Каждый пользователь имеет свою /home/username каталог, где username – имя юзера. В этой папке создаются каталоги по умолчанию:

Documents

Downloads

Desktop

Pictures

Trash

Computer

Переключится между режимами вида для просмотра большего количества атрибутов у файлов и каталогов можно использоваться CTRL-1, CTRL-2, CTRL-3 в коне File Manager.

CTRL+H – показать скрытые и системные файлы.

**Поиск**. Nautilus – тулза для поиска файлов/каталогов. ALT-F2 – откроется file manager. **CTRL-F** – открывается поиск по file manager’у.

**Редактирование файла.** Из пуска можно запустить **gedit**.

**Удалить файл.** Deleting a file in **Nautilus** will automatically move the deleted files to the .local/share/Trash/files/ directory (a trash can of sorts) under the user's **HOME** directory. **Shift-delete –** удалить без удаления в корзину. Никогда не удаляйте свою /home/username директорию. Это сотрет все окружение и настройки пользователя.

Доступ к системным настройкам (дата, время, разрешение дисплея, менеджеру сетевых подключений и т.п.)

**Установка и удаление программ в GUI**

Пакет – кусок системы, программа. К примеру браузер, компилятор какого-то языка программирования. Пакеты часто зависят друг от друга. То есть не могут быть запущены/установлены, пока не установлены пакеты, от которых зависит текущий. К примеру, Mozilla FF не получится установить, если в системе нет библиотек для работы с SSL/TLS шифрованием.

Менеджеры пакетов в Debian-системах:

dpkg – низкоуровневый менеджер, умеет устанавливать/удалять/собирать пакеты. Не умеет автоматически устанавливать зависимости между пакетами.

apt – высокоуровневый менеджер – понимает зависимости.

В других дистрибутивах другой набор менеджеров пакетов rpm/yum (Fedora, CentOS, RHEL); rpm/yast (openSUSE).

1. **Интерфейс командной строки(CLI)**

**Достоинства CLI**: меньше overhead, любая задача может быть выполнена в CLI, можно создавать сценарии и серии сценариев, есть возможно удаленного доступа через CLI к любой машине, подключенной к сети, любую GUI-application можно стартануть из CLI.

**Команда в CLI состоит из**:

<Программа> <параметры> <аргументы>.

Программа – программа, которую мы хотим запустить.

Опции – могут изменять вывод команды или добавлять некий функционал комманде. Опции начинаются с – или с --

Аргумент – над чем выполняется Программа.

Не у каждой команды есть опции и аргументы.

**$ sudo** <doing something> - выполнить команду от имени суперпользователя (администратора).

Чтобы работать в CLI необходимо запустить терминальный эмулятор. Различные терминальные эмуляторы:

gnome-terminal

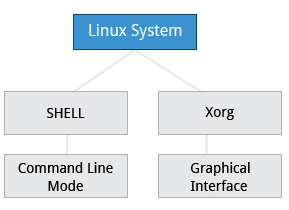
xterm

rxvt

konsole

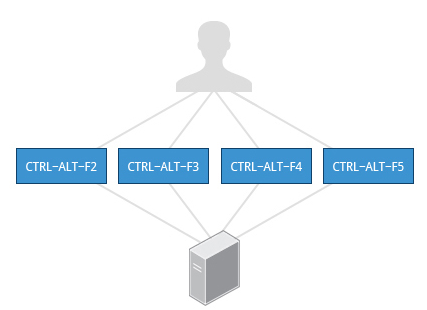
terminator

**X Window System** (иксы) – GUI, который можно при желании выключить, дабы не расходовать дорогостоящие ресурсы (память, CPU).



Virtual Terminals

**Virtual Terminals (VT)** are **console** sessions that use the entire display and keyboard outside of a graphical environment. Such terminals are considered "virtual" because although there can be multiple active terminals, only one terminal remains visible at a time. A VT is not quite the same as a command line terminal window; you can have many of those visible at once on a graphical desktop.



One virtual terminal (usually number one or seven) is reserved for the graphical environment, and text logins are enabled on the unused VTs. **Ubuntu** uses VT 7, but **CentOS/RHEL** and **openSUSE** use VT 1 for the graphical display.

An example of a situation where using the VTs is helpful when you run into problems with the graphical desktop. In this situation, you can switch to one of the text VTs and troubleshoot.

To switch between the VTs, press **CTRL-ALT-corresponding function key** for the VT. For example, press **CTRL-ALT-F6** for VT 6. (Actually you only have to press **ALT-F6** key combination if you are ina VT not running **X** and want to switch to another VT.)

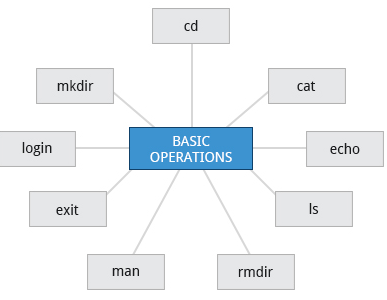
**Turning off the Graphical Desktop**

Linux distributions can start and stop the graphical desktop in various ways. For **Debian**-based systems, the **Desktop Manager** runs as a service which can be simply stopped. For RPM-based systems, the **Desktop** **Manager** is run directly by **init** when set to run level 5; switching to a different runlevel stops the desktop.

Use the ***sudo service gdm stop*** or ***sudo service lightdm/stop***  stop commands, to stop the graphical user interface in **Debian**-based systems. On **RPM**-based systems typing ***sudo telinit 3*** may have the same effect of killing the GUI.

**Базовые операции в системе.**

How to log in and log out from the system, restart or shutdown the system, locate applications, access directories, identify the absolute and relative paths, and explore the filesystem.



**Logging In and Out**

Once your session is started (either by logging in to a text terminal or via a graphical terminal program) you can also connect and log in to remote systems via the **Secure** **Shell (SSH)** utility. For example, by typing ssh [username@remote-server.com](mailto:username@remote-server.com)

**Rebooting and Shutting Down**

The **halt** and **poweroff**commands issue

***$*** ***shutdown -h*** to halt the system;

 reboot issues ***$*** ***shutdown -r*** and causes the machine to reboot instead of just shutting down. Both rebooting and shutting down from the command line requires superuser (root) access.

**Locating Applications**

**Whereis (путь к манам и самой утилите) или which (путь к утилите)**

В зависимости от дистрибутива, типа приложения/службы, установленное приложение может жить в

**/bin**

**/usr/bin**

**/sbin**

**/usr/sbin**

**/opt**

директориях.

Чтобы найти где установлено то или иное приложения (к примеру, изменить конфигурацию приложения) используется утилита **which**:

***$ which diff***Другая утилита ***whereis*** делает тоже самое, но ищет более широко **which** (не по точному совпадению):

***$ whereis diff***

Можно через пробел найти несколько программ

***$ whereis ping lvm sudo***

**Абсолютный и относительный путь.**

**Абсолютный** – начинается с /

**Относительный** – начинается с текущего каталога.

К примеру,

***$ cd /home/alexandr/Desktop –*** абсолютный путь

***$ cd .***.**/../etc/var –** относительный пусть

*. (present directory), .. (parent directory) and ~ (your home directory)*

*$ ls ls –l ls –a ls –d*

$ !! – повтор последней команды, можно курсором «вверх»

**Accessing Directories**

*$ cd*

*$ cd..*

*$ cd –*

*$ cd ~*

1. **Основные понятия о файлах, типы файлов в Linux.**

Существует 2 типа файлов – регулярные (обычные) и конфигурационные (с точкой)

***$ ls –a***  - покажет скрытые файлы, имя которых начинается с точки .

|  |  |
| --- | --- |
| cat | Посмотреть не очень длинный файл. Если указать несколько файлов в качестве аргументов, то вывод команды будет «сцеплен» |
| tac | То же, что и **cat**, только файл читается с конца |
| less | Посмотреть длинный файл постранично |
| tail | Посмотреть конец файла (последние 10 строк). Ключ **–n** можно указать количество строк ***$ less <anyfile> -n 15 –*** посмотреть последние 15 строк файла. |
| head | Посмотреть первые 10 строк. Ключ **–n** – изменить количество строк |

Можно создать пустой файл:

***$ touch <newfile>***

Команда ***touch*** всего лишь «трогает» файл, меняет в нем атрибут даты создания/модификации.

***$ file anyfile.txt*** – узнать содержимое файла, его тип.

cp, mv, mkdir, rm, rm –rf.

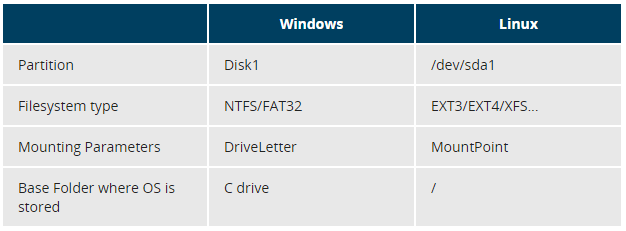
1. **Иерархия файловой системы Linux.**

Файловая система – метод хранения и организации коллекций случайных данных в удобной форме. Аналогия с холодильником и полками для разных продуктов.

* Традиционные disk filesystems: ext2, ext3, ext4, XFS, Btrfs, JFS, NTFS, etc.
* Flash storage filesystems: ubifs, JFFS2, YAFFS, etc.
* Database filesystems
* Специализированного назначения filesystems: procfs, sysfs, tmpfs, debugfs, etc.

Раздел (partition) – логическая часть диска, тогда как файловая система – метод организации хранения/поиска файлов на диске (чаще всего на разделе).

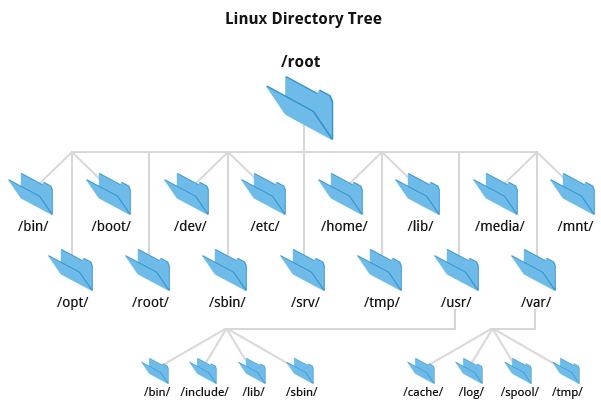
Различия в файловых система Linux и Windows:



Linux-based ОС хранят важные файлы в соответствии со стандартом FHS (filesystem hierarchy standard). Поэтому от дистрибутива к дистрибутиву переучивать местоположение основных файлов и конфигураций не нужно.

Linux использует forward-slash / для разделения пути и не имеет такого понятия как «буква диска». Новые диски маунтятся (монтируются) как каталоги в единую файловую систему, чаще всего в каталог /media.

Имена в файловой системе Linux **регистроЗависимые** (case-sensetive).



**Работа с файловой системой и разделами.**

Инструменты для работы с разделами и устройствами:

* **parted**
* gparted (parted но в графике)
* **fdisk – традиционный инструмент в linux.**
* gdisk (с графикой)

***$ sudo parted –l –*** просмотреть разделы

***$ sudo fdisk /dev/sdb***

***(parted) n –*** следовать указаниям мастера. Создать extended раздел

Далее создать ФС на разделе.

***$ mkfs –t ext4 /dev/sdb***

Осталось подмонтировать созданную ФС в иерархию каталогов.

***$ mkdir /home/extraspace –*** создадим каталог, куда подмонтируем наш раздел с ФС

***$ sudo mount –t ext4 /dev/sdb /home/extraspace***

Скопируем туда что-нибудь

***$ sudo cp <anyfile> /home/extraspace***

Отмонтируем фс, то есть удалим наш раздел с созданной на нем ФС из иерархии каталогов

***$ sudo umount /home/extraspace***

И вернем назад…

***$ sudo mount /dev/sdb /home/extraspace***

… ***–t ext4*** - linux умный сам догадался, какая файловая система создана на разделе.

Пропишем автоматическое подмонтирование при загрузке

***$ sudo nano /etc/fstab***

Добавим строчку в конфигурацию таблицы ФС

***/dev/sdb /home/extraspace ext4 defaults 0 2***

… сохранимся, перезагрузимся, вуаля.

**Модуль 3 Командная оболочка**

1. **Терминал. Команда.**

**Действие в CLI состоит из**:

команда опции аргументы.

Команда – программа, которую мы хотим запустить.

Опции – могут изменять вывод команды или добавлять некий функционал комманде. Опции начинаются с – или с --

Аргумент – над чем выполняется команда.

Не у каждой команды есть опции и аргументы.

1. **Переменные окружение (опущено)**
2. **Понятие «Суперпользователя»**

Root – полноправный админ. Обычный пользователь может использовать***su*** или ***sudo*** для получения временных прав администратора. Эти методы разные.

***$ su*** – при введении этой команды пользователь должен ввести пароль от root-пользвотеля. Простому пользователю никогда не нужно знать пароль от root. Кроме того, введя ***su*** пользователь как бы «логинится в root», после чего он может делать что угодно с системой не вводя пароль от root. Действия команды имеют **ограниченное журналирование**

***$ sudo*** – пользователь должен подтвердить свои намерения, введя свой пароль (не root). Этот метод более безопасен поскольку любое действие, требующее получения root-привилегий, всякий раз потребует подтверждение намерений, запрашивая у пользователя его пароль. В данном случае все действия **журналируются** **детально**.

***$ sudo.*** Пользователи, авторизованные на доступ к ***sudo,*** должны быть внесены в конфигурацию, находящуюся в ***/etc/sudoers /etc/sudoers.d***. Всякий раз, когда пользователь пытается что-либо выполнить из-под sudo, система обращается к файлу sudoers, чтобы убедиться в наличии прав на доступ к команде sudo данному пользователю. Редактирование файла ***sudoers*** осуществляется при помощи утилиты

***$ sudo visudo*** – эта утилита проверяет синтаксис измененных прав на доступ к sudo.

***# User privilege specification***

***root ALL=(ALL:ALL) ALL***

***anyuser ALL=(ALL:ALL) ALL***

После вышестоящих изменений, отмеченных желтым, пользователь «anyuser» получает права запускать что-то из под ***sudo***.

Все попытки аутентификаций, а также действия из-под ***sudo***  хранятся в ***/var/log/auth.log.***

Демонстрация на примере  ***$ sudo fdisk –l***  и без sudo.

В других дистрибутивах журналы хранятся в ***/var/log/messages*** или  ***/var/log/secure***

Данный журнал хранит детальную информацию о:

- вызывающий пользователь

- из какого терминала была запущена сессия

- рабочий каталог

- из-под чьих привилегий была выполнена команда

- выполняемая команда с аргументами.

1. **Использование руководств и информационных страниц (man, info)**

***$ man ls***

***$ man fdisk***

***$ ifconfig –h***

***$ ifconfig --help***

1. **Управление пользователями и группами**

Ядро позволяет авторизованным пользователям получить доступа к файлам и папкам. Каждому пользователю ставится в соответствие пара username == уникальный UID (userID), данные об пользователе хранятся в отдельной базе данных. Информация о пользователях хранится в /etc/passwd:

***alexandr:x:1000:1000:Alexandr Matskevich,,,:/home/alexandr:/bin/bash***

***sshd:x:121:65534::/var/run/sshd:/usr/sbin/nologin***

***newuser:x:1001:1001:newuser,,,:/home/newuser:/bin/bash***

*Username* – логин пользователя

*Password* – пароль никогда не хранится в открытом виде. Вместо него можно увидеть символ **х. \* - запрещен вход в систему (юзер, из-под которого запускается фоновая служба – демон), :: на месте пароля – пароль не установлен.**

UID – user id. Ядро использует именно UID при обращении к правам пользователя, а не к username.

UID 0 is reserved for root user. UID's ranging from 1-99 are reserved for other predefined accounts. UID's ranging from 100-999 are reserved for system accounts and groups (except for RHEL, which reserves only up to 499). Normal users have UID's of 1000 or greater, except on RHEL where they start at 500

*GID* - group id. The primary Group ID (GID); Group Identification Number stored in the /etc/group file. По умолчанию GID основной группы локального юзера совпадает с UID.

*User info* – доп. инфа такая, как фулнейм, телефон, кабинет…

*Home directory -*  абсолютный путь к домашнему каталогу */home/username*

*Shell –* абсолютный путь к оболочке по умолчанию */bin/bash* или */bin/sh*

***$ cat /etc/passwd***  - посмотреть всех локальных пользователей в системе.

Для распределения обязанностей, задач и прав есть несколько типов учетных записей:

- root

- system

- normal

- network

*$ last* – посмотреть последних залогиненых локальных пользователей. Удалить ненужных, неактивных, если в этом есть необходимость. В хелпах можно найти ключи, чтобы просмотреть активность пользователей с одной даты до другой.

***Root –*** самый главный пользователь в системе. Он имеет все права: добавить пользователя, удалить, изменить пароль, просмотр журнала/логов, установка софта. Из-под рута промпт обычно заканчивается символом **#.**

Привилегии root-пользователя: управление пользователями, программными пакетами, изменение/удаление системных файлов, перезагрузка.

Обычным пользователям также могут быть разрешены некоторые из этих привилегий (установить пакет, обновить некоторые настройки), однако, рут-права необходимы для административных задач в системе, к примеру, перезапуск служб/процессов или их остановка, ручная установка deb-пакетов.

**Добавить пользователя, установить пароль, добавить в группу**

***$ Sudo adduser test - создать***

***$ Sudo userdel test - удалить***

***$ Sudo passwd test*** (ввести пароль, повторить пароль)

После этого в файл ***/etc/passwd*** добавляется строчка вида

***test:x:502:502::/home/test:/bin/bash***

***или***

***test:x:502:502::/home/test:/bin/sh***

Здесь ID юзера (502) и его shell (bash, sh)

***$ sudo usermod turkey -l newturkey –*** *переименовать юзернейм* ***turkey*** *в* ***newturkey****. При этом домашний каталог будет старым.*

Посмотреть имя юзера, под которым я сейчас залогинен:

***$ who***

***$ who –a***

***$ whoami***

***Группы***

***$ cat /etc/group –*** *посмотреть список всех существующих групп.*

***$ groups –*** *посмотреть список групп, в которых состоит текущий пользователь*

***$ sudo groupadd anewgroup*** Добавить группу «anewgroup»

***$ sudo groupdel anewgroup*** *– удалить группу “anewgroup”*

***$ groups turkey*** *– в какой группе находится пользователь turkey?*

***$ sudo usermod turkey*** ***-G anewgroup*** *– добавить юзера “turkey” в группу «anewgroup”*

Removing a user from the group is a somewhat trickier. The**-G** option to **usermod** must give a complete list of groups. Thus if you do:

***$ sudo usermod turkey -G turkey*** – «*удалить» пользователя turkey из группы «anewgroup” и оставить только в группе «turkey”. То есть перечисляем список групп, в которых должен состоять юзернейм* ***turkey****.*

***$ su “anyuser”*** – переключится в shell юзером «**anyuser**”. Нужно ввести пароль от anyuser. Команда некошерная, потому как можно натворить из shell’a root’а.

***$ sudo <command>*** – SupersUser DO command выполнить команду от имени root. После выполнения осуществляется возврат в свой shell.

Где хранятся юзерские пароли? А нигде. Хэши от паролей хранятся в файле ***/etc/shadow***.

1. **Полезные возможности (автодополнение, история команд, псевдонимы(alias))**

**История введенных команд**

***$ history***

Или посмотреть в файл, где вся история сохраняется:

***$ cat ~/.bash\_history***

Параметры этого файла (хранятся в виде переменных в файле ***set***):

alexandr@ubhome:~$ echo $HISTSIZE

1000

alexandr@ubhome:~$ echo $HISTFILESIZE

2000

alexandr@ubhome:~$ echo $HISTFILE

/home/alexandr/.bash\_history

**Алиасы**

Алиасы – это «псевдокоманды» или просто свои, кастомные команды, допиленные и обозванные по другому, т.е. псевдонимы

Хранятся в файле ~/.bashrc

Напишем свой alias:

***$ alias l2=’ls –l /etc’***

***$ alias*** *– проверить записался ли наш alias*

***$ letc*** *– Запустить свежий алиас*

***$ unalias letc*** *– удалить алиас «letc»*

Когда удобно использовать алиасы?

1) Короче команда

2) Предопределить новое исполнение стандартной программы по умолчанию.

3) Опечатки основных команд

***$ alias pdw=pwd***

4) Сделать запуск программы интерактивным

***$ rm anyfile –*** *просто удаляет файл*

***$ alias rm=’rm –i’***

***$ rm scl.sh***

***rm: remove regular file 'scl.sh'? yes***

5) … может что-то еще

**Основные консольные команды Linux**

1. **Основные операции при работе с файлами и каталогами**
   1. управление файлами

**Создать файл, удалить, переименовать директорию**

*$ touch anyfile.txt – сздать файл*

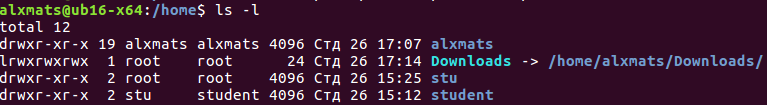
*$ file anyfile.txt – узнать что за файл.*

cp, mv, mkdir, rm, rm –rf.

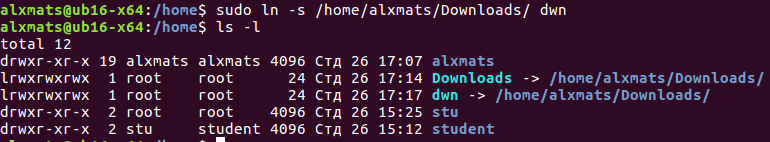
* 1. управление каталогами
  2. управление символьными ссылками

***$ ln <target path> <local link name>***

***$ ln –s /home/alxmats/Downloads – создается символическая ссылка (ярлык) на папку Downloads.***



Её можно обозвать по-другому:



1. **Просмотр содержимого файлов (less, more)**
2. **Поиск файлов (find, locate) и Маски для поиска**

Есть 2 инструмента: locate, find.

***$ locate zip***

***$ locate traceroute***

**Locate** файл/директорию – этот инструмент ищет в не в файловой систем а в**.** Если новый файл не был проиндексирован, то найти его не получится с помощью ***locate*.**

Нужно обновить индекс

***$ sudo updated -***  обновили индекс

***$ locate <anyfile>***  *-* файл ищется в уже обновленном индексе.

***Find* –** более мощный инструмент для поиска. С гибкими фильтрами.

***$ find –type d*** – папку

***$ find –type f*** – файл

***$ find –type f*** –name ttt.txt – найти файл с именем ttt.txt

|  |  |
| --- | --- |
| **Wildcard** | **Result** |
| ? | Matches any single character |
| \* | Matches any string of characters |
| [set] | Matches any character in the set of characters, for example [adf] will match any of "a", "d", or "f" |
| [!set] | Matches any character not in the set of characters |

***$ find –type f –iname \*.php*** – найти все php-файлы с регистронезависимым именем.

***$ find /etc/*** - найти по пути

***$ find –type f –perm 0664*** – найти файл с разрешениями chmod=644 (rw,w,w)

***$ find –type f –not –name \*.txt*** – найти все файлы, кроме \*.txt

***$ find –type f –size +1k*** – найти файлы более 1Кб (+1М – более 1Мб)

***$ find –type f /etc –iname \*.conf –maxdepth 2*** – найти все conf-файлы не глубже указанного пути (глубина расположения файла).

1. **Работа с архивами**

К примеру есть папка **dir** с проектом:

***alexandr@ubnt32:~/Desktop$ ls -l***

***total 4***

***drwxrwxr-x 2 alexandr alexandr 4096 Лют 13 12:13 dir***

И в этой папке куча файлов вашего проекта:

***alexandr@ubnt32:~/Desktop$ ls -l dir/***

***total 16***

***-rw-rw-r-- 1 alexandr alexandr 30 Лют 13 12:10 newfile1***

***-rw-rw-r-- 1 alexandr alexandr 30 Лют 13 12:10 newfile2***

***-rw-rw-r-- 1 alexandr alexandr 30 Лют 13 12:10 newfile3***

***-rw-rw-r-- 1 alexandr alexandr 30 Лют 13 12:10 newfile4***

Появилась необходимость сделать бэкап, или отправить по почте/отдать на сервер/скинуть на флеш/и тому подобно. Придется делать эту операцию с каждым файлом. Куда удобнее все это собрать в архив, а еще и сжать. В результате будет 1 файл, вместо четырех:

***alexandr@ubnt32:~/Desktop$ tar zcvf mynewarchieve.tar.gz dir/***

***dir/***

***dir/newfile3***

***dir/newfile1***

***dir/newfile2***

***dir/newfile4***

***alexandr@ubnt32:~/Desktop$ ls -l***

***total 8***

***drwxrwxr-x 2 alexandr alexandr 4096 Лют 13 12:13 dir***

***-rw-rw-r-- 1 alexandr alexandr 231 Лют 13 12:18 mynewarchieve.tar.gz***

Команда ***tar zcvf mynewarchieve.tar.gz dir/*** создает архив из папки dir (которая, к слову, находится в текущем каталоге, откуда дается команда) и сжимает его, в итоге получаем 1 файл ***mynewarchieve.tar.gz***

Чтобы распаковать этот архив делаем следующее:

- создаем новую папку

***alexandr@ubnt32:~/Desktop$ mkdir dir2***

- переходим в нее

***alexandr@ubnt32:~/Desktop$ cd dir2***

- распаковываем в нее содержимое архива:

***alexandr@ubnt32:~/Desktop/dir2$ tar xvf ../mynewarchieve.tar.gz***

В это команде мы говорим «перейти на каталог вверх, найти там архив с именем ***mynewarchieve.tar.gz*** и распаковать его в текущий каталог, из которого дается эта команда (Desktop/dir2)

1. **Изменение значений прав доступа к файлам(chmod)**

**Разграничение прав. Permission and ownership.**

**SUID** (Set owner User ID upon execution) – запуск программы/службы от имени владельца. Аналог «run as» в Windows. SUID

Что может делать обычный пользователя (non-root account):

- запустить клиент какого-либо сетевого приложения/службы (ssh, ftp)

- использовать оборудование (принтер)

- операции над файлами, на доступ к которым есть соответствующие привилегии.

- запустить SUID-application.

3 уровня: чтение, запись(удаление), запуск. Или RWX (Read-Write-eXecute)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Decimal notation |
| 4 | 2 | 1 |  |
| r | w | x |  |
| 1 | 1 | 1 | 7 |
| 1 | 1 | 0 | 6 |
| … |  |  |  |
| 0 | 0 | 0 | 0 |

Таблица для u (user/owner), g (группы), o (other, остальные) своя.

***$ ls –l*** – посмотреть права на каждую папку и файл (не считая скрытых файлов)

***сhown*** – сменить владельца:группу принадлежности файла

***$ chown testuser:testgroup file.txt***

***chmod*** – сменить разрешения

***$ chmod 646 file.txt***

***$ chmod uo+x,g-w file.txt*** – uo(владелец и иные юзеры) – добавить право на выполнение, группе – убрать права на запись в файл file.txt

***$ chmod 646 dir/***  - изменить права доступа на 646 для ВСЕГО содержимого внутри каталога ***dir.***

***$ chgrp ANYNEWGROUP /home/alexander*** – изменить принадлежность папки группе ANYNEWGROUP.

*Модуль 5*

**Обработка текста. Сценарии**

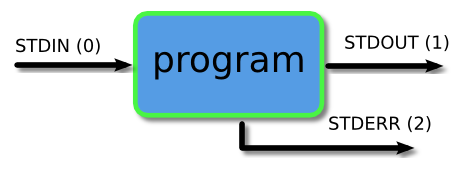
1. Перенаправление ввода/вывода. Конвейер команд

При выполнении команды используются три файловых потока (дескриптора):

**Stdin** – стандартный ввод (дескриптор №0), обычно это ввод с клавиатуры, но можно и из файла.

**Stdout** – станадртный вывод (дескриптор №1) – окно терминала.

**Stderr** – вывод ошибки (дескриптор №2) – окно терминала – часто перенаправляется в файл журнала (лог).



**Перенаправление**.

***$ do\_something < input-file*** перенаправить ввод из файла

***$ do\_something > output-file*** перенаправить вывод в файл

Если нужно перенаправить вывод потока ошибок, то можно указать перенаправление номера дескриптора:

***$ do\_something 2> error-file*** (№2 – дескриптор потока ошибок)

Если необходимо перенаправить и вывод и ошибки в один файл (это более наглядно) то можно направить оба дескриптора в одно место.

***$ do\_something > all-output-file 2>&1***

Можно указать вместо ***>*** знак ***>>*** , что означает буквально «добавить» вывод команды к существующем файлу, то есть сделать ***prepend.***

**Конвейер**

Философия Linux подразумевает, что, выполняя несколько простых коротких программ параллельно можно получить большой вывод в качестве результата. Это лучше, чем иметь 1 большую программу со множеством функций/фич/обвесов.

Такая запись:

***$ command1 | command2 | command3***

… предполагает, что вывод команды 1 станет вводом для команды 2 и т.д. Вывод команды 3 – терминал.

Примеры piping

<http://ryanstutorials.net/linuxtutorial/piping.php>

1. **Регулярные выражения(grep)**

Grep – предназначен для поиска текста, более тонкого поиска и его фильтрации в дескрипторы (ввод, вывод)

*Модуль 6*

**Управление пакетами. Удаленный доступ**

1. **Менеджеры пакетов (dpkg, apt, aptitude ). Управление пакетами**

Установка программ из репозитория и из локального пакета.

$ apt-get – install из репозитория

$ dpkg – install local .deb-пакеты

$ sudo dpkg –r – деинсталлировать

$ sudo dpkg –i - инсталлировать

Список репозиториев */etc/apt/sources.list*

1. **Удаленный доступ**
   1. **Удаленный доступ к терминалу по протоколу SSH**

SSH – secure shell – протокол для шифрованной связи с удаленным устройством через сеть посредством интерфейса CLI. На удаленном устройстве должен работать SSH-server. Чтобы подключится к удаленному ssh-серверу на подключающейся машине должен быть установлен ssh-клиент.

***$ ssh 192.168.1.5***

***Или***

***$ ssh anycomputer.net***

После подключения мы получаем доступ к удаленной машине, команды, выполненные в терминале, теперь выполняются не на локальной машине, а не удаленной.

Выкинуть пользователя:

***$who –u***

***$kill “pid”***

1. **PHP-server**

***Sudo apg-get install php.***

As I understand , You want to start php built-in server so you can easily do that:

cd /path/to/your/app – путь к файлу index.html

php -S localhost:8000

1. **Install ATOM.io**

**ATOM** нет в официальном репозитории. Вместо этого он лежит в Персональные архивы пакетов (PPA).

Cat /etc/apt/sources.list – здесь только список официальных репозиториев программ.

**Sudo add-apt-rep ppa:webupd8team/atom** – добавим ссылку на неофициальный репозиторий софта для Ubuntu

**Sudo apt-get update –** обновим список репозиториев.

**Sudo apt-get install atom** – установим atom.