LINUX =

what OS is

basics

utilities and progs

all Linux comands

what Linux is

Linux architecture

main GNU utilities

users

groups

permissions

shell and terminal

bash scripting

processes

logs  
memory info

TTY

system variables

file system

file types

file management

inod; hard and symbolic links

filesystem hierarchy standard

searching files

/dev

**LINUX**

more about: kali linux - [link](https://www.kali.org)

основы linux, habr - [link](https://habr.com/ru/post/110012/)

еще основы - [link](https://habr.com/ru/post/99041/)

security дестрибутивы - [link](https://ravesli.com/luchshie-distributivy-linux-dlya-obespecheniya-bezopasnosti-i-konfidentsialnosti/)

динамические библиотеки в линукс - [link](https://developer.ibm.com/tutorials/l-dynamic-libraries/)

**\*\*\*\*\* WHAT OS IS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

ОПЕРАЦИОННЫЕ СИСТЕМЫ:

- для пользователя

- чтобы взаимодействовать с аппаратной частью

- посредством прикладного программного обеспечения

ОСНОВНЫЕ ФУНКЦИИ:

- управление памятью и контроль её использования приложениями.

- управление использованием вычислительной мощности приложениями

- управление файловыми системами и жесткими дисками

- предоставлнеие интерфейсов для создания драйверов

- предоставление GUI

- работа служб, запускающих серверные приложения

ОСНОВНЫЕ ЧАСТИ ОС

**1.** Ядро — состоит из планировщика, менеджера ресурсов и предварительно загруженных драйверов оборудования.

**2.** HAL (Hardware Abstraction Layer) — слой абстрагирования, реализованный в программном обеспечении, позволяющий взаимодействовать инструкциям высокоуровневых языков программирования с аппаратным обеспечением.

**3.** GUI (Graphical User Interface) — система средств, позволяющая пользователю взаимодействовать с программным и аппаратным обеспечением компьютера.

**4.** Утилиты, библиотеки и фреймворки,

которые совместно используются всеми службами и приложениями.

**5.** Службы и приложения конечных пользователей.

**\*\*\*\*\* BASICS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DICTIONARY

GNU (проект GNU is Not Unix) = набор свободного ПО для Unix

LINUX = ядро UNIX-подобной операционной системы

ДИСТРИБУТИВ = любая OS на основе ядра Linux

GNU/LINUX = полноценная OS из ядра и программного обеспечения

УТИЛИТА = вспомогательная узкоспециальная программа в составе ОС

ALL LINUX COMANDS

Базовые команды = [link](https://ravesli.com/bazovye-komandy-v-linux-for-beginners/)

Все команды = [link](https://ravesli.com/spisok-vseh-komand-v-linux/)

УСТАНОВКА ПО из РЕПОЗИТОРИЯ

Для установки ПО из репозитория обязательно смотреть на описание, которое дается к этому ПО (обращать внимание на зависимости).

Скачать исходники

~$ wget <url> ; скачать по url (как правило \*.tar.gz)

~$ git clone <url> ; скачать из git (как правило \*.git)

Если скачан архив, то разархивировать с помощью утилиты tar

~$ tar [options] <archive> --directory <did> ; dir must exists

**-x** ; распаковать

**-v** ; вывод дополнительной информации stdout

**-f** ; распаковать в файл (не в stdout)

**-z** ; использовать gzip

Если нет дополнительных настроек, перейти в папку и

~$ make ; сборка

~$ make install ; установка

УСТАНОВКА ЯДРА LINUX = [link](https://losst.ru/sobiraem-yadro-linux)

UTILITIES and PROGS

CAT ; работа с небольшими файлами в терминале

SUDO ; "подмена" пользователя для выполнения команды

TAR ; архиватор

APT ; менеджер пакетов (вместо apt-get + dpkg + … )

GREP ; поиск по содержимому файлов

GDB ; отладчик исполняемых файлов

HTOP; исследование списка процессов

\*\*\*\*\* **what LINUX is** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Сейчас бесплатное программное обеспечение распространяется по лицензии под названием GNU GPL (general public license).

LINUX = ядро операционной системы, разработанное в 90х годах Линусом Торвальдсом. Также распространяется по GNU GPL. Эмблема – пингвин Tux (Torvalds Unix)

LINUX было разработано при изучении проекта OS MINIX.

Этот проект = учебная операционная система на базе OS UNIX (которая сама по себе также является сободно распространяемой).

Полноценная OS состоит из ядра LINUX + программ GNU GPL. Любая OS, на основе ядра LINUX называется дистрибутивом LINUX

\*\*\*\*\* **LINUX ARCHITECTURE** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

УТИЛИТЫ = программное обеспечение

ОБОЛОЧКА ("shell", командный интерепретатор) = интерфейс

взаимодействия пользователя и ядра OS. Абстрагирует

внутреннее устройство системы. Дополнительно далее.

ЯДРО = непосредственное взаимодействие с железом

микроядерная арх. = независимые модули подгружаются в память

монолитная арх. = все ядро одновременно в памяти

гибридная арх. = OS в памяти, драйвера подгружаются

[ на данный момент в ядре 20.000.000 плюс строк кода ]

ЖЕЛЕЗО = аппаратное обеспечение

\*\*\*\*\* **MAIN GNU UTILITIES** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GRUB = загрузчик OS. Первая загружаемая программа.

BASH = командная оболочка.

- представляет интерфейс командной строки

- возможность настройки конфигурации рабочего окружения

- создание скриптов на языке bash

DAEMONS = фоновые системные процессы

PACKAGE MANAGER = установка, настройка, обновление и удаление ПО

- установка производится из репозиториев

DISPLAY SERVER = ответственный за работу GUI

DESKTOP ENVIRONMENT = разновидность GUI, абстракция системы

USER APPS = приложения пользователя

\*\*\*\*\* **USERS** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ПОЛЬЗОВАТЕЛИ В LINUX = основа управления

- правами доступа

- уровнем привелегий

Текущие пользователи на виртуалке ubuntu\_installed

- root : sudo

- tr1ang1e : srv

ВЛАДЕЛЕЦ = пользователь, создавший файл

ГРУППА = несколько пользователей

ОСТАЛЬНЫЕ = любой пользователь, имеющий доступ к файлу

СОЗДАНИЕ НОВОГО ПОЛЬЗОВАТЕЛЯ

~$ useradd [options] <username> ; формат команды

-m = создать домашнюю директорию

-s = указать конкретную оболочку

~$ sudo useradd –m –s /bin/bash new\_user ; новый пользователь

~$ sudo passwd new\_user ; задать пароль

~$ sudo userdel <username> ; удалить пользов.

УЗНАТЬ ИНФОРМАЦИЮ О ПОЛЬЗОВАТЕЛЕ

~$ id <user> ; уникальный id и группы пользователя

~$ id –gn <user> ; основная группа пользователя

-$ group <user> ; все группы пользователя

~$ cat /etc/passwd ; вся инфа о всех пользователях

формат информации в файле /etc/passwd

**username** : **pass** : **id** : **id\_группы** : **gecos** : **home\_каталог** : **оболочка**

GECOS = поле основных данных о пользователя. Заполнить:

~$ chfn username ; дальше все интерактивно

ПЕРЕКЛЮЧЕНИЕ ПОЛЬЗОВАТЕЛЕЙ

~$ su ; переключение на root

~$ su <username> ; переключение на username

SUDO

SUDO = substitute user and do

~$ sudo [-u username] <command>

по умолчанию без указания пользователя – от имени root

Описание прав пользователя: ~$ sudo -l

Редактирование прав пользователей = файл **/etc/sudoers**

~$ sudoedit <filename> ; редактирование любого системного файла

\*\*\*\*\* **GROUPS** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ИЗМЕНЕНИЯ В ГРУППАХ

-$ newgrp <group> ; временное изменение основной группы

\*\*\*\*\* **PERMISSIONS (права доступа)** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Каждый файл имеет три типа разрешений:

read r = чтение (просмотр содержимого)

write w = изменение (модификация содержимого)

execute x = выполнение (запуск исполняемого файла)

~$ ls -li

N **xxxx**.**xxx**.**xxx** N user group size date name

0 1 2 3 4 5 6 7

0 = inode (флаг –i)

1 = тип + разрешение

2 = жесткая ссылка : количество ссылок на файл (default = 1)

3 = владелец

4 = группа

5 = размер в байтах

6 = последнее изменение

7 = имя

владелец остальные

группа

rwx.rwx.rwx

**xxx**.**xxx**.**xxx**

ИЗМЕНЕНИЕ ПРАВ ДОСТУПА

~$ chmod <permissions> <file>

permissions = пишутся через запятую

- для кого: u (владелец), g (группа), o (остальные), a (все)

- что сделать: + (добавить), - (убрать), = (переназначить)

~$ chmod u=rwx,g=rw,o=r test.txt ; -rwxrw-r--

~$ chmod g-w,o+w test.txt ; -rwxr--rw-

~$ chmod a+r-w test.txt ; -r--r--r—

Допускается представление разрешений в числовом виде:

0 = ---

1 = --x 5 = r-x

2 = -w- 6 = rw-

4 = r-- 7 = rwx

~$ chmod 740 test.txt ; -rwxr-----

ИЗМЕНЕНИЕ ВЛАДЕЛЬЦА или ГРУППЫ

~$ chown <user> <file> ; только пользователь

~$ chgrp <group> <file> ; только группа

~$ chown <user>:<group> <file> ; пользователь и группа

~$ chown :<group> <file> ; только группа

ПРИОРИТЕТ СЧИТЫВАНИЯ ПРАВ ДОСТУПА

Приортиет выражается в порядке прочтения битов доступа:

1. владелец

2. группа

3. остальные

таким образом, если права ----r--rw- ,

то владелец не сможет работать с файлом

(хотя бы он и входил в группу и тем более в "остальных")

\*\*\*\*\* **SHELL and TERMINAL** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SHELL (оболочка) =

1. командный интерпретатор для взаимодействия user : OS

2. интерпретируемый язык программирования написания скриптов

3. как правило является POSIX-совместимой

Пользователь может:

- давать отдельные команды

- запускать цепочки команд (скрипты)

POSIX = portable open system interface = набор стандартов о:

- интерфейс, предоставляемый OS прикладной программе

- библиотека языка C

- набор приложений и их интерфейсов

КАКАЯ ОБОЛОЧКА У МЕНЯ

~$ echo $SHELL ; текущая оболочка

~$ cat /etc/shells ; все доступные оболочки

СМЕНА ОБОЛОЧКИ

~$ <shell\_name> ; временная смена

~$ chsh –s <shell\_path> ; сменя оболочки по умолчанию

ТЕРМИНАЛ = программа для взаимодействия user : bash

- переключение с помощью ctrl+alt+F1-12

- как правило, F2 = терминал с GUI

КОМАНДЫ ТЕРМИНАЛА =

встроенные в shell / программы, запускаемые shell’ом

~$ type <command> ; узнать тип

~$ help ; все внутренние команды

ПРИГЛАШЕНИЕ = сообщение от оболочки о возможности ввода команды. - $ (обычный user) , # (root user)

- ~ может означать, что находимся в домащнем каталоге user'а

- возможное дополнение: user@host

\*\*\*\*\* **PROCESSES** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

КЛАССИФИКАЦИЯ

1. процессы переднего плана / интерактивные процессы

- инициируются только пользователем

- занимают терминал на время выполнения

2. фоновые / автоматические процессы

- не подключены к терминалу

- могут быть как пользовательскими, так и системными

DAEMONS = фоновые системные обязательные процессы.

запускаются при старте системы

могут управляться пользователем через init-процесс

ИДЕНТИФИКАЦИЯ ПРОЦЕССОВ

Каждый процесс имеет два уникальных номера, один из которых указывает на сам процесс, а другой – на тот процесс, который является родительским (порождающим)

- PID = process id

- PPID = parent process id

Новому процессу может быть назначен любой из свободных (в том числе ранее использовавшийся, но освободившийся) PID.

INIT или SYSTEMD = родитель всех процессов (initialization)

сравнение этих двух систем = [link](https://ravesli.com/linux-init-systems/)

Процесс ЗОМБИ = такой, чей родительский процесс был убит прежде завершения самого этого процесса. Таким процессам в качестве родительского назначается init, а сами они не используются.

Получит PID и другую информацию

~$ echo $$ $PPID ; PID и PPID оболочки

~$ pidof <process> ; узнать PID

~$ htop ; утилита для мониторинга

-$ pstree ; дерево процессов

ПРИОРИТЕТ ПРОЦЕССА

В утилитах по отображению находится в столбце NI

~$ nice –n <num> <process> ; запустить процесс с приортетом

~$ renice +/-<num> <pid> ; изменить приортет процесса

СОСТОЯНИЕ ПРОЦЕССА

СОЗДАН

ЗАВЕРШЕН

ГОТОВ

ВЫПОЛНЯЕТСЯ

ОЖИДАЕТ

прерывание

**отправка планировщику**

возникновение события или ожидание ресурсов

завершение событи или получение ресурсов

- прерываемые

- непрерываемые

УПРАВЛЕНИЕ ПРОЦЕССАМИ

~$ jobs -l ; отобразить фоновые процессы и PID

~$ <process> & ; запустить процесс в фоне

~$ <process> #затем ctrl+z ; отправить процесс в ожидание

~$ bg <process\_jobs\_num> ; запустить ожидающий процесс в фоне

~$ fg <process\_jobs\_num> ; вернуть процесс из фоновых

~$ kill –l ; список всех доступных сигналов

~$ kill <SIG\_NUM> <PID> ; сигнал sig\_num -> процессу pid

1 = управляющий терминал закрыт

2 = остановка как будто ctrl+c

9 = немедленное убийство процесса без доп. операций очистки

15 = сигнал завершения + доп. операции очистки за (defalut)

20 = остановка как будто ctrl+z

~$ kill –KILL <PID> ; жесткое убийство процесса

\*\*\*\*\* **LOGS** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/var/log ; директория логов

boot.log — журнал загрузки системы

kern.log — журнал ядра

syslog, messages — журналы общей активности

auth.log, secure — журналы аутентификации и безопасности

debug — журнал отладки

daemon.log — журнал демонов

faillog — информация о неудачных входах в систему

dmesg — журнал сообщений драйверов устройств

Некоторые логи можно прочитать в обычном тескстовом редакторе. Но не которые записаны в таком формате, что, открыв их в редакторе, ничего нельзя будет понять. Для них предназначены сециальные утилиты:

/var/log/wtmp ; журналы учета входов пользователей в систему

/var/log/utmp ; утилита ~$ utmpdump /var/log/wtmp

SYSTEMD и JOURNAL

systemd – подсистема инициализации и управления службами (может применяться вместо SysV = init). У нее есть демон логов journal

~$ cd /var/log/journal ; переход в необходимую директорию

~$ journalctl -<flag> ; формат команды

ПРИОРТИЕТ СООБЩЕНИЙ ЛОГ-файлов

emerg — наивысший приорите, паника

alert — тревога, стоит волноваться

crit — критическое событие, стоит насторожиться

err — ошибка

warning — предупреждение

notice — уведомление, можно не заморачиваться

info — информационное сообщение

debug — отладочная информация

Можно отфильтровать, например: ~$ grep 'err' /var/log/syslog

\*\*\*\*\* **MEMORY INFO** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-$ cat /proc/meminfo ; основная информация о состоянии памяти

-$ free –h ; без –h данные в кибибайтах (1024 байт)

-$ vmstat ; мониторинг производительности

-$ sudo swapon ; инфо о пямяти подкачки

ПАМЯТЬ ПОДКАЧКИ = память на жестком диске, которая может использоваться для "расширения" оперативной памяти в качестве хранилища временно неиспользуемых данных. При необходимости эти данные, "вытесненные" в пространство подкачки, подгружаются обратно в оперативную память

1 тип = раздел подкачки (partition) ; см. об этом отдельно

2 тип = файл подкачки (file) ; см. об этом отдельно

SWAPPING = процесс вытеснения временно неиспользуемых данных в пространство подкачки. Ответственный = менеджер памяти.

ЧАСТОТА ИСПОЛЬЗОВАНИЯ ПАМЯТИ ПОДКАЧКИ

файл с ответственным параметром: /proc/sys/vm/swappiness

изменение параметра: от 0 (избегать подкачки) до 100

-$ sudo sysctl vm.swappiness=42 ; временное изменение

Для постоянного изменения необходимо добавить в самый конец файла /etc/sysctl.conf строку vm.swappiness=42

ВИРТУАЛЬНАЯ ПАМЯТЬ = память, которой процесс оперирует внутри самого себя. Называется виртуальной, так как не имеет прямого соответствия ФИЗИЧЕСКОЙ ПАМЯТИ. Существует сложный механизм отображения виртуальных адресов на физические – задача ОС.

PAGING = механизм, который используется при применении концепции виртуальной памяти. Заключается в делении всего адресного пространства на страницы (page / например, 4к). Страница виртуальной памяти отображается на страницу физической памяти. При этом адрес физической страницы может меняться, однако приложение не заметит этого изменения, так как действует в виртуальном адресном пространстве.

\*\*\*\*\* **TTY** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TTY = ТЕРМИНАЛ = программа для взаимодействия с оболочкой

TTY = teletype

Система генерирует для пользователя несколько tty.

Навигация по ним: CTRL + ALT + F\_

3 - 6 = полноэкранные tty3 – tty6 соответственно

2 = GUI с собственный tty и множественными PTS

1 = переход на экран авторизации пользователей

PTS = псевдо-терминал / виртуальный tty = одновременно несколько

PTS = pseudo-teletype

-$ tty ; номер TTY либо PTS (в зависимости от)

\*\*\*\*\* **SYSTEM VARIABLES** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Переменные в системе = набор именованных значений, используемых приложениями. Применябтся для необходимых настроек.

ENVIRONMENT VARIABLE (переменных окружения / среды):

- доступны по всей системе

- наследуются всеми дочерними оболочками + процессами

SHELL VARIABLE (переменные оболочки):

- применяются только к текущему экземпляру оболочки

- у каждой оболочки собственный набор

Привила определения переменных

- только заглавные буквы и подчеркивание

- нет пробелов около оператора присваивания

- несколько значение переменной разделяются оператором :

Узнать переменные:

-$ set ; all

-$ printenv ; environmental

-$ echo &<VAR> ; shell variable VAR

Установить временные переменные:

-$ VAR=42 ; shell variable

-$ export VAR ; shell -> environmental variable

-$ export VAR=42 ; environmental variable

При выходе из оболочки (= закрытии терминала) они будут удалены

Установить постоянные переменные = добавить в конец файла:

- для текущего польщователя: ~/.bashrc

- для всех пользователей: /etc/environment or /etc/profile

Дополнительные файлы с переменными, доступными по правилам

- ~/.bash\_profile = правило (?)

- /etc/bash.bashrc = правило (?)

Удалить переменные

-$ unset VAR

- удалить из соответствующего файла

\*\*\*\*\* **FILE SYSTEM** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FS = ФАЙЛОВАЯ СИСТЕМА = часть программного обеспечения:

1. предоставляет интерфейс для доступа к файлам

2. обеспечивает координацию доступа к файлам

A. набор структур данных, описывающих файлы + саму память

B. комплекс программных средств для управления ими

У каждого устройства, имеющего память, есть собственная файловая система. В компьютере файловая система интегрирована в OS.

МОНТИРОВАНИЕ = подключение файловой системы внешнего устройства в файловую систему компьютера = [link](https://losst.ru/montirovanie-diska-v-linux), [link](https://losst.ru/montirovanie-fleshki-linux#Монтирование_флешки_в_Linux)

VFS = ВИРТУАЛЬНАЯ файловая система = механизм OS, являющийся абстракцией над файловой системой. Предоставляет файловой системе интерфейс взаимодействия с памятью. Необходим для того, чтобы разные файловые системы, совместимые с этим интерфейсом:

- не требовали изменений в ядре OS

- не требовали изменений в "поведении" пользователя

Таким обращзом, ответственность за совместимость внешнего накопителя с OS сводится к совместимости с VFS и возлагается на создателя внешнего накопителя.

КОМПЬЮТЕР

user

OS

ПАМЯТЬ

VFS

FS

ВНЕШНИЙ ЛОКАЛЬНЫЙ НОСИТЕЛЬ

FS

ПАМЯТЬ

СЕТЕВОЙ НОСИТЕЛЬ

FS

ПАМЯТЬ

$ df –Th ; инфо о дисковом пространстве плюс FS в OS

\*\*\*\*\* **FILE TYPES** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

7 основных ТИПОВ ФАЙЛОВ = xxxxxxxxxx

- обычный файл

d директория

c символьное устройство (последовательный ввод-вывод)

b блочное устройство

s локальные сокеты

p именованные каналы

l символьные ссылки (магкие = ярлык, жесткие = псевдоним) [link](https://losst.ru/simvolicheskie-i-zhestkie-ssylki-linux)

Дополнительную информацию можно получить с помощью команды:

~$ file <filename>

В системе линукс используется принцип **ВСЕ ЕСТЬ ФАЙЛ**.

Этот принцип разработан для того, чтобы избежать большого количества интерфейсов – роль интерфейса системы/устройств и проч. выполняют файлы.

Каждый из них можно открыть в текстовом редакторе и, работая с содержимым, взаимодействовать с той "сущностью", которая представлена этим файлом.

Не всегда с этой задачей справится обыкновенный текстовый редактор (например, при чтении бинарника).

Однако в таких случаях существуют специальные утилиты.

\*\*\*\*\* **FILE MANAGEMENT** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-$ touch <file> ; create file

-$ mkdir <dir> ; create directory

-$ mv <what> <to> ; move (or rename) file/directory

**-f** ; forced replace

**-n** ; don't replace

**-i** ; ask if replace

-$ cp <what> <to> ; copy file/directory

-r ; copy directory recursively

\*\*\*\*\* **INOD.** **HARD and SYMBOLIC LINKS** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Имя файла связывается со структурой INOD, в которую входят:

- индексный дескриптор (уникальный идентификатор)

- метаинформация (атрибуты файла и проч.)

Таблицы inod'ов хранятся, как правило, в начале разделов.

ДИСК

РАЗДЕЛ

ФАЙЛ

ФАЙЛ

ФАЙЛ

ИМЯ ФАЙЛА - INOD

ТАБЛИЦА INOD'ов

ДИРЕКТОРИЯ = файл специального формата,

который содержит информацию об именах вложенных файлов (в т.ч. директорий) и номерах их inod'ов.

Таким образом, каждая директоия также имеет свою запись в таблице inod'ов.

При создании HARD LINK ее имя связывается с той же структурой.

Характеристики:

- связана с содержимым файла (по сути, псевдоним)

- работает только внутри своей фаловой системы

- работает вне зависимости от расположения файла в ней

- нельзя использовать с директориями

- файл жив, пока жива хотя бы одна жесткая ссылка

SYMBOLIC LINK это файл, который содержит только путь. Если этот путь валидный (то есть ведет к другому файлу), то по нему можно получить доступ к тому самому файлу – в пртивном случае, путь просто никуда не ведет.

- не связана с содержимым файла

- работает из любого места любой файловой системы

- валидная ссылка при перемещении файла становится невалидной

- можно использовать с директориями

- никак не влияет на продолжительность жизни файла

- всегда имеет разрешения rwxrwxrwx , что не имеет значения

$ ln <source> <link\_name> ; hard link

$ ln –s <source> <link\_name> ; symbolic link

\*\*\*\*\* **FILESYSTEM HIERARCHY STANDARD** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FHS = filesystem hierarchy standard = стандарт иерахии файловой системы, определяет структуру и содержание (назначение) каталогов дистрибутива Linux.

/ : корневая директория

/boot : загрузочные файлы (GRUB, ядра Linux)

/etc : конфигурационные файлы

/bin : бинарники системных программ и утилит

/sbin : бинарники утилит системного администрирования

/lib : библиотеки для бинарников из bin и sbin

/usr : пользовательские бинарники и программные данные

/opt : чаще всего для проприетарного по

/proc : системная информация времени выполнения

/var : файлы изменяемых данных (журналы, кэш программ, инфа)

/srv : системные служебные данные

/home : домашние директории пользователей

/root : домашняя директория root-пользователя

/run : временные файлы (не могут быть удалены)

/tmp : временные файлы (могут быть удалены)

/dev : файлы устройств (символьные и блочные устройства)

/media : точка монтирования съемных носителей

/mnt : точка монтирования файловых систем

/lost+found : поврежденные файлы с шансом на восстановление

\*\*\*\*\* **SEARCHING FILES** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

~$ find [optional] <path> <flags> <expression>

- [optional] = -H , -L , -P , -O<level> , -D <debugopts>

- <path> : по умолчанию, директория = текущая

: по умолчанию, поиск рекурсивный

- <flags> : -name , -inum , -mtime …

- выражение может быть без кавычек, если не нужно экранировать

Полная информация о find ~$ info find

Краткая информация о find ~$ find --help

Если добавить флаг –L, то утилита find – при считывании файла символической ссылки – будет учитывать характеристики не самой ссылки, а того файла, НА КОТОРЫЙ она сылается. Однако учитываться будут только поля из соответвтующего INODE.

Имя файла НЕ ЯВЛЯЕТСЯ полем INODE

Например: ~$ touch test

~$ ln –s test link

~$ find –name 'test' ; ./test

~$ find –L –name 'test' ; ./test

~$ find –inum 123456 ; ./test

~$ find \_L –inum 123456 ; ./test ./link

ПОИСК БИТЫХ ССЫЛОК ~$ find <path> -xtype l

ВЫПОЛНЕНИЕ КОМАНД НА РЕЗУЛЬТАТАХ ПОИСКА

~$ find … -exec rm {} \;

- после команды find флаг –exec

- далее команда, которая выполнится после поиска

- {} указывает на то, что команда применится к результатам

- после каждой команды нужна экранированная ;

\*\*\*\*\* **/DEV** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/dev/null : "черная дыра" для данных

/dev/zero : "черная дыра" плюс бесконечный поставщик нолей

/dev/random : бесконечный поставщик случайных чисел

\*\*\*\*\* **ПОТОКИ ВЫВОДА** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

STDOUT 1 по умолчанию связан с терминалом

STDERR 2 по умолчанию связан с терминалом

Для перенаправления потоков (N = номер потока 1, 2 или & (оба)):

~$ … N><source>

~$ echo 'test' 1>/dev/null

\*\*\*\*\* **утилита** **APT** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ОПЦИИ

-v, --version - выводит версию утилиты

-h, --help - выводит справку по использованию утилиты

-y, --yes - автоматически отвечать "да"

--assume-no - автоматически отвечать "нет

-d, --download-only - только скачать пакеты

-f, --fix-broken - исправить недостающие зависимости

--no-download - ничего не загружать (пакеты из кэша)

-s, --simulate - режим симуляции

--allow-unauthenticated – установка без GPG подписи

--no-install-recommends - не устанавливать рекомендованные

--only-upgrade - только обновлять пакеты

--allow-downgrades - разрешить откатывать версию пакетов

--reinstall - переустановить установленный пакет

КОМАНДЫ

install - установить пакет;

remove - удалить пакет без конфигурационных файлов

purge - удалить пакет вместе с конфигурационными файлами

autoremove - очистить ненужные пакеты;

autoclean - очистить кэш пакетов;

update - обновить списки пакетов из репозиториев;

upgrade - обновить версию пакета до последней

full-upgrade - полное обновление системы

list - список установленных пакетов;

search - поиск пакетов;

show - посмотреть информацию о пакете;

download - скачать пакет в текущую папку;

source - скачать исходный код пакета в текущую папку;

build-dep - установить необходимый пакета зависимости