

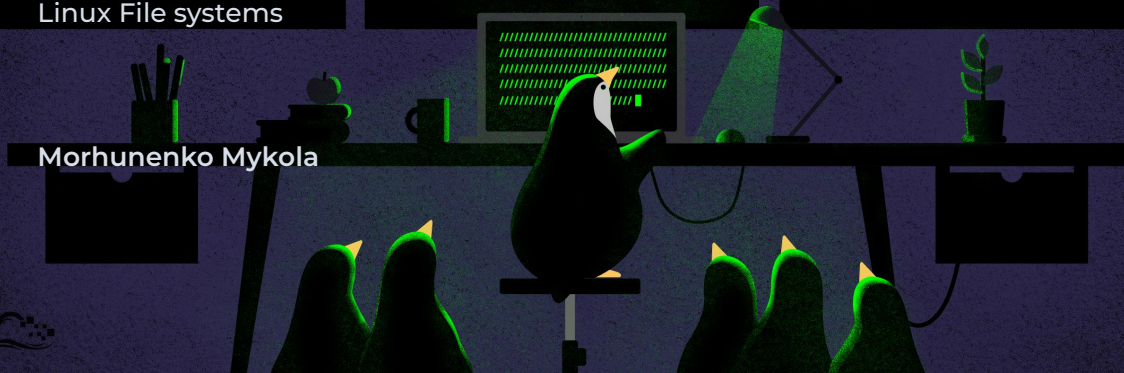


APPS@UCU

Linux course

Linux File systems

Morhunenko Mykola



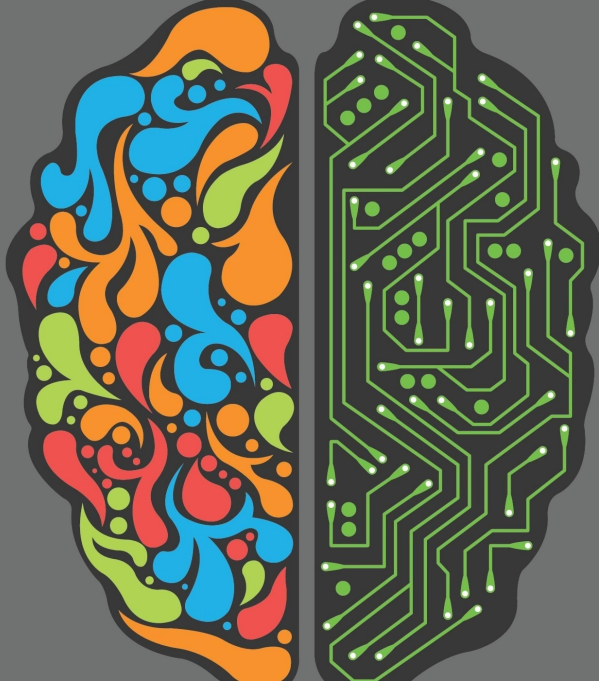
Contents

- 1 Memory overview
- 2 Everything is a file
- 3 Fyle systems
- 4 The most important filesystems overview
- 5 The most important filesystems overview
- 6 Working with file systems
- 7 Linux Filesystems Hierarchy (LFS)
- 8 Sources

Intro

- This is not an overview of some **hardware** memory stuff
- Neither a presentation with deep File systems implementation details
- More about that you should learn at the **Operating systems** course
- This is just an overview of **file systems** that system administrators use in their everyday life
- If you think that you are not a system administrator - think one more time, because you administrate your own system every day

Memory



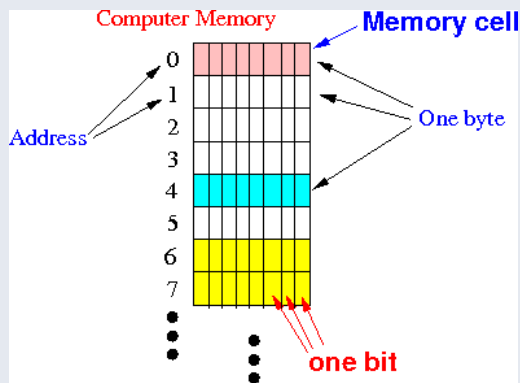
Drives

- All data stored on some physical devices
- It has different storage approaches on each device (HDD, SSD, CD, DVD, Flash, RAM, DDR memory modules)
- But now we are going to overview the memory from **user point of view**
- How to manage files and file systems, how to choose the most suitable



Memory storage

- Memory as abstraction looks like an array, where bites are stored one by one in a row
- **File system** - a method of data structure that the operating system uses to control how data is stored and retrieved
- A **file** is an ordered collection of data blocks
- In Linux system, everything is a file, and if it is not a file, it is a process
- So File systems are essential for this OS





Everything is a file

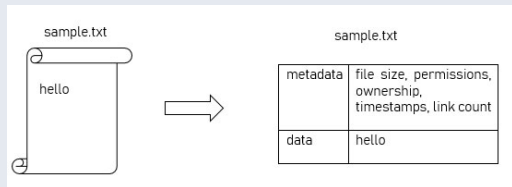
File types

- There are a lot of file types, but the most important for us are:
- **Regular Files** - some files with data stored inside
- **Directories** - files, that allowed to group other files and keep tree filesystem structure
- **Character files** - for simulating character devices as terminals, keyboard, network etc
- **Block files** - for modelling block devices as disks, flash drives
- **Links** - entry points to other files
- There are **pipes** , **sockets**

- rw-----	: Regular File
d rw-r-xr-x	: Directory File
l rw-rw-rw-	: Link File
C rw-rw----	: Character Device File
S rw-rw-rw-	: Socket File
p rw-----	: Named Pipe File
b rw-rw----	: Block Device File

File metadata

- File also save a **metadata** about itself, as:
- Protection, password
- Creator, owner
- Flags (r w x)
- Size
- Creation time, last update time (timestamp)

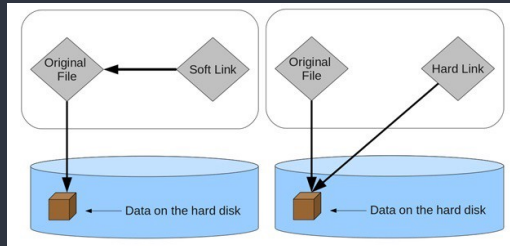


Inode

- The **inode** is data structure, that describes files on **Unix-like OS's**
- Each inode stores disk block location, some attributes, file's metadata
- **Directory** - just a file with list of inodes
- File's inode number can be found with **ls -li** command
- From the inode number, the kernel's file system driver can access the inode contents, including the location of the file, thereby allowing access to the file
- More about **inodes** in the **Operating systems** course

Links

- There are two types of links: **symbolic (soft)** and **hard**
- They are totally different types of file
- Maybe first few years you will not use it
- But with experience it becomes more and more useful
- Here we will make only a brief overview and comparison



Hard links

- Exact replica of a file
- Share same inode with other hard links
- Can not be made across filesystems
- Changes in **hl** will reflect in other files
- Deleting of a hardlink wil not affect other files
- Can links to files only

Soft links

- Alias to a file
- Has another inode
- Can be established outside filesystem
- Link becomes inaccessible without original file
- Can links to both files and directories



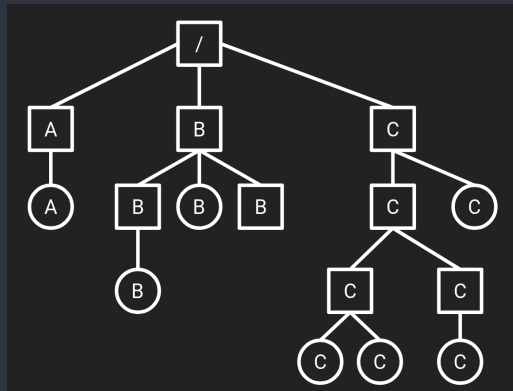
File systems

Fyle systems types overview

- There are several file systems types. Just for your information. the most important will be in orange colour
- **Disk file systems** for simple disks, a.e. FAT16/32, NTFS, ext2-4, brtfs etc
- **Flash file systems** - consider speciality of flesh memory devices
- **Database file systems** - another concept for file management
- Transactional file systems
- **Network file systems** - acts as a client for a remote file access protocol, providing access to files on a server, a.e. FTP
- **Shared disk file systems** - a number of machines (usually servers) all have access to the same external disk subsystem
- Flat file systems - no subdirectories, directory entries for all files are stored in a single directory

Fyle system abstraction

- We used to see a filesystem as a tree. It is the most comfortable structure as for now
- There is a CLI tool to see your filesystem structure called **tree**
- Using such abstraction programmer works with files and directories, not with memory cells or some low-level stuff, but with files, directories, and subdirectories



Overview of the most important filesystems

More details on the [Operating systems](#) course

- Remark: [ext](#) stands for extended
- [ext2](#) - year 1993. File size can be to 2TB, file system size to 32TB
- [ext3](#) - year 2001. Linux Kernel > 2.4.15. Max 32'000 subdirectories. Main benefit - allows different types of journalling (tracking of all changes). Easy to convert ext2 -> ext3. (later) Can be mount as ext4
- [ext4](#) - year 2008. Linux Kernel > 2.6.19. Max 64'000 subdirectories. File system size up to 1EB (10e12GB). Option of turning the journaling feature "off". Also some features as multiblock allocation, delayed allocation, journal checksum, fast fsck, etc was introduced
- [btrfs](#) or [B-Tree filesystem](#) - modern Copy-on-Write (CoW) filesystem. [Comparison of ext4 and btrfs link](#). As for me, the most important features are [fs snapshot](#) and [multiple devices support](#), [built-in RAID support](#)
- [ext4](#) designed to be simple and stable, mostly for local-using, while [btrfs](#) to be high-performance, high-capacity and high-performance, mostly for storage servers

Overview of the most important filesystems

More details on the [Operating systems](#) course

- [ZFS](#) - this fs mostly used on data storages
- It has RAID support, Copy-on-write, Data integrity verification and automatic repair, Snapshots, Maximum 256 Quadrillion Zettabytes storage and Pooled storage
- It combines both fs and volume manager in one. Easy to add physical drive and extend partition size or replace physical drives, to use and maintain
- [FAT](#) - File Allocation Table fs. There are FAT16, 32, 64. Nowadays wide used on USB flash drives. More info and work with this fs on [Operating systems](#) course
- [NTFS](#) - fs used only on [windows](#) os, so it is a proprietary journaling file system, But it can be mount to Linux OS (so you can access your win files from Linux in case of dual-boot)

Swap

- **Swap** is not a filesystem type
- It is a space on a disk, used when there is no space left in RAM, or because of some optimization processes
- Also the only possible way for **hybernation**
- It could be both **swap partition** and **swap file** , but first is much better
- **mkswap** - to make a swap partition
- **swapon / swapoff** - obvious

Working with file systems



Review of previous topics

- It's part of presentation about `shell`, but let's make a brief overview
- Every process has its own working directory.
- `pwdx $(pgrep process_name)` - show working directories for a process_name
- `pwd` - print working (current) directory
- `ls` - list what is inside the working directory
- `cd` - change directory
- `./` - special, current directory
- `../` - special, parent directory (in a tree structure)
- `~/` - `$HOME` directory for current user
- `cp <from....> <to>` - copy
- `mv <from....> <to>` - rename (inside one fs, or move - from one to another fs)
- `mkdir` - make directory
- `touch <filename>` - update the last access date (if no such file - create)
- `rm <filename>` - remove
- `cat` - show the file content

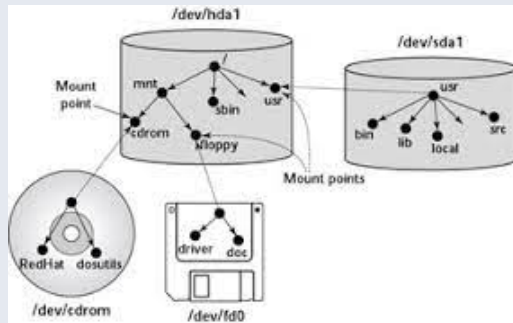
Devices

- Everything is a file , devices are not an exception
- All devices are in `/dev` folder
- Devices could be either secondary storages or mous, keyboard, terminals, cpu, gpu etc
- Devices can be either block or character devices
- Easy to remember:
- Block devices store or hold data
- Character devices - transmit or transfer data



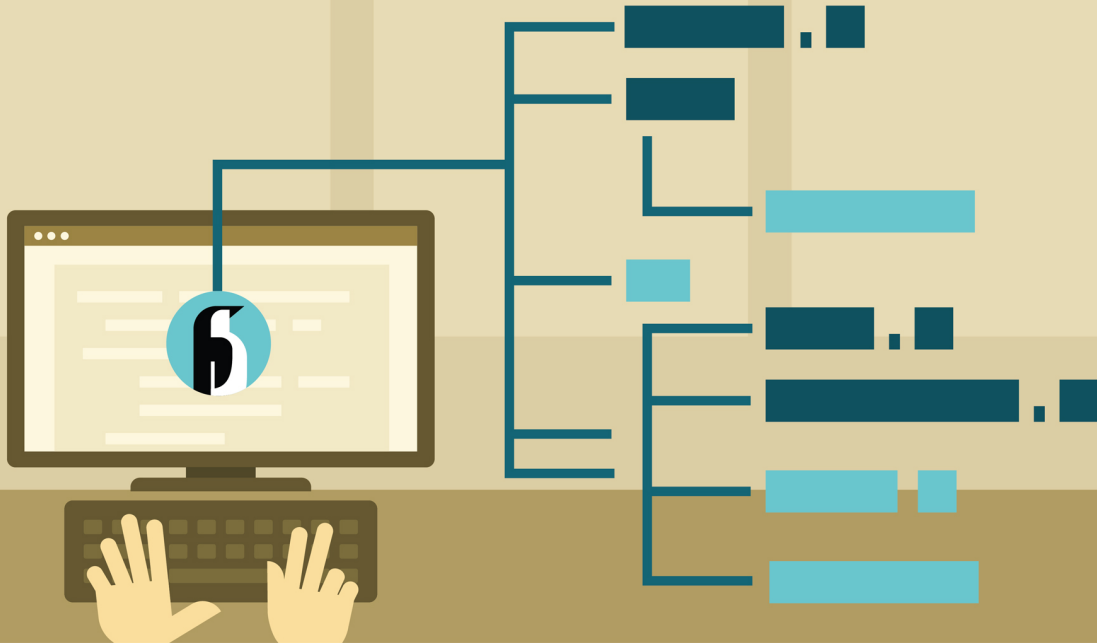
Mounting

- **Mounting** - attaching some additional fs to already mounted
- By default, user use only one filesystem, and it's mountpoint is `/`
- Then `/boot` is also another fs, that is not used by user directly (more about that **bootloader** topic)
- On one physical device there could be few filesystems (more about that **partition tables** topic)



A smorgasbord of important commands and

- `mkfs` - make a file system on a device
- `mkfs.filesystemtype /dev/X` - create a filesystem on existing logical device
- `fdisk -l` - to see all devices available for mounting
- `mount /dev/X /mnt` - to mount device. `mnt` is used as a convention, and it is important
- `/etc/fstab` - file with all "default mountings" during startup. Usually has only `/` and `boot` fs's
- `UUID` - Universally unique identifier - 128bit label. The probability that a UUID will be duplicated is close enough to zero to be negligible. `HERE` - unique device identifier
- `df, du` - are tools for capacity and used memory stats, but I recommend `ncdu`
- `parted` - tool for creating and editing partitions (`or use GParted`)



Linux Filesystems Hierarchy Standard(FHS)

- This topic is worth a separate lecture
- We will make a brief overview
- Every time while using your OS you can open this presentation
- But soon enough you will remember all this stuff
- Linux Filesystem Hierarchy Standard
- Maintained by Linux Foundation
- All distros can voluntarily conform to the FHS, and most of them do
- In general, file system structure is important
- Please , Do not keep all your files in /home/username !
- There are /Downloads, /Documents, /Pictures, /Programs
- That will be much easier to move around and remember all the paths if there is some pattern

/ ROOT

/ BIN "ESSENTIAL BINARIES"

CAT
CHGRP
CHMOD
CHOWN
CP
DATA
DD
DF
DMESG
ECHO
FALSE
HOSTNAME
KILL
LN
LOGIN
LS
MKDIR
MKNOD
MORE
MOUNT
MV
PS
PWD
RM
RMDIF
SED
SH
STTY
SU
SYNCH
TRUE
UMOUNT
UNAME

/ BOOT "STATIC FILES OF BOOT LOADER"

KERNEL
SYSTEM.MAP
VMLINUZ
INITRD
GRUB
MODULE.INFO
BOOT

/ ETC "HOST SPECIFIC SYSTEM CONFIG"

CSH.LOGIN
EXPORTS
FSTAB
FTPUSERS
GATEWAYS
GETTYDEFS
GROUP
HOST.CONF
HOSTS
HOSTS.ALLOW
HOSTS.DENY
HOSTS.EQUIV
HOSTS.LPD
INETD.CONF
INITTAB
ISSUE
LS.SO.CONF
MOTD
MTAB
MTOOLS
NETWORKS
PASSWD
PRINTCAP
PROFILE
PROTOCOLS
RESOLV.CONF
RPC
SECURETTY
SERVICES
SHELLS
SYSLOG.CONF

/ OPT "CONFIG FILE FOR ADD ON APPLICATION SOFTWARE"

/ USR "SHAREABLE AND READ-ONLY DATA"

/ LOCAL

"LOCAL
SOFTWARE"

/ BIN
/ GAMES
/ INCLUDE
/ LIB
/ MAN
/ SBIN
/ SHARE
/ SRC

/ SHARE

"STATIC DATA
SHAREABLE
AMONG ALL
ARCHITECTURES"

/ MAN

"MANUAL PAGES"
/ MAN1 "USER PROGRAMS"
/ MAN2 "SYSTEM CALLS"
/ MAN3 "LIB FUNCTIONS"
/ MAN4 "SPECIAL FILE"
/ MAN5 "FILE FORMATS"
/ MAN6 "GAMES"
/ MAN7 "MISC"
/ MAN8 "SYSTEM ADMIN"

/ BIN

"MOST USER COMMANDS"

/ INCLUDE

"STANDARD INCLUDE
FILES FOR 'C' PROG"

/ LIB

"OBJ, BIN, LIB
FILES FOR PROG
AND PACKAGES"

/ SBIN

"NON ESSENTIAL
BINARIES"

/ VAR

"VARIABLE DATA FILES"

/ CACHE

"APPLICATION
CACHE DATA"

/ LIB

"VARIABLE STATE
INFORMATION
REMAINS AFTER
REBOOT"

/ YP

"DATA FOR
NIS SERVICES"

/ LOCK

"LOCK FILES FOR
SHARED RESOURCES"

/ OPT

"VARIABLE DATA OF
PACKAGES INSTALLED"

/ RUN

"INFO OF SYSTEM
SINCE IT WAS BOOTED"

/ TMP

"AVAILABLE FOR PROG"

/ SPOOL

"DATA AWAITING
PROCESSING"

/ LPD

/ MQUEUE

/ NEWS

/ RWHO

/ UUCP

/ LOG

"LOG FILES
AND DIR"

LASTLOG
MESSAGES
WTMP

/ SBIN

"SYSTEM BINARIES"

FASTBOOT
FASTHALT
FDISK
FSCK
GETTY
HALT
IFCONFIG
INIT
MKFS
MKSWAP
REBOOT
ROUTE
SWAPON
SWAPOFF
UPDATE

/ TMP

"TEMPORARY FILES
DELETED ON BOOTUP"

/ DEV

"LOCATION OF SPECIAL
OR DEVICE FILES
[CONTAINS MAKEDEV]"

/ HOME

"USER HOME
DIRECTORIES"

/ LIB

"LIBRARY AND
KERNEL MODULES"

/ MNT

"MOUNT FILES
FOR TEMPORARY
FILESYSTEMS"

/ OPT

"ADD-ON APPLICATION
SOFTWARE"

/ ROOT

"HOME DIR. FOR
ROOT USER"

- `/` - Root directory of the entire file system hierarchy
- BUT `/` at the end of a path means that it is a directory, not a file
- `cat /etc/fstab` could show you which device is mounted to `/ (root)` point
- `/bin` - NOW - symlink to `/usr/bin`
- `/boot` - boot loader files are here. There can be `/boot/grub` and `/boot/efi` folders, some other files and directories related to bootloaders
- `/dev` - devices (including `/dev/null`, `/dev/random`, `/dev/ttyX`, `/dev/sdX`)
- `/etc` - (et cetera - historically). Now - directory with all global system configurations and specifications. Don't recommend to change anything here
- `/home` - directory that stores ALL users data (except root)
- `/root` - home for root
- `/lib` - NOW - symlink to `/usr/lib`
- `/mnt` - Recommended as a mountpoint for mounting devices
- `/opt` - optional software, propriet sw, most of sw from AUR is here
- `/proc` - virtual filesystem providing info about kernel and all running processes. Generated and populated by OS
- `/sbin` - NOW - symlink to `/usr/bin`

- `/srv` - server specific data (a.e. to use pc as ftp or another server, all shared data is here)
- `/sys` - contains an information and interaction tools with kernel, devices, kernel modules
- `/tmp` - directory for temporary files as `makepkg` cache, etc. Cleared after each reboot
- `/usr` - Universal System Resources - all of the user data
 - `/usr/bin` - file contains programs binaries for all users
 - `/usr/include` - standard include files
 - `/usr/lib` - file contains libraries for `/usr/bin`
 - `/usr/local` - tertiary hierarchy for local data, specific to this host. Contains `/bin`, `/lib`, `share` inside
 - `/usr/share` - architecture-independent data, a.e. fonts, icons, themes, etc
 - `/usr/src` - source code of some programs a.e. kernel, rust, nvidia
- `/var` - variable files, that are permanently changing during normal work a.e. logs, cache, email files. Not cleared between reboots
- **Important, must read** : difference between `/bin`, `/sbin`, `/usr/bin`
- But it is the info from 2010. Now it is **Ok** to have them as symlinks to `/usr/bin`

Sources

Sources

- UCU Linux Club resources
- File systems Wiki
- Linux file systems
- LFSH Wiki
- Differences between hard and soft links on Unix systems
- Mounting and unmounting on Linux
- UUID Wiki
- /bin and /usr/bin differences
- Understanding the bin, sbin, usr/bin , usr/sbin split
- ext2-3-4 differences
- btrfs vs ext4
- The case for /usr merge