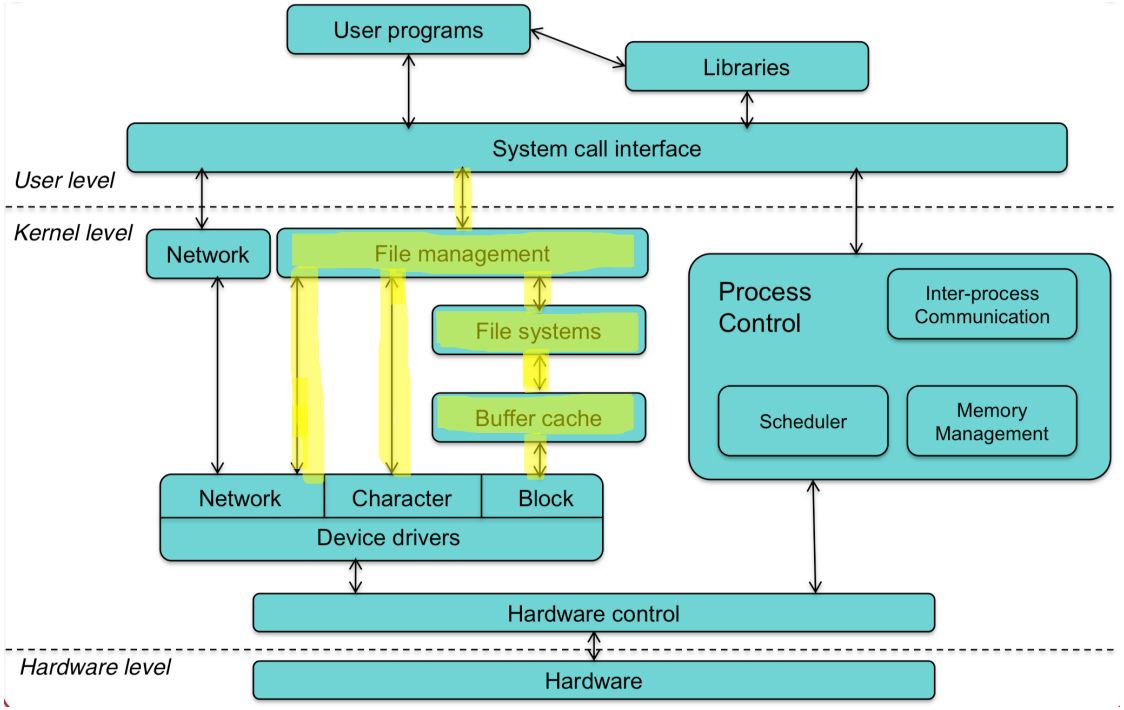
LS-06:
FILE SYSTEM EXAMPLES



Agenda

Standard FS

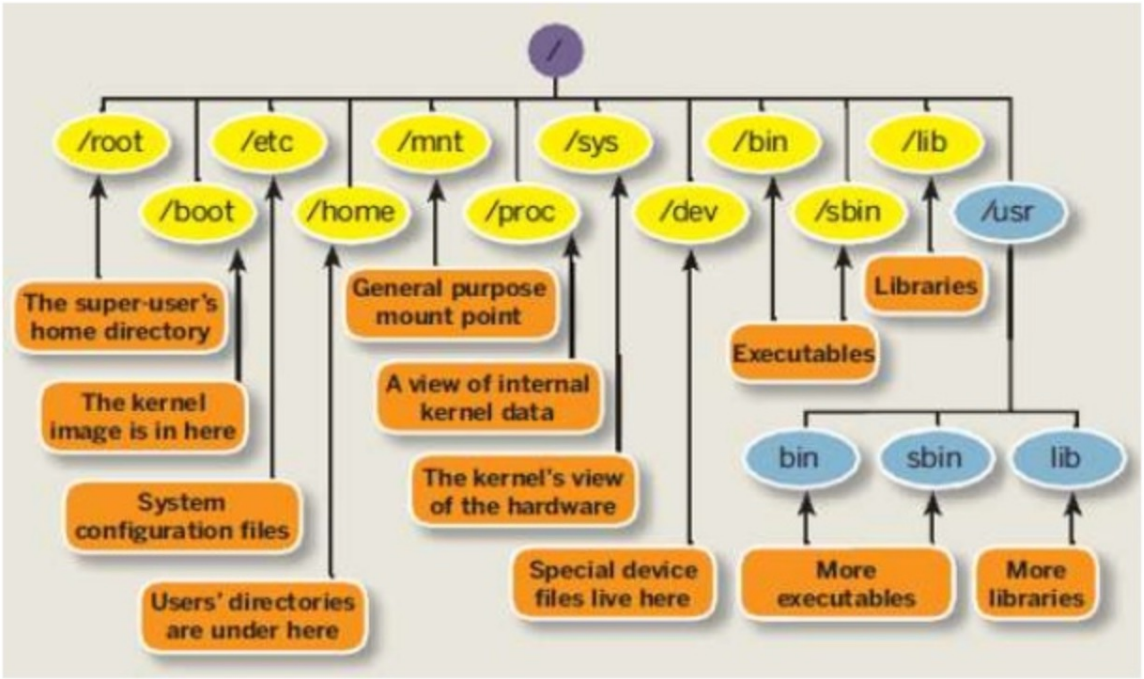
- Old FS
- Linux Standard FS
- Windows Standard FS

Linux FSH

- Linux File System Hierarchy (FSH)
- Linux File Systems Commands
- Linux VFS
- Linux Pseudo FS

Modern FS

- Journaling FS
- Advanced FS
- Distributed FS



STANDARD FS

Old FS
Linux Standard FS
Windows Standard FS

Standard File Systems Examples

Historical & Standard FS Examples

- Mark 1 FS (1958)
- TAR FS (1979)
- RT-11 (1975)

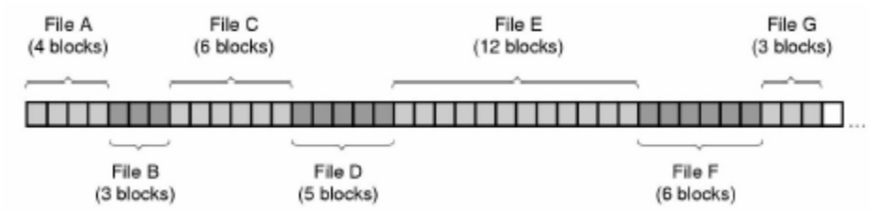
- s5fs - UNIX System V AT&T (1982)
- ffs - Fast File System BSD (1983)
- ufs - Unix FS BSD & Sun Microsystem Solaris (1984)
- Modern BSD use 3-layers UFS/FFS/LFS
- HFS – HierarchicalFileSystem,MacOS<8.1,16bit,<1998
  →HFS+ (MacOS 8.1, 32bit-address) (since 1998)
  →HFSJ (MacOS 10.2.2, journal)
  → HFSX or APFS (MacOS 10.3, more features)

- CP/M – Digital Research (1977)
- FAT 12/16/32
- extFAT
- hpfs IBM-Microsoft High performance FS, OS/2 (1988)
- ntfs – Microsoft New Technology FS, Windows (1993)

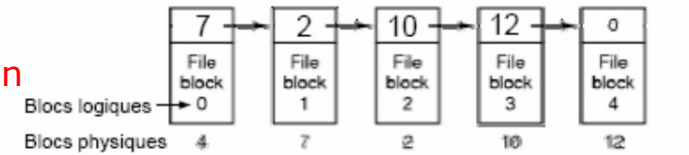
Class of file system by block allocation-addresation

- Contiguous blocks
- Linked-list blocks
- FAT-based
- Inode-based
- Extent-based
- Balanced Tree.

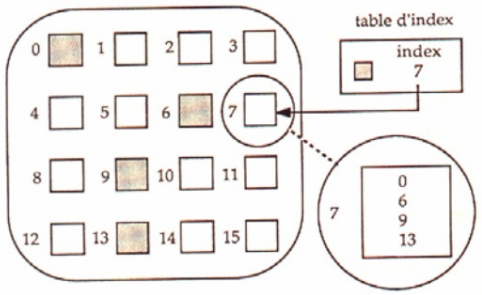
Continue Allocation (TAR, RT-11)



Chain Allocation (FAT)



Index Allocation (s5fs)



File Systems (ERMA) Mark 1

ERAM - first file system (1958)

- The oldest file system ever recorded is the **Electronic Recording Machine Accounting (ERMA) Mark 1**, a hierarchical file system that was introduced in 1958 at the Eastern Joint Computer Conference.
- In summary, the purpose of the file system was to reduce the inefficiencies and errors that resulted from the lack of an organized system
- The idea was to provide more accurate information more quickly and efficiently.

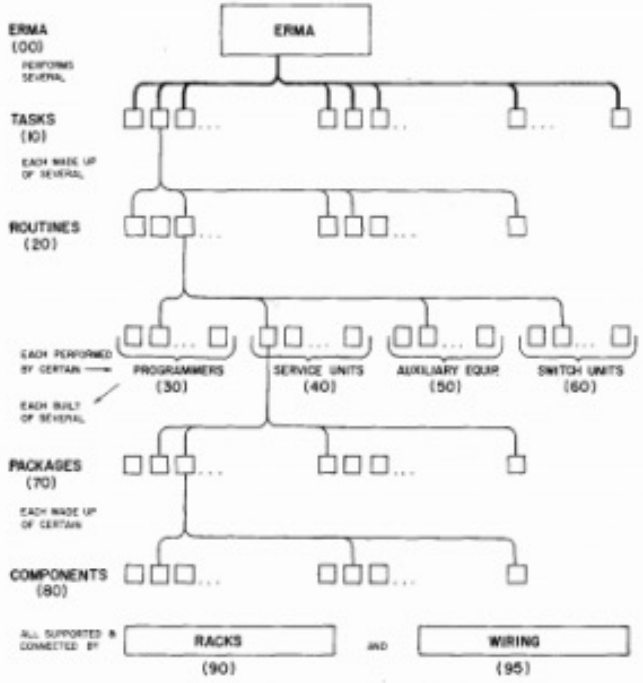


Fig. 2. The ERMA Mark I hierarchical structure

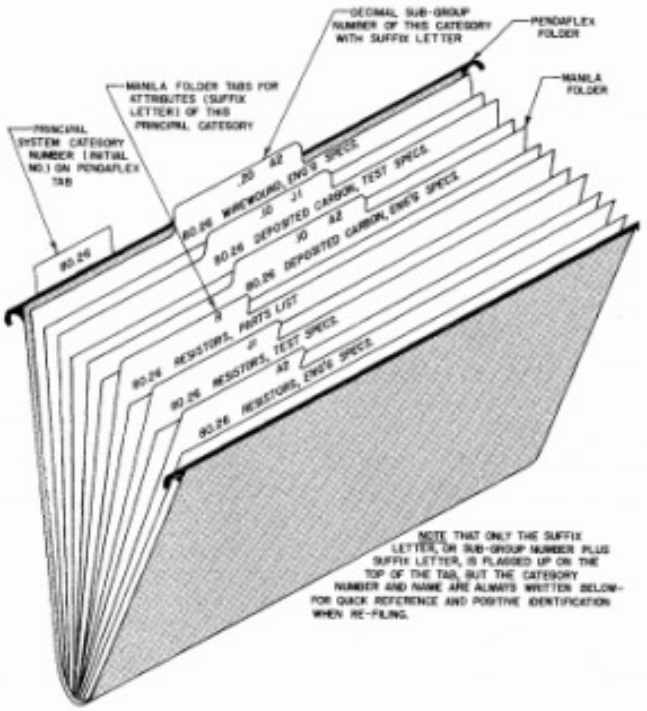
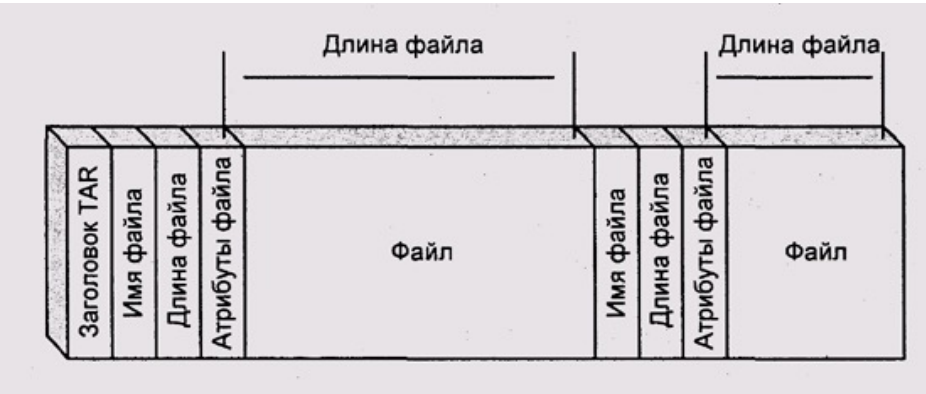


Fig. 3. Records storage and arrangement

A representation of the ERMA Mark 1 file system

File Systems TAR

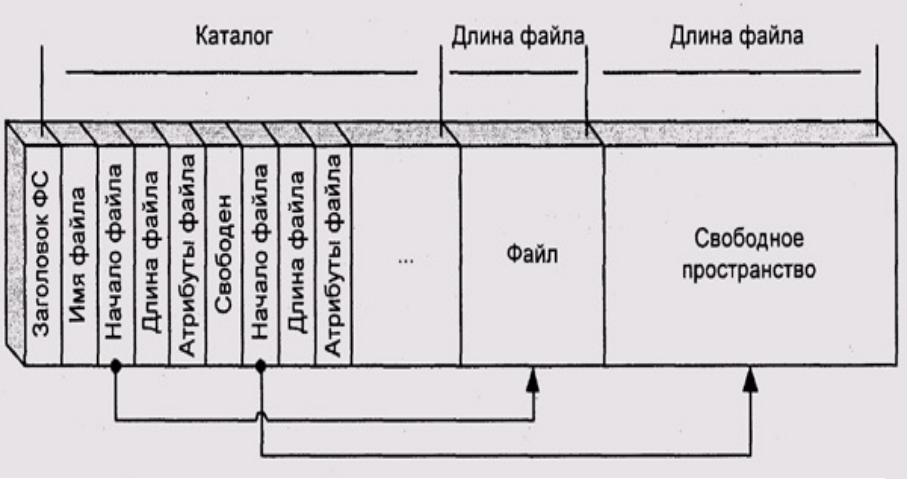
- TAR (Tape ARchiver) (1979)
- TAR together with GZ **active use today** for archiving file structure in one compact file for sending this file over networks, for packet distribution, for tape archiving.
- Structure:
  - TAR Header
  - File-1 name, size, attributes, blocks
  - …
  - File-N name, size, attributes, blocks
  - Free Space
- Advantages: simple, high read/create speed
- Disadvantages:
  - Long time of file seek (seek have before read, find, create,…), because need read parameters of every file.
  - Problem after delete file – holes structure.
  - Problem after resize file



Continue Allocation (TAR, RT-11)

File Systems RT-11

- RT-11 (Real Time OS, PDP-11 DEC (1975-199x)
- More usability then at TAR
- Have only 1 catalog
- Structure:
  - RT-11 Header
  - Catalog
    - File-1 (name, start, size, attributes)
    - …
    - File-n (name, start, size, attributes)
    - FreeSpace-1
    - …
    - File-n+1 (name, start, size, attributes)
  - File-1 blocks
  - …
  - FreeSpace1 blocks
  - File-n+1 blocks
  - …
- Periodically need compacting of file system after remove and resize files.

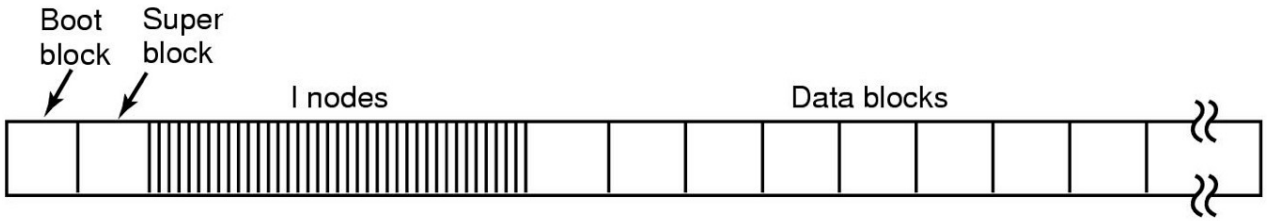


Continue Allocation (TAR, RT-11)

File Systems s5fs

s5fs Structure (One partition)

- s5fs - original FS of UNIX System V, AT&T Corp. (1982)

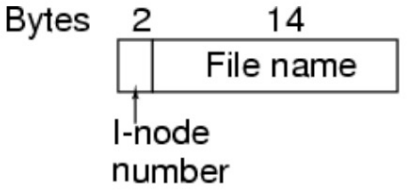


s5fs Super Block contains:

- size in blocks of the file system
- size in blocks of the inode array
- number of free blocks
- number of free inodes
- partial list of free inodes
  - An inode with di_mode == 0 is free.
  - When the partial list becomes empty the array of inodes is scanned to find more free inodes.
- partial list of free blocks
  - the first part of the list is in the superblock and the remaining in other blocks — it's not possible to inspect a block to see if it's in use or not.

s5fs Directory Structure

- Under UNIX directories are special (OS writable only) files.
- The directory file is an unsorted linked list (records) of filenames to file-inode (attributes and location of file on hard disk)



5	apples
4	oranges
5	aboli
2	.
7	..

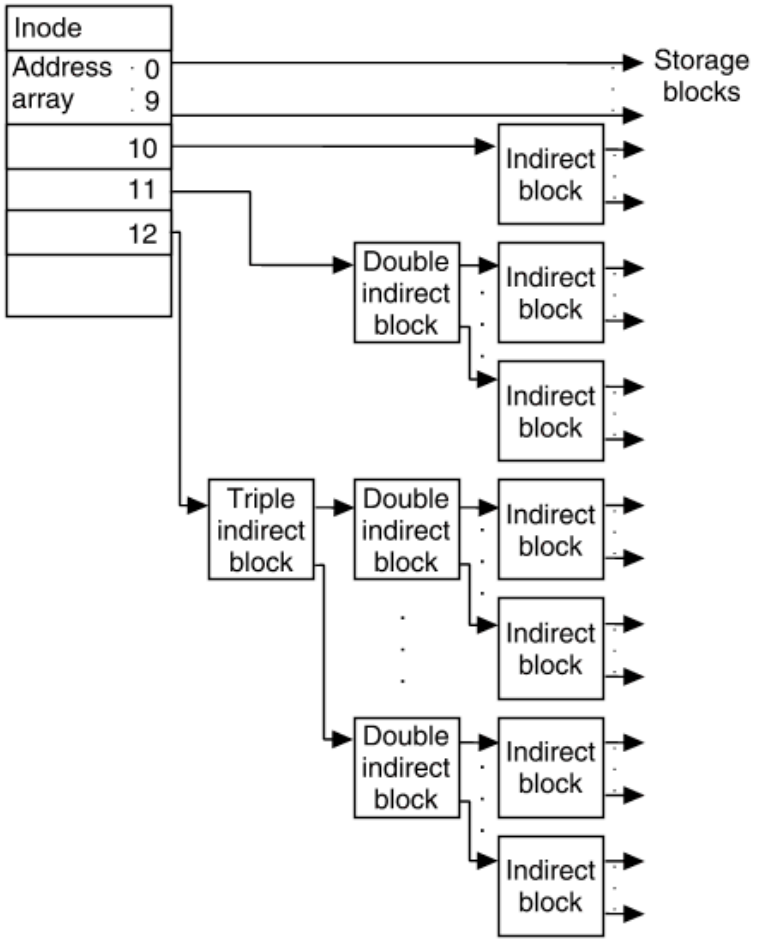
File Systems s5fs (Cont.1)

s5fs i-node Structure

File mode
Link count
Owner's id
Group id
File size
Last access time
Last mod time
Last inode access time
Addresses of first 10 blocks
Single indirect ptr
Double indirect ptr
Triple indirect ptr

- i-nodes contain a lot of file information in 64B (B - Byte)
  - mode: type (4 bits) and mode (12 bits) of file (2B)
  - number of links to the file (2B)
  - owner's UID (2B)
  - owners GID (2B)
  - number of bytes in file (4B)
  - 3 times (last accessed, modified, inode changed) (3x4B)
  - physical disk addresses (direct pointers) (10x3B)
  - physical disk addresses (indirect pointers) (3x3B)
  - generation (1B)
- An inode with mode = 0 is free
- MaxFileSize(forBlocksSize=512bytes) =(10+

s5fs File Blocks allocation



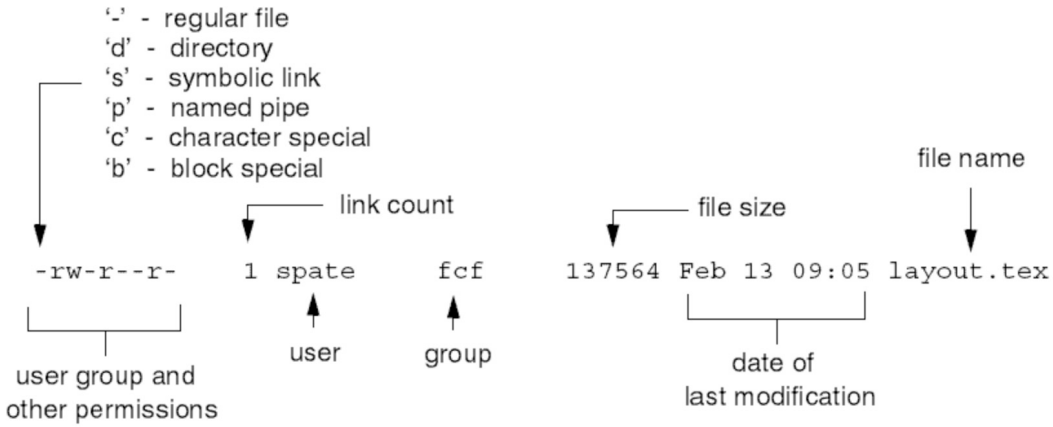
File Systems s5fs (Cont.2)

s5fs File Types

- File types&properties shown by typing ls -l
- Create Hard link and Symbolic link

  $ ln oldName newName

  $ ln -s oldName newName



Disadvantages s5fs

- Short file name <=14 symbols.
- Data blocks allocated randomly on all disk
- Inodes allocated randomly for catalogs
- Low of block seek speed (inode at start of disk, data blocks end).
- Max numbers of userID $<2^{16}=65536$
- Max numbers of files $< 2^{16} =65536$
- Max file size 1 by address length $1 < 2^{16}$ Bytes = 32 MiB.
- Max file size 2 by point < (10+256+256*256+256*256*256)*512 = =8623625216 Bytes = 8224 MiB.

- Analyse s5fs and ffs
  - s5fs use only 4% of disk bandwidth.
  - Read throughput increased from 29KB/s on s5fs to 221 KB/s on FFS.
  - Write throughput increased from 48KB/s on s5fs to 142 KB/s on FFS.
  - Optimization results of ffs 10-20 times vs original s5fs speeds!

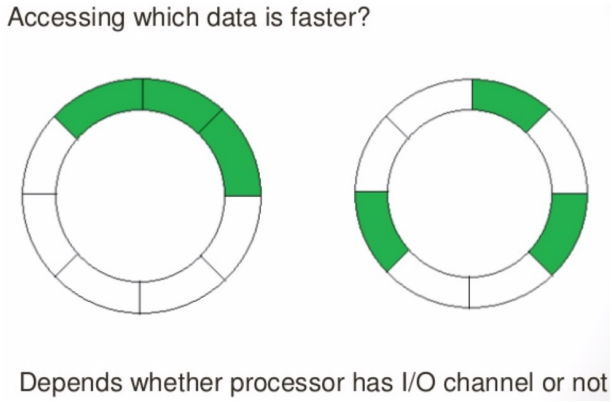
File Systems ffs and ufs

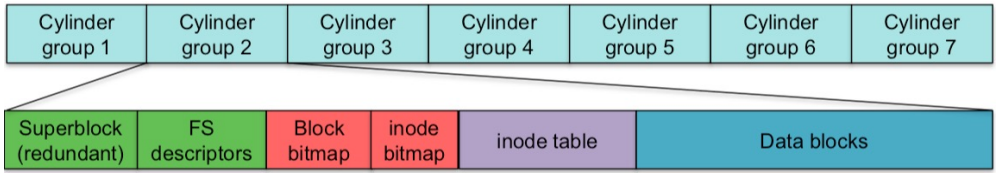
ffs - Fast File System, 4.2BSD, 1983

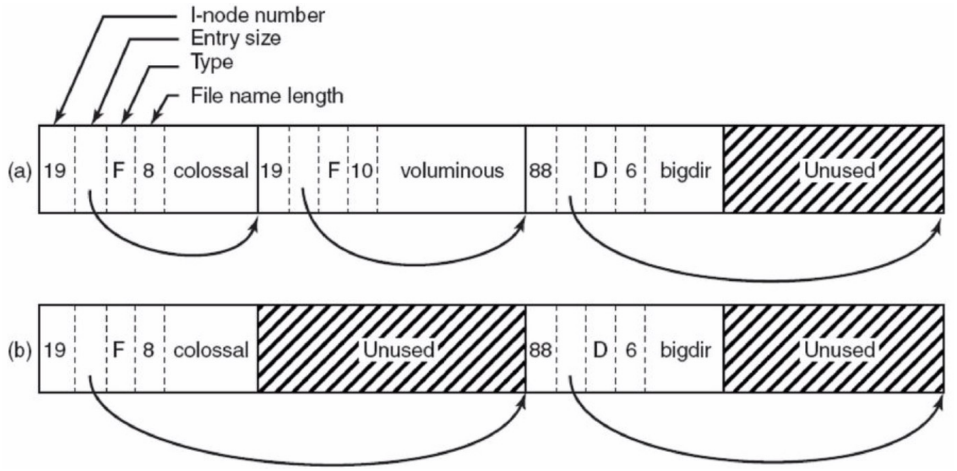
ufs – UNIX File System, 4.4BSD, 1984

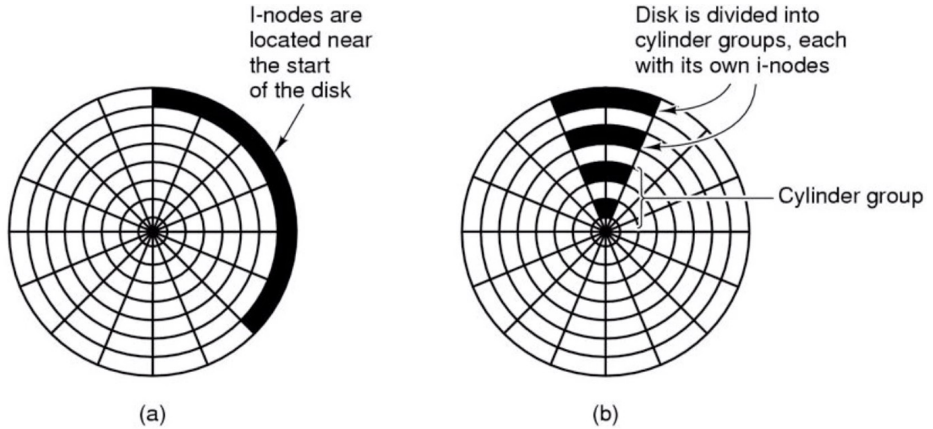
Optimizations:

- ffs divides a partition into a number of cylinder groups.
- Super Block is split in two parts: with FS Parameters and with Status of a Cylinder Group (CG).
- FS Parameters duplicated in each CG for reliability.
- Long file names realized (up to 255 characters) over Linked List directory structure (before and after file delete) →
- ffs allocated all files in a directory in one CG. To do so, use a different CG for a newly created directory.
- ffs allocated both inode and data blocks of a file in one CG for decrease time **seek:** →
- ffs allocated new blocks of a file are to reduce rotate **wait:**









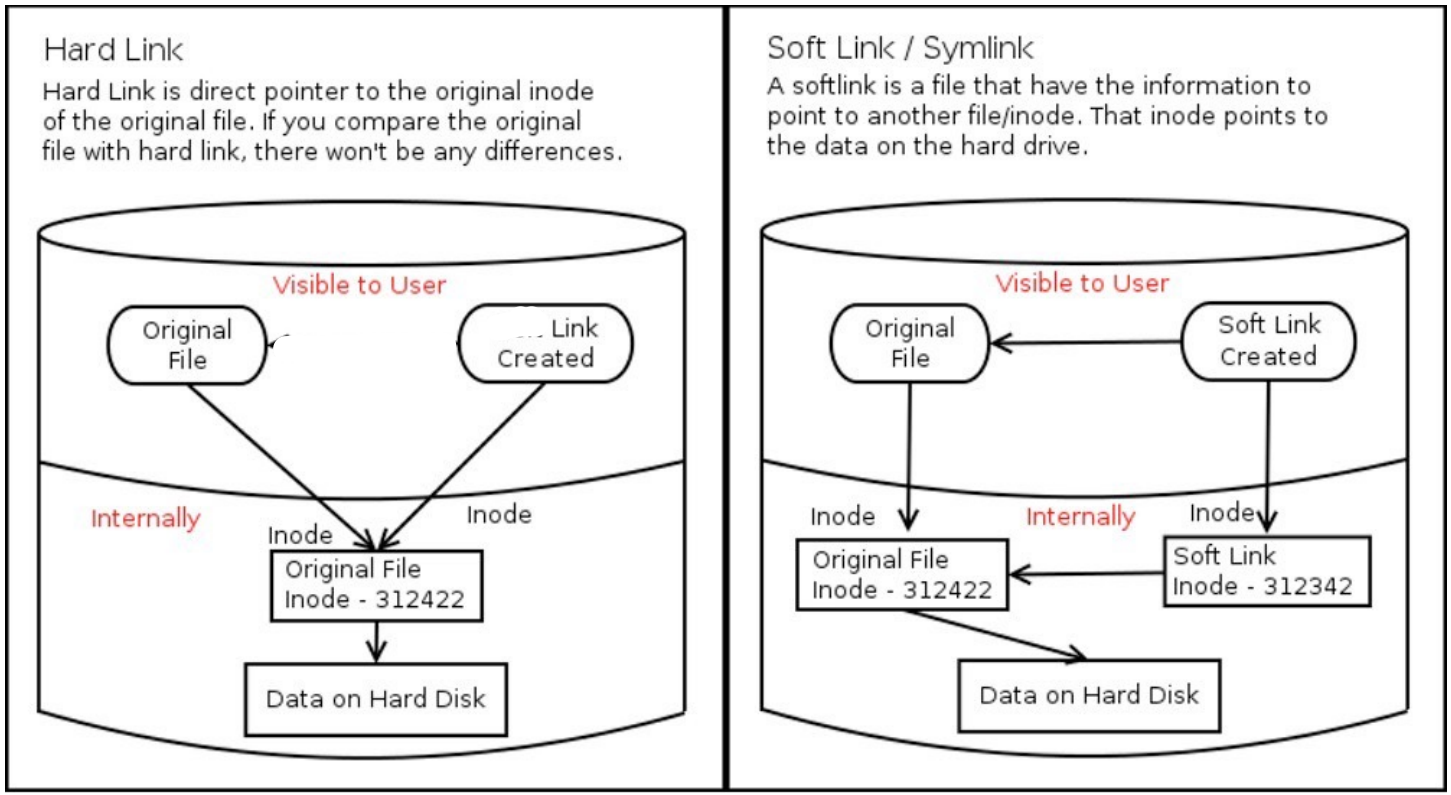
File Systems ffs and ufs (Cont.1)

New Features:

- Larger cluster support (0.5-8KiB). Cluster is divided into a number of fragments (2-8) for write up to 8 small files (minimizing inside fragmentation).
- FFS metadata writes are synchronous for anti-crash. After crash used fsck command.
- File Locking without deadlock detection.
- Quota sub-system add.
- Symbolic (Soft) links add.

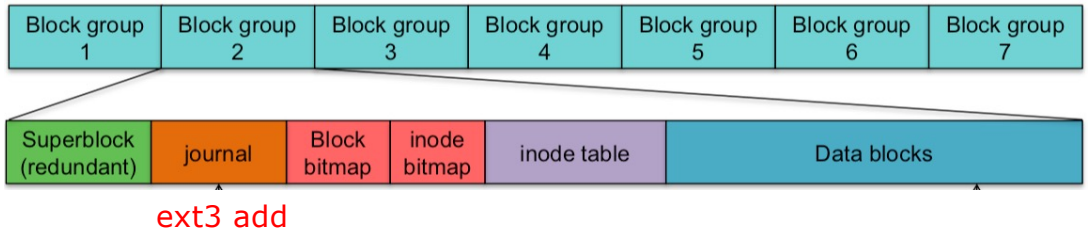
Hard and Symbolic links creation:
$ ln oldName newName
$ ln -s oldName newName

Hard links vs Symbolic links

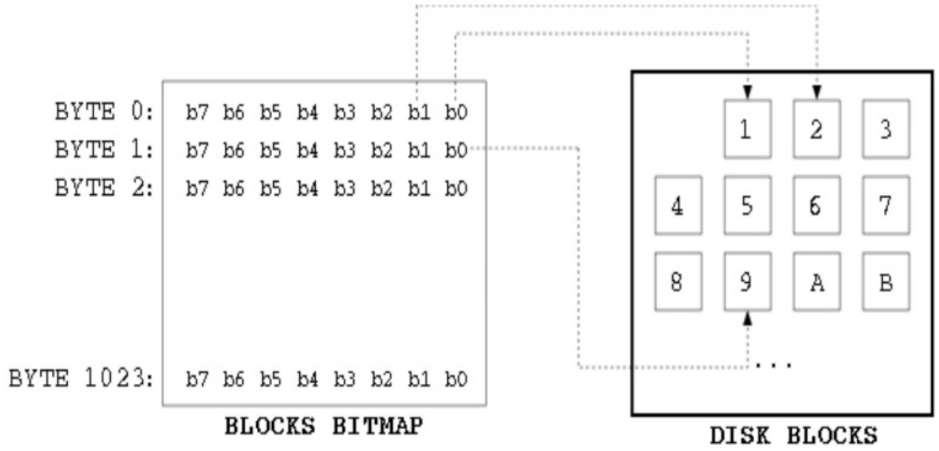


File Systems extfs, ext2fs

- extfs – Extended FS, 1992
- ext2fs – Extended Second FS, 1993
  - Layout of ext2 partition and Block group
  - Group Description
  - Bitmaps 1Block (1Block=512Byte=4096bit)
  - Inode Description



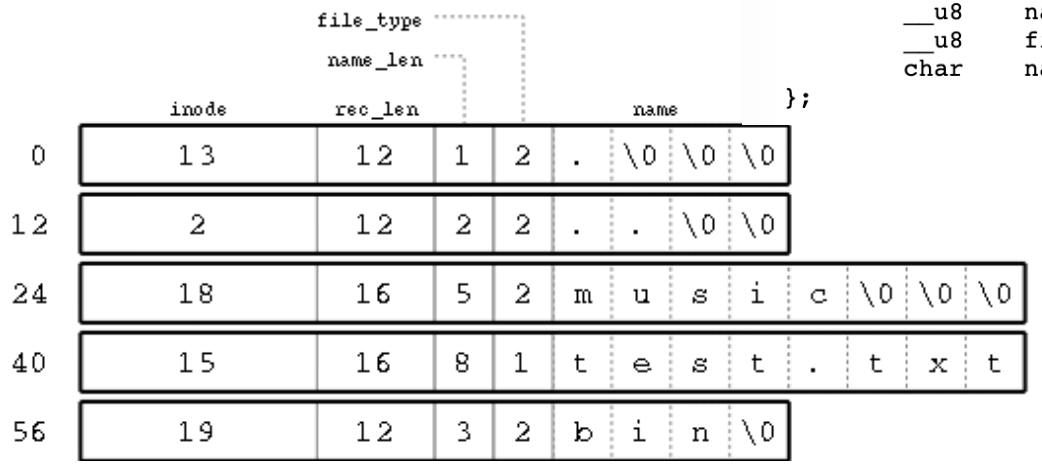

```
struct ext2_group_desc
{
        __u32   bg_block_bitmap;        /* Blocks bitmap block */
        __u32   bg_inode_bitmap;        /* Inodes bitmap block */
        __u32   bg_inode_table;         /* Inodes table block */
        __u16   bg_free_blocks_count;   /* Free blocks count */
        __u16   bg_free_inodes_count;   /* Free inodes count */
        __u16   bg_used_dirs_count;     /* Directories count */
        __u16   bg_pad;
        __u32   bg_reserved[3];

struct ext2_inode {
        __u16   i_mode;         /* File type and access rights */
        __u16   i_uid;          /* Low 16 bits of Owner Uid */
        __u32   i_size;         /* Size in bytes */
        __u32   i_atime;        /* Access time */
        __u32   i_ctime;        /* Creation time */
        __u32   i_mtime;        /* Modification time */
        __u32   i_dtime;        /* Deletion Time */
        __u16   i_gid;          /* Low 16 bits of Group Id */
        __u16   i_links_count;  /* Links count */
        __u32   i_blocks;       /* Blocks count */
        __u32   i_flags;        /* File flags */
        ...
        __u32   i_block[EXT2_N_BLOCKS];  /* Pointers to blocks */
        ...
```



File Systems ext2fs (Cont.1)

- Directory Structure ext2fs

```
struct ext2_dir_entry_2 {
        __u32   inode;                  /* Inode number */
        __u16   rec_len;                /* Directory entry length */
        __u8    name_len;               /* Name length */
        __u8    file_type;
        char    name[EXT2_NAME_LEN];    /* File name */
};
```



.

..

music/

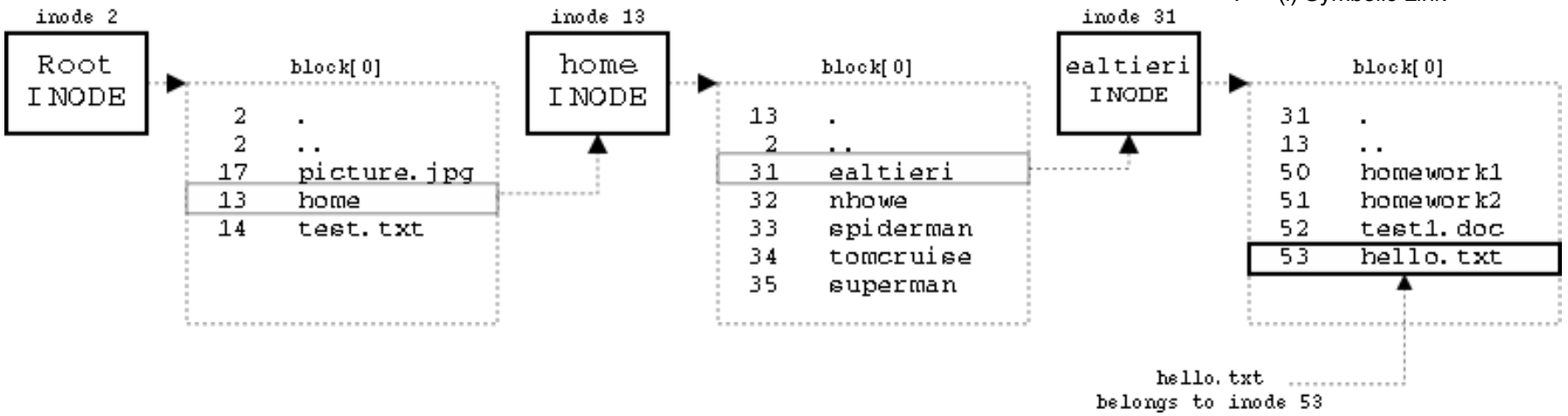
test.txt

bin/

FileType

0	Unknown
1	(-) Regular File
2	(d) Directory
3	(c) Character Device
4	(b) Block Device
5	(p) Named pipe
6	(s) Socket
7	(l) Symbolic Link

- Locating a file **/home/ealtieri/hello.txt**

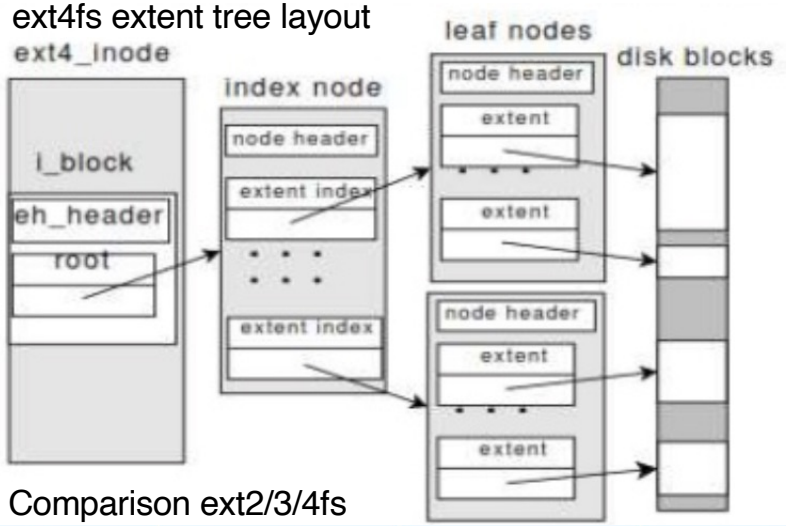


File Systems ex3tfs, ext4fs

- ext3fs – Extended Third FS +journaling , 2001
- Disadvantages of extfs, ext2fs, ext3fs:
  - - block allocation
  - - max blocks numbers= 2^32
  - - max fs size= 32TB
  - - 32 bit inode max time 18.01.2038
  - - max 31998 subdirectories

- ext4fs – Extended Fourth FS, 2008
- Advantages of ext4fs:
  - + max block numbers= 2^48
  - + max fs size=1EB=2^10PB=2^20TB
  - + unlimited subdirectory numbers
  - + extents mapping on double and triple indirect
  - + inode size= 256 byte →
  - + nanoseconds
  - + inode versions
  - + extend attributes
- Future for Linux FS → BtrFS

- ext4fs extent tree layout



- Comparison ext2/3/4fs

Point	ext2	ext3	ext4
Maximum individual file size	16GB – 2TB	16GB – 2TB	16GB – 16TB
Maximum file system size	2TB – 32TB	2TB – 32TB	1EB
Journalling	Not available	Available	Available and can be turned “off” too
Number of directories	31998	31998	Unlimited
Journal checksum	No	No	Yes
Multi-block allocation and delayed allocation	No	No	Yes

Linux File Systems Comparison

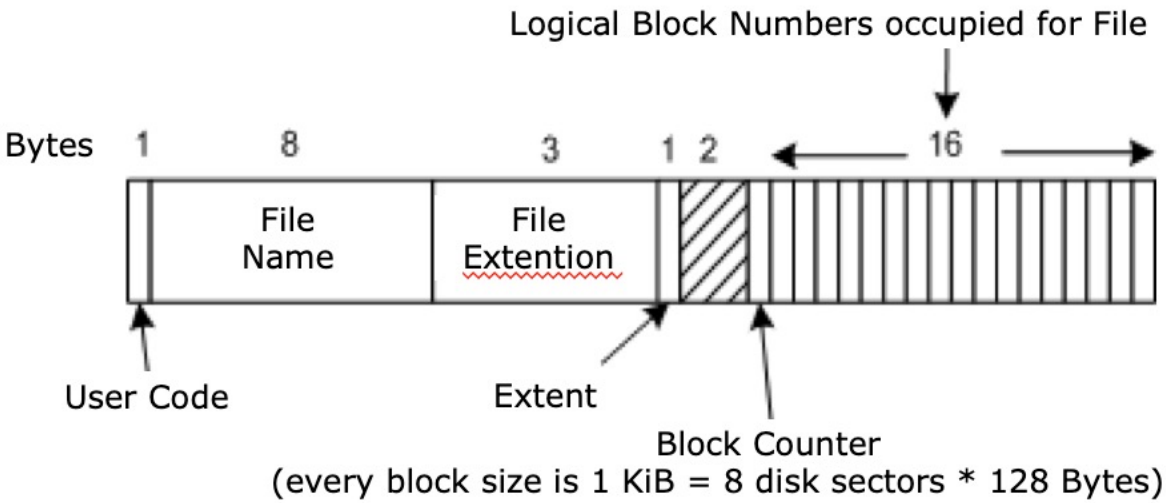
- Comparison of file systems https://en.wikipedia.org/wiki/Comparison_of_file_systems

Feature	EXT4	XFS	BTRFS
Architecture	Hashed B-tree	B+ tree	Extent based
Introduced	2006	1994	2009
Max volume size	1 Ebytes	8 Ebytes	16 Ebytes
Max file size	16 Tbytes	8 Ebytes	16 Ebytes
Max number of files	4 billion	2^{64}	2^{64}
Max file name size	255 bytes	255 bytes	255 bytes
Attributes	Yes	Yes	Yes
Transparent compression	No	No	Yes
Transparent encryption	Yes	No	Planned
Copy-on-Write (COW)	No	Planned	Yes
Snapshots	No	Planned	Yes

File Systems CP/M

CP/M FS - Control Program for Microcomputers.

- Operating system CP/M from Digital Research (1977) - is the predecessor of IBM/MS-DOS.
- CP/M FS have only one directory, with 32 bytes records.
- File Name Template: 8 + 3 uppercase characters.
- Bitmap of occupied/free blocks is calculated after each reboot and save only in RAM (for 180KiB disk need only 23 bytes array). After shutdown, it is not written to disk.
- Maximum file size 16KB (16 block numbers * 1KB).
- For files up to 32 Kbytes, two records can be used, for up to 48 Kbytes three records, etc.
- The sequence number of the entry is stored in the extent field.
- The user code protects files - the user only works with his files.
- Directory Record Structure:



File Systems FAT-12/16/32/64(exFAT)

Advantages:

- These file systems don't include a journal, so they're ideal for external USB drives.
- They're a de facto standard that every operating system—Windows, macOS, Linux, and other devices—can read. This makes them the ideal file system to use an external drive for different systems.
- FAT16 max file size = Max Cluster Numbers * Max Cluster Size=2^{16}*2^{6}KiB=2^{22}KiB=4GiB.
- FAT32 is older then exFAT.
- FAT64 (or exFAT) (developed Microsoft 2005) - file system optimized for flash devices, is the ideal option, as it supports files over 4 GB in size and partitions over 8 TB in size, unlike FAT32.
- Since 08/2019 – start integration exFAT to Linux Kernel.

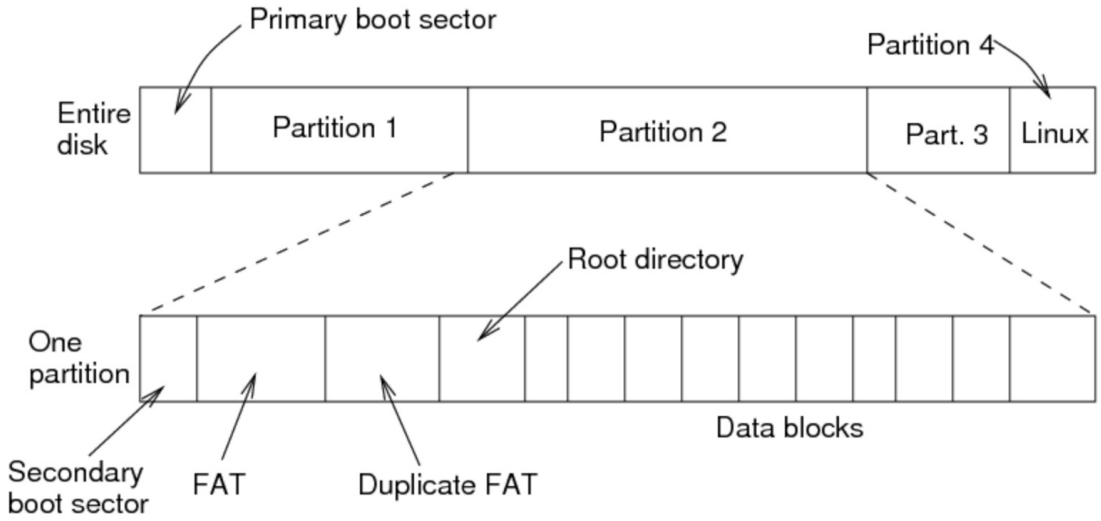
Feature	FAT	FAT32	exFAT	NTFS	ReFS
Maximum volume size	4 GB	32 GB	128 PB	256 TB	4.7 ZB (zettabytes)
Maximum file size	4 GB	4 GB	16 EB (exabytes)	18 EB (exabytes)	18 EB (exabytes)
Maximum filename length	8.3 characters	255 characters	255 characters	255 characters	255 characters
Maximum cluster size	64 KB	32 KB	32 MB	2048 KB	64 KB
File compression	No	No	No	Yes	No
File encryption	No	No	No	Yes	No
Permissions	No	No	No	Yes	Yes

Disadvantages:

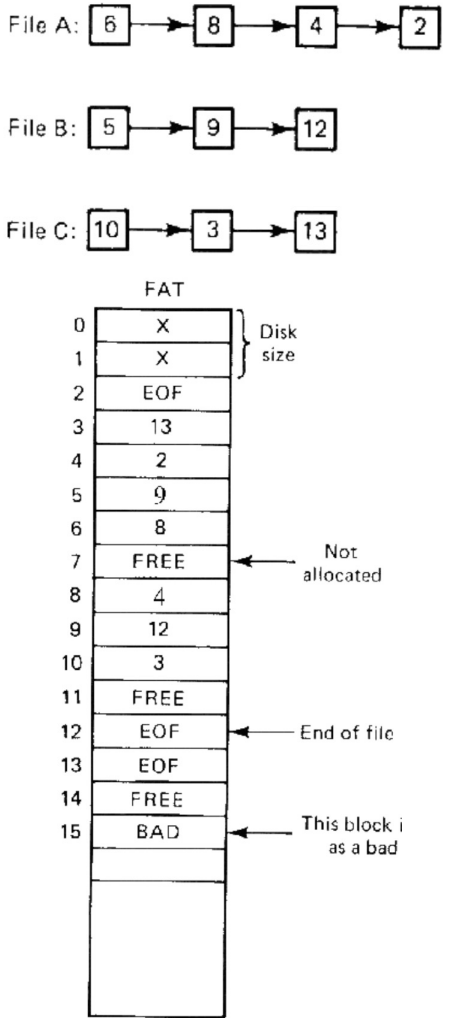
- FAT file systems don't include a permission attributes (no local security, have only network security).
- FAT-16/32 cannot store a file larger than 4GB.
- For partition more then 200 MiB performance with FAT 16/32 will quickly decrease.
- Less reliability

File Systems FAT-12/16

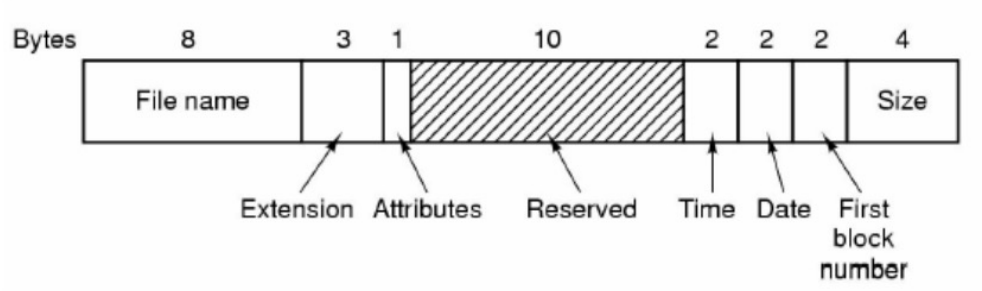
- **FAT FS Structure**



- **FAT Structure example**



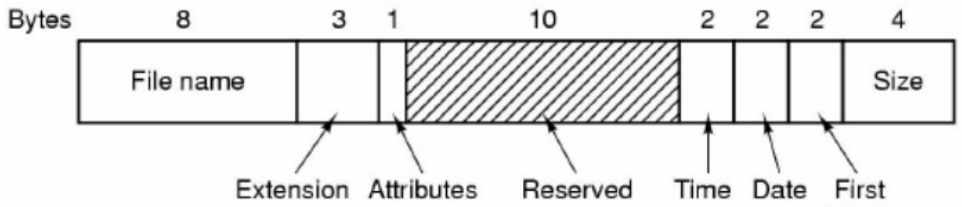
- **FAT FS Directory Structure (MS-DOS FAT-12,16)**

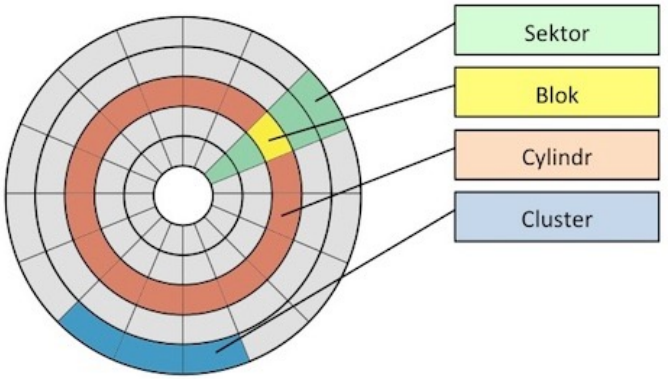


Max File Size =

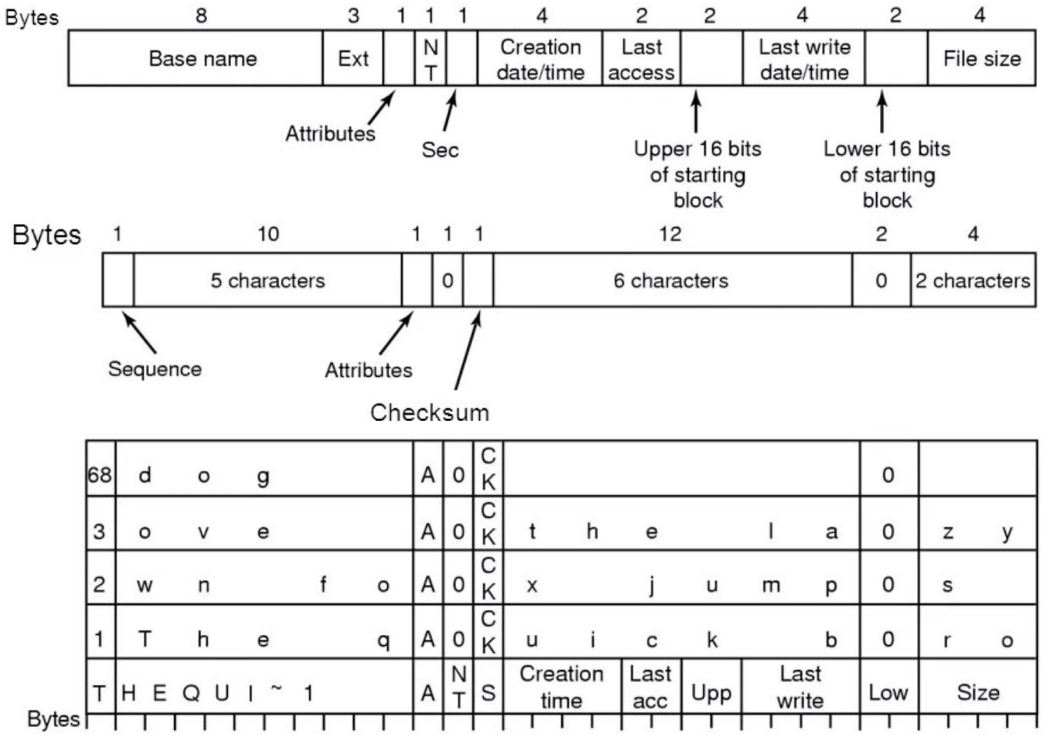
File Systems FAT-32

- **FAT FS Directory Structure (MS-DOS FAT-12,16)**





- **Windows 98 (extFAT-32) Directory Structure (long Name support)**

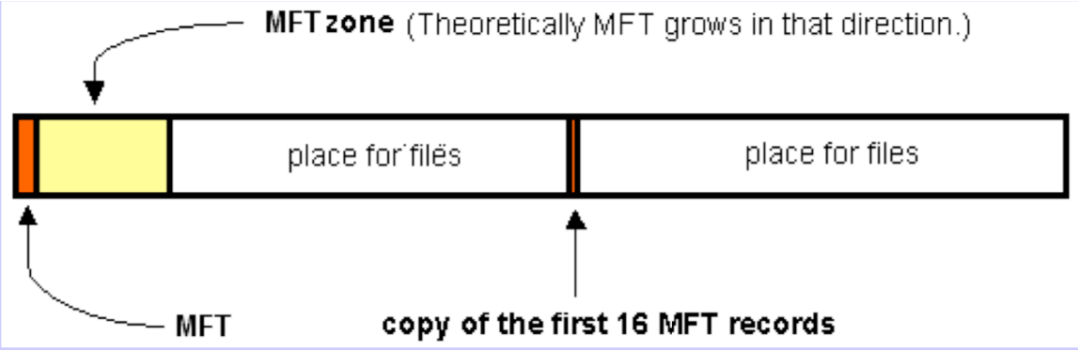


FAT Cluster Sizes by Volume Size

Drive Size (Range)	Cluster Size
Less than 16MB	4096 bytes (4KB)
16MB to 128MB	2048 bytes (2KB)
128MB to 256MB	4096 bytes (4KB)
256MB to 512MB	8192 bytes (8KB)
512MB to 1024MB	16,384 bytes (16KB)
1024MB to 2048MB	32,768 bytes (32KB)

File Systems ntfs

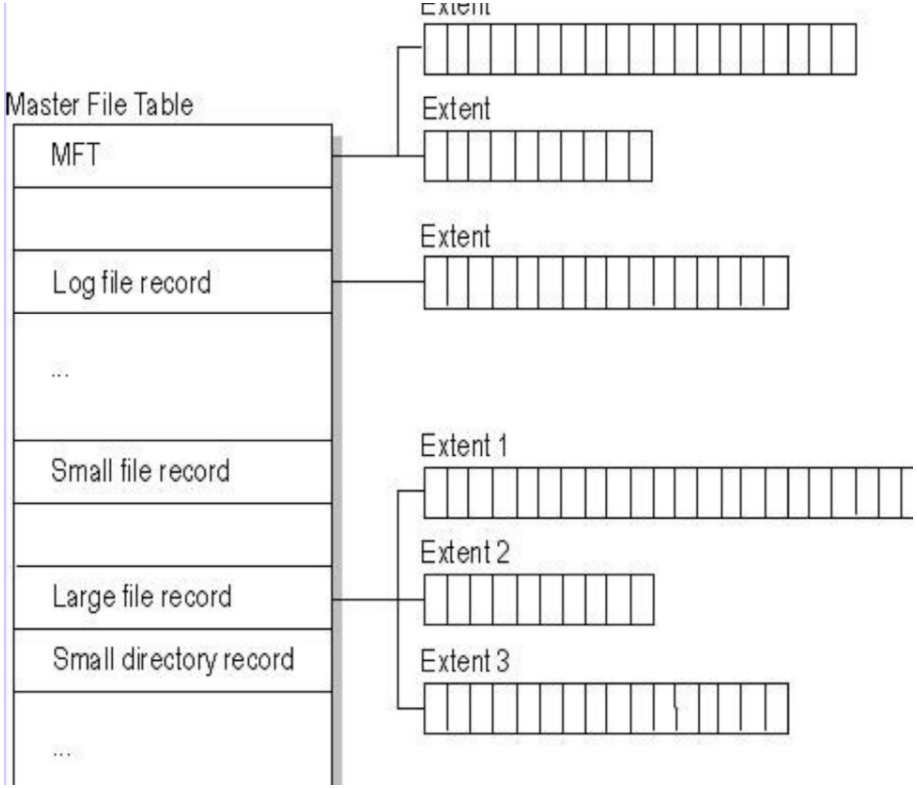
- ntfs MFT - Master File Table
- Each File has an entry in the Master File table.
- The first entry describes the MFT itself.
- The following are log or Update Sequence Number options.
- The structure can be used NTFSInfo →
- (https://docs.microsoft.com/en-us/sysinternals/downloads/ntfsinfo)




```
C:\Users\ys\Downloads\NTFSInfo>ntfsinfo.exe c:

NtfsInfo v1.2 - NTFS Information Dump
Copyright (C) 2005-2016 Mark Russinovich
Sysinternals - www.sysinternals.com


Volume Size
-----------
Volume size               : 99899 MB
Total sectors             : 204595199
Total clusters            : 25574399
Free clusters             : 1624934
Free space                : 6347 MB (6% of drive)

Allocation Size
---------------
Bytes per sector          : 512
Bytes per cluster         : 4096
Bytes per MFT record      : 0
Clusters per MFT record: 0

MFT Information
---------------
MFT size                  : 667 MB (0% of drive)
MFT start cluster         : 786432
MFT zone clusters         : 23527648 - 23578560
MFT zone size             : 198 MB (0% of drive)
MFT mirror start          : 2

Meta-Data files
---------------

C:\Users\ys\Downloads\NTFSInfo>_
```

File Systems ntfs (Cont.1)

- MFT Structure

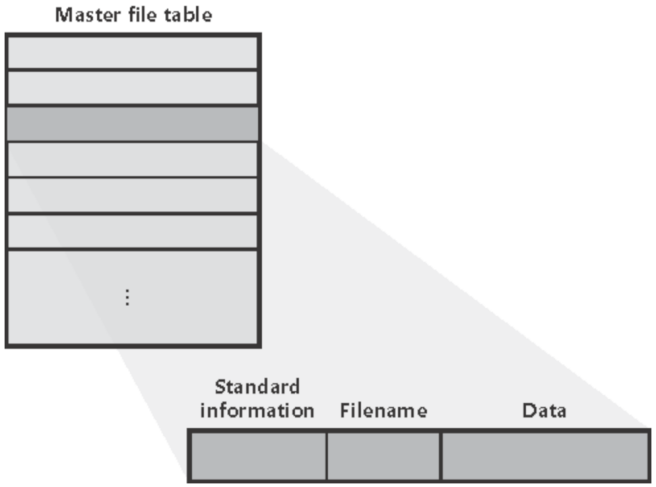


- MFT Metafiles

File		
0	$Mft - MFT	Reserved for NTFS metadata files
1	$MftMirr - MFT mirror	
2	$LogFile - Log file	
3	$Volume - Volume file	
4	$AttrDef - Attribute definition table	
5	\ - Root directory	
6	$Bitmap - Volume cluster allocation file	
7	$Boot - Boot sector	
8	$BadClus - Bad-cluster file	
9	$Secure - Security settings file	
10	$UpCase - Uppercase character mapping	
11	$Extend - Extended metadata directory	
12	Unused	
15	Unused	
16	User files and directories	

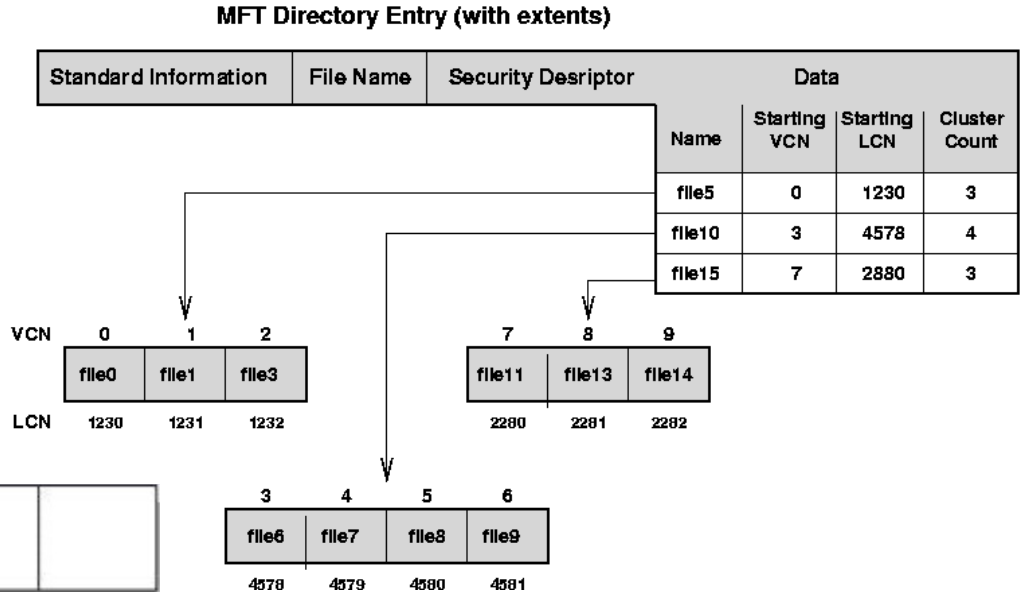
File Systems ntfs (Cont.2)

- MFT Record



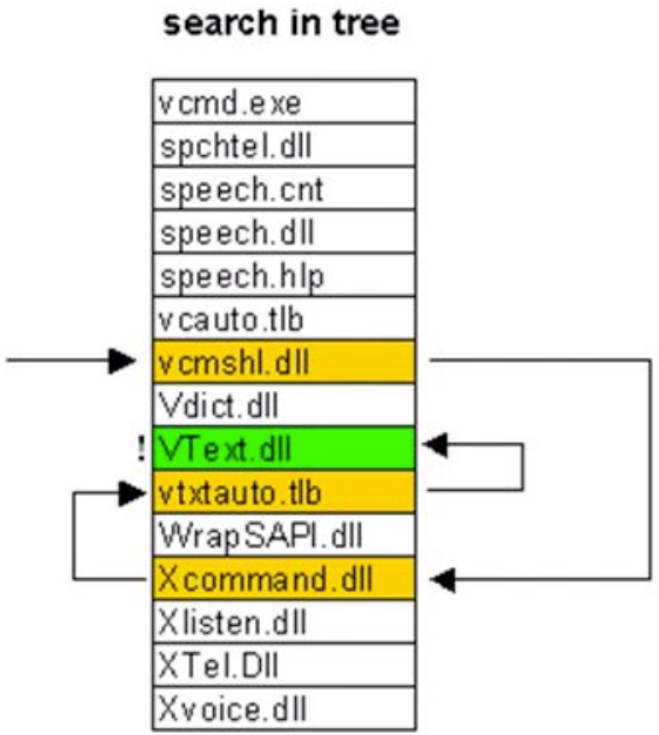
- MFT Record for a Small File or Directory

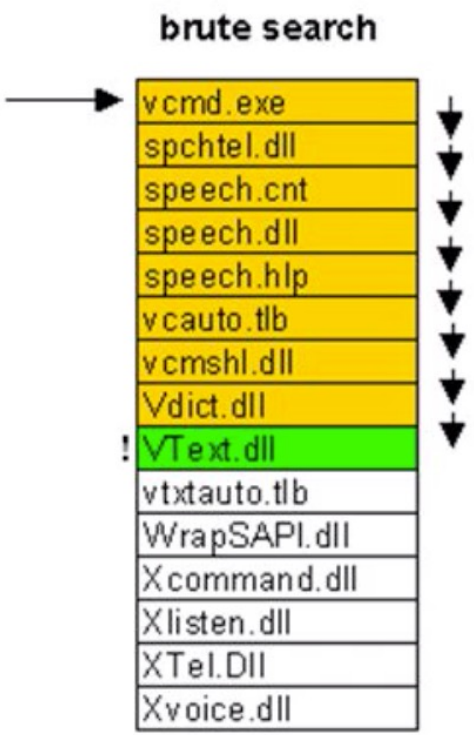
- MFT Record for a Big File or Directory (extents)
  - VCN - Virtual Cluster Number,
  - LCN - Logical Cluster Number,
  - Extent – (LCNstart, Lenght).



File Systems ntfs (Cont.3)

- NTFS vs FAT File Search Speed





$avgN_{NumberSeek}=LOG_2(M_{DirectorySize})$

$avgN_{NumberSeek}=LOG_2(10240)=13,3$

AvgSearchTime = 13,3*12ms=160ms=0,16sec

$avgN_{NumberSeek}=M_{DirectorySize}/2$

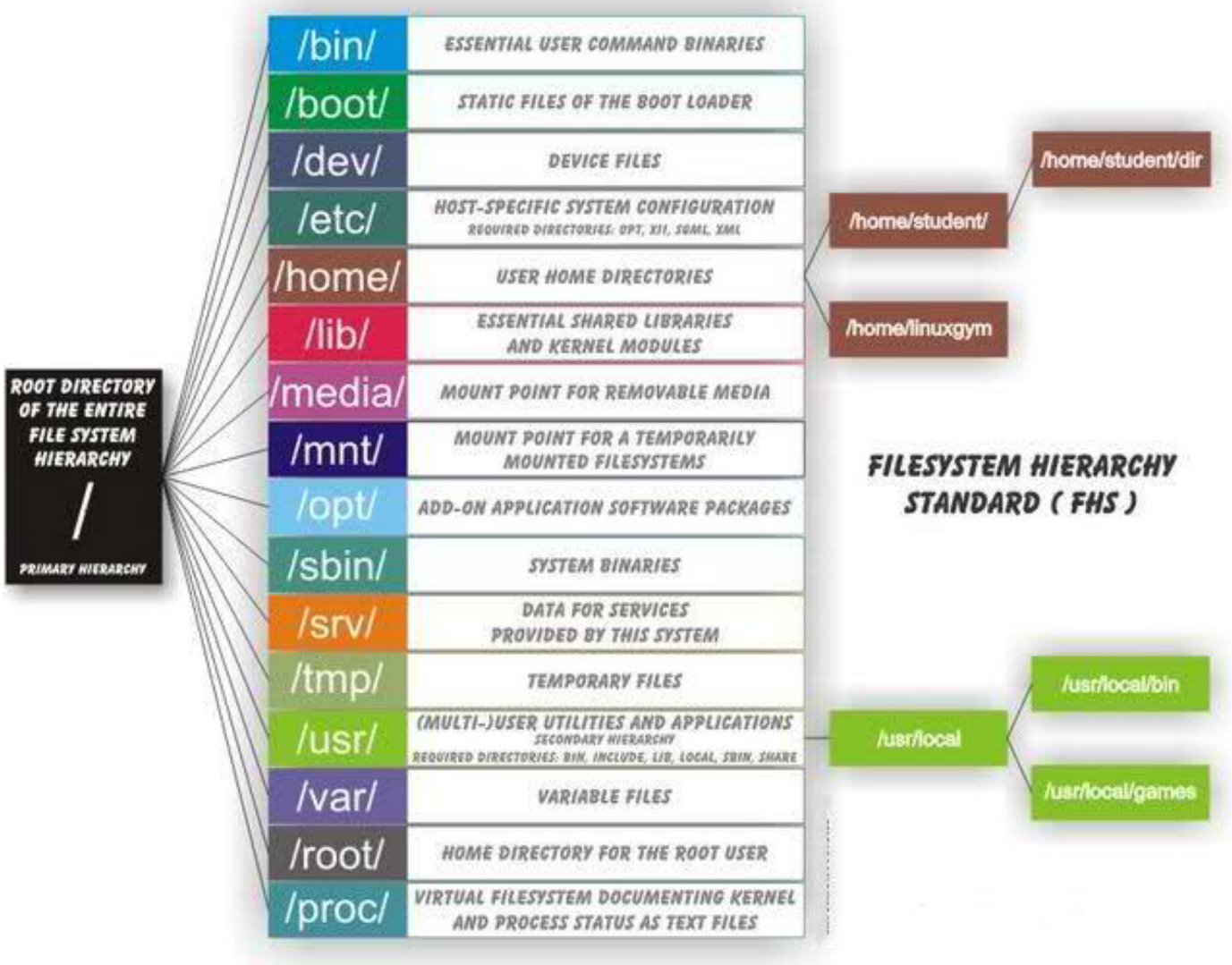
$avgN_{NumberSeek}=10240/2=5120$

AvgSearchTime = 5120*12ms=61440ms=60sec=1min

LINUX FSH

Linux File System Hierarchy (FSH)
Linux File Systems Commands
Linux VFS
Linux Pseudo FS

Linux File System Hierarchy (FSH) Standard



Attention! FSH knowledge is very Important Information!

Linux FSH Standard (Cont.1)

- **TASK. Use a (cd+ls+tree+cat+more) or mc for exploring FSH on your Linux.**
- Describing briefly the purpose of each **first level directory**
  - /bin : All the executable binary programs (file) required during booting, repairing, files required to run into single-user-mode, and other important, basic commands viz., cat, du, df, tar, rpm, wc, history, etc.
  - /boot : Holds important files during boot-up process, including Linux Kernel.
  - /dev : Contains device files for all the hardware devices on the machine e.g., cdrom, cpu, etc
  - /etc : Contains Application's configuration files, startup, shutdown, start, stop script for every individual program.
  - /home : Home directory of the users. Every time a new user is created, a directory in the name of user is created within home directory which contains other directories like Desktop, Downloads, Documents, etc. (according with /etc/skeleton configuration)
  - /lib : The Lib directory contains kernel modules and shared library images required to boot the system and run commands in root file system.
  - /lost+found : This Directory is installed during installation of Linux, useful for recovering files which may be broken due to unexpected shut-down.
  - /media : Temporary mount directory is created for removable devices viz., media/cdrom.
  - /mnt : Temporary mount directory for mounting file system.
  - /opt : Optional is abbreviated as opt. Contains third party application software. Viz., Java, etc.
  - /proc : A virtual and pseudo file-system which contains information about running process with a particular Process-id aka pid.
  - /root : This is the home directory of root user and should never be confused with '/'
  - /run : This directory is the only clean solution for early-runtime-dir problem.
  - /sbin : Contains binary executable programs, required by System Administrator, for Maintenance. Viz., iptables, fdisk, ifconfig, swapon, reboot, etc.
  - /srv : Service is abbreviated as 'srv'. This directory contains server specific and service related files.
  - /sys : Modern Linux distrib's include a /sys directory as a virtual filesystem, which stores and allows modification of the devices connected to the system.
  - /tmp :System's Temporary Directory, Accessible by users and root. Stores temporary files for user and system, till next boot.
  - /usr : Contains executable binaries, documentation, source code, libraries for second level program.
  - /var : Stands for variable. The contents of this file is expected to grow. This directory contains log, lock, spool, mail and temp files.

Linux FSH Standard (Cont.2)

- Linux is a complex system which requires a more complex and efficient way to start, stop, maintain and reboot a system. There is a well **defined special files**: configuration files, binaries, man pages, info files, etc. for every process in Linux.
  - /boot/vmlinuz : The Linux Kernel file.
  - /dev/hda : Device file for the first IDE HDD (Hard Disk Drive)
  - /dev/null : A pseudo device, that don't exist. Sometime garbage output is redirected to /dev/null, so that it gets lost, forever.
  - /etc/bashrc : Contains system defaults and aliases used by bash shell.
  - /etc/crontab : A shell script to run specified commands on a predefined time Interval.
  - /etc/exports : Information of the file system available on network.
  - /etc/fstab : Information of Disk Drive and their mount point.
  - /etc/group : Information of Security Group.
  - /etc/grub.conf : grub bootloader configuration file.
  - /etc/init.d : Service startup Script.
  - /etc/lilo.conf : lilo bootloader configuration file.
  - /etc/hosts : Information of Ip addresses and corresponding host names.
  - /etc/hosts.allow : List of hosts allowed to access services on the local machine.
  - /etc/host.deny : List of hosts denied to access services on the local machine.
  - /etc/inittab : INIT process and their interaction at various run level.
  - /etc/issue : Allows to edit the pre-login message.
  - /etc/modules.conf : Configuration files for system modules.
  - /etc/motd : motd stands for Message Of The Day, The Message users gets upon login.
  - /etc/mtab : Currently mounted blocks information.
  - /etc/passwd : Contains password of system users in a shadow file, a security implementation.
  - /etc/printcap : Printer Information

Linux FSH Standard (Cont.3)

- Linux is a complex system which requires a more complex and efficient way to start, stop, maintain and reboot a system. There is a well **defined special files**: configuration files, binaries, man pages, info files, etc. for every process in Linux.
  - /etc/profile : Bash shell defaults
  - /etc/profile.d : Application script, executed after login.
  - /etc/rc.d : Information about run level specific script.
  - /etc/rc.d/init.d : Run Level Initialisation Script.
  - /etc/resolv.conf : Domain Name Servers (DNS) being used by System.
  - /etc/securetty : Terminal List, where root login is possible.
  - /etc/skel : Script that populates new user home directory.
  - /etc/termcap : An ASCII file that defines the behaviour of Terminal, console and printers.
  - /etc/X11 : Configuration files of X-window System.
  - /usr/bin : Normal user executable commands.
  - /usr/bin/X11 : Binaries of X windows System.
  - /usr/include : Contains include files used by ‘c‘ program.
  - /usr/share : Shared directories of man files, info files, etc.
  - /usr/lib : Library files which are required during program compilation.
  - /usr/sbin : Commands for Super User, for System Administration.
  - /version : Linux Version Information.
  - /var/log/lastlog : log of last boot process.
  - /var/log/messages : log of messages produced by syslog daemon at boot.
  - /var/log/wtmp : list login time and duration of each user on the system currently.
  -

Linux File Systems Commands

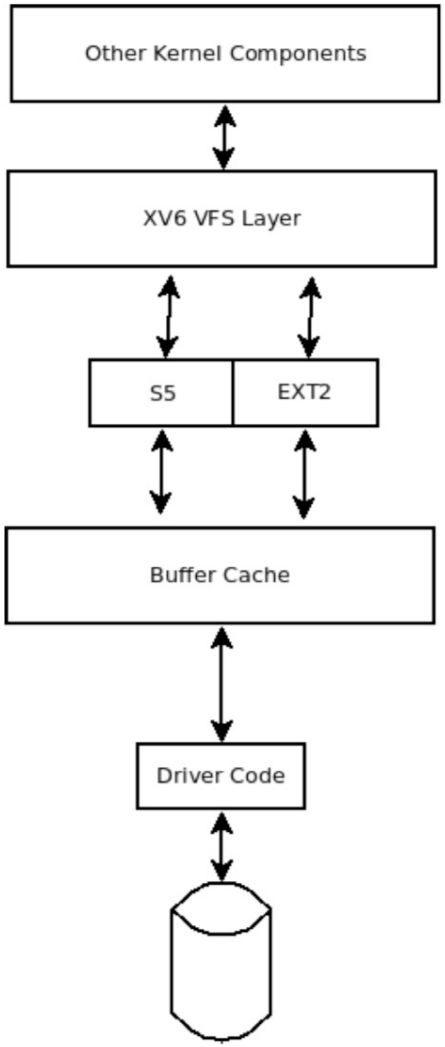
- File commands
  - pwd
  - mkdir - rmdir
  - chdir
  - ls
  - file
  - touch
  - find
  - ln
  - cat
  - rename
  - rm
  - mv
  - link - unlink
  - chown
  - chgrp
  - chmod
  - chattr
  - lsattr
  - umask
- File system commands
  - dd
  - du - df
  - mount
  - umount
  - cat /etc/fstab
  - cat /etc/mtab
  - stat /etc/passwd – info about inode
  - stat -f /etc/passwd – info about fs
  - fsck – fs check
  - fdisk – create, resize, test, list the drive partitions
  - partx, parted – create, resize, test, list the drive partitions
  - mke2fs /dev/hdb2 [-b 1024|...|4096] – make file system
  - tune2fs – fs reconfiguration and info about fs (some pages)
  - dump2fs – info about fs (some pages)
- Test on your Linux as root (use sudo -i before)
  - # fdisk -l or # parted -l
  - # stat -f /tmp
  - # tune2fs -l /dev/sda1 or # /sbin/dump2fs -h /dev/sda1
- System commands
  - clear
  - history
  - date
  - passwd
  - exit
  - reboot
  - poweroff
  - man
  - ps
  - pstree
  - kill
  - su
  - sudo
  - lsusb
  - lsmod
  - modinfo
  - rmmod
  - modprobe
  - udevadm

Linux VFS

- VFS since UNIX ver. IV.
- First implemented SunOS
- Support all fs: ffs, ufs, ext2fs, ...

- Early UNIX systems could only mount one file system type.
- An object oriented approach is now used UNIX systems.
- A vnode (virtual node) represents a file in the kernel.
- A vfs (virtual file system) represents a file system in the kernel. The vnode (virtual node) struct contains file-system indepependent attributes and two pointers:
  - first pointer to file-system specific inode information;
  - second pointer to file-system specific function pointers (shared by each inode of that file system type).

- The VFS layer is added to the UNIX kernel to allow applications to access different types of FS's in a uniform way.



Linux Pseudo FS's

Pseudo FS - files do not exist on disk; they are virtual, fake files that the kernel creates dynamically in memory:

- procfs,
- udev (sysfs, devfs),
- debugfs,
- usbfs,
- swap,
- nfs,
- sockfs,
- fifofs,
- autofs.

	File System	Type	Device	Description
1	ufs	regular	Disk	UNIX Fast File System
2	tmpfs	regular	Memory	Uses Memory and Swap
3	nfs	pseudo	Network	Network File System
4	cachefs	pseudo	File System	Uses a local disk as a cache for other NFS
5	autofs	pseudo	File System	Uses a dynamic layout to mount other file systems
6	specfs	pseudo	Device Drives	File System for the /dev devices
7	procfs	pseudo	Kernel	/proc file system representing processes
8	sockfs	pseudo	Network	File system for socket connections
9	fifofs	pseudo	Files	FIFO file system

Linux Pseudo FS: procfs, sysfs, devfs, swap

prosfs - originally process filesystem

- mounted to /proc.
- each process gets a directory (named by the process id (pid)) under /proc
- /proc/PID directories you find a few files and links.
- /proc/sys/ - sub-tree allows you edit system information
- /proc example files:
  - cpuinfo - Information about CPUs
  - filesystems - Current FS supported by the kernel
  - interrupts – Info about system interrupts
  - meminfo - Info about system memory
  - partitions - Info about available system partitions
- procfs info use tools **top**, **ps** & sys-call sysctl(), ioctl()

Other commands & files were also included under /proc for providing system information: cpuinfo, meminfo, uptime, interrupts, mounts, lsof.

Manager udev (old HAL, dbus) event driven and support sysfs and devfs trees for mapping of real&pseudo devices.

sysfs - intended to provide tree with grouping information about devices state.

- mounted to /sys
- intended to provide tree with information ab. devices
- created for minimizing big tree of procfs
- one item per file and strict documentation rule

devfs - mapping of real $ pseudo hardware for work with devices.

- mounted to /dev
- include mem, null, zero, urandom, random, sda, sda1, tty, char/(null, zero), block/(sda, sda1)

swap - used as virtual memory and doesn't have a file system structure.

- You can't mount it to view its contents.
- Swap is used as "scratch space" by the Linux kernel to temporarily store data that can't fit in RAM.
- It's also used for hibernating.
- While Windows stores its paging file as a file on its main system partition, Linux just reserves a separate empty partition for swap space.

Linux Pseudo FS: Example 1. Exploring procfs

Example 1. Exploring procfs.

- Virtual File System procfs contained information about processes and other system information.
- procfs is mapped to /proc and mounted at boot time.
- Information about any files is available in the man page by running:

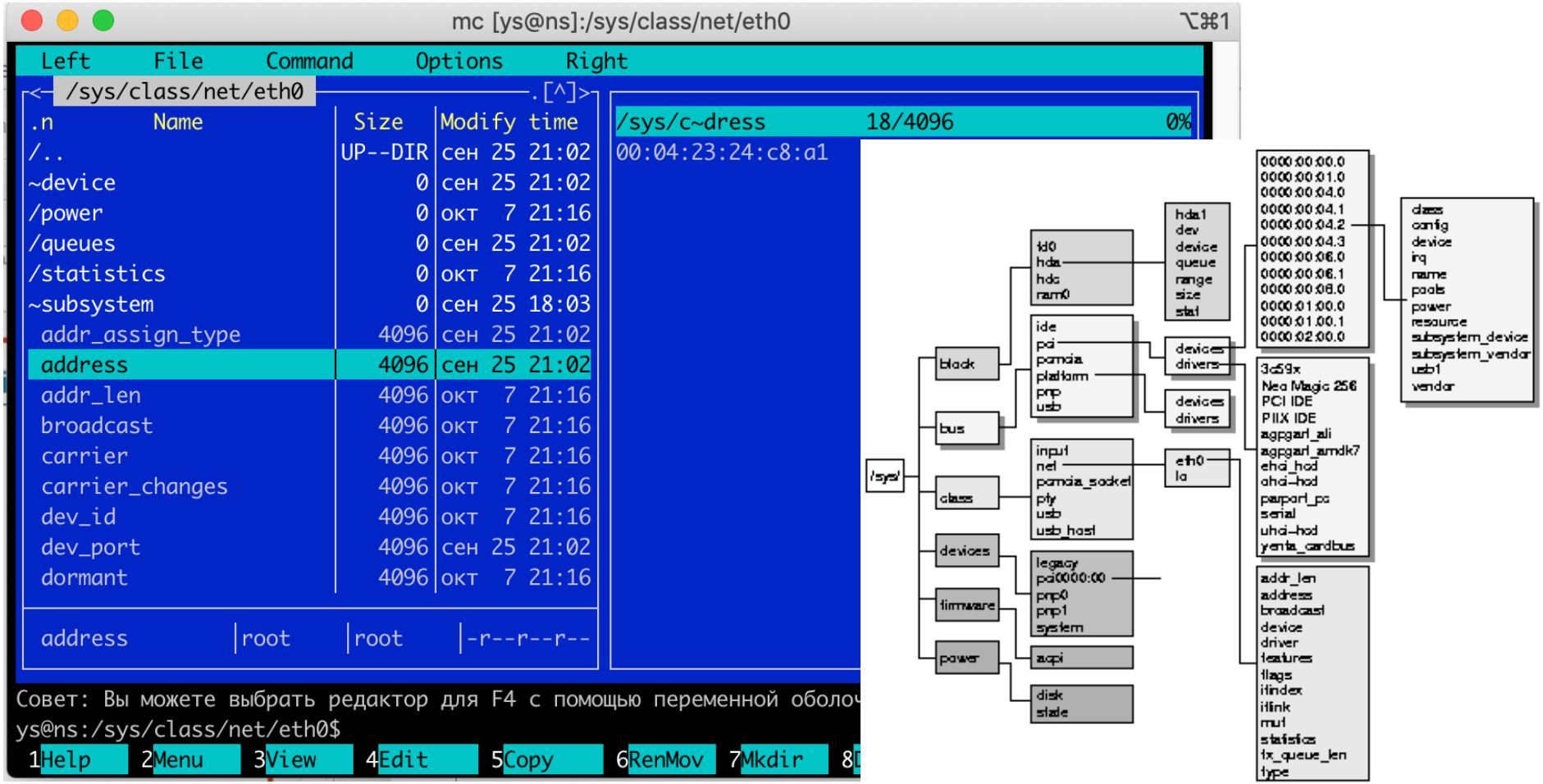
  $ man 5 proc (and after search /proc/<filename>)
- For read info from /proc files use or mc-viewer, or command cat:

  $ cat /proc/<filename>
- Quick info about /proc's files:
  - /proc/cmdline – Kernel command line information.
  - /proc/console – Information about current consoles including tty.
  - /proc/devices – Device drivers currently configured for the running kernel.
  - /proc/loadavg – System load average.
  - /proc/locks – Files currently locked by kernel.
  - /proc/modules – Currently loaded kernel modules.
  - /proc/mounts – List of all mounts in use by system.
  - /proc/pci – Information about every PCI device.
  - /proc/stat – Record or various statistics kept from last reboot.
  - /proc/swap – Information about swap space.
  - /proc/uptime – System uptime information (in seconds).
  - /proc/version – Kernel version, gcc version, and Linux distribution installed.
- /proc/PID interesting folders, files, links:
  - fd - file descriptors (0,1,2,…)
  - environ - environmental variables
  - cmdline – process command line
  - io – input-output statistics
  - limits – process limits
  - mounts – process related information
  - cwd – link to current work directory
  - exe – link to process executable file
  - root - link to process work directory

Linux Pseudo FS: Example 2. Exploring sysfs

Example 2. Exploring sysfs with mc-viewer (Use interesting directory and: <F9> → Right → Quick View)

/sys/class/net/eth0/address



Linux Pseudo FS: Example 3. Exploring devfs

Example 3. Exploring devfs with command line (Use: cd, ls, head, cat).

```
ys@ns:/dev$ cd /dev
ys@ns:/dev$ ls
agpgart          fd          loop4             ppp      snapshot  tty16  tty3   tty43  tty57      ttyS11  ttyS25  uinput   vcsa5
autofs           full        loop5             psaux    snd       tty17  tty30  tty44  tty58      ttyS12  ttyS26  urandom  vcsa6
block            fuse        loop6             ptmx     sr0       tty18  tty31  tty45  tty59      ttyS13  ttyS27  userio   vfio
bsg              hpet        loop7             pts      stderr    tty19  tty32  tty46  tty6       ttyS14  ttyS28  vcs      vga_arbiter
btrfs-control    hugepages   loop-control      random   stdin     tty2   tty33  tty47  tty60      ttyS15  ttyS29  vcs1     vhci
bus              hwrng       mapper            rfkill   stdout    tty20  tty34  tty48  tty61      ttyS16  ttyS3   vcs2     vhost-net
cdrom            initctl     mcelog            rtc      tty       tty21  tty35  tty49  tty62      ttyS17  ttyS30  vcs3     zero
char             input       mem               rtc0     tty0      tty22  tty36  tty5   tty63      ttyS18  ttyS31  vcs4
console          kmsg        memory_bandwidth  sda      tty1      tty23  tty37  tty50  tty7       ttyS19  ttyS4   vcs5
core             lightnvm    mqueue            sda1     tty10     tty24  tty38  tty51  tty8       ttyS2   ttyS5   vcs6
cpu              log         net               sda2     tty11     tty25  tty39  tty52  tty9       ttyS20  ttyS6   vcsa
cpu_dma_latency  loop0       network_latency   sda5     tty12     tty26  tty4   tty53  ttyprintk  ttyS21  ttyS7   vcsa1
cuse             loop1       network_throughput sg0     tty13     tty27  tty40  tty54  ttyS0      ttyS22  ttyS8   vcsa2
disk             loop2       null              sg1      tty14     tty28  tty41  tty55  ttyS1      ttyS23  ttyS9   vcsa3
ecryptfs         loop3       port              shm      tty15     tty29  tty42  tty56  ttyS10     ttyS24  uhid    vcsa4
ys@ns:/dev$ ls -l null zero sda sda1 sda2 random mem
crw-r----- 1 root kmem 1, 1 сен 25 18:03 mem
crw-rw-rw- 1 root root 1, 3 сен 25 18:03 null
crw-rw-rw- 1 root root 1, 8 сен 25 18:03 random
brw-rw---- 1 root disk 8, 0 сен 25 18:03 sda
brw-rw---- 1 root disk 8, 1 сен 25 18:03 sda1
brw-rw---- 1 root disk 8, 2 сен 25 18:03 sda2
crw-rw-rw- 1 root root 1, 5 сен 25 18:03 zero
ys@ns:/dev$ head -1 urandom
ys@ns:/dev$
```

MODERN FS

Journaling FS
Advanced FS
Distributed FS

File Systems Features

More this Features implemented to Modern Advanced FS

- Holes – no store blocks of zeros in a file.
- Journaling
- Compression – transparently compress files.
- Online fsck (File System Check)
- Defragmentation
- De-duplication
- Quotas – you want to keep any one user from filling up the disk
- Encryption
- Undelete
- Secure Delete (Real rewriting Data Block)
- Snapshots (Save state of data block)
- Big Data
- Locking – may want to prevent more than one person writing a file at a time as it can get corrupted

Advanced File Systems

Modern File System include more of features of FS

- WAFL Theoretical (1994)
- WAFL ONTAP (OS Data ONTAP for Filers NetApp) (2003)
- ZFS (Oracle/Sun Microsystem Solaris) (2005)
- BtrFS (Linux/Oracle/Sun Microsystem) (2013)
- ReFS (Microsoft Windows Server) (2012)
- APFS (Apple) (2017)
- HAMMER2 (DragonFlyBSD) (2018)

File Holes File Systems

File Holes

- Why store blocks of zeros in a file? Why not instead note when a file has a ”hole” in it?
- This lets large files that are mostly zeros not take up much space on disk.
- No data blocks are allocated for holes
- Reading the hole returns zeroes.
- Backup programs which work at the file level (and not disk level) will not be aware of the hole and write zeroes.
- A UNIX file may contain holes due to the process issued an seek
- Windows ntfs also support File Holes.

Journaling File Systems

Journaling (or **Log structured**) file systems record each metadata update to the file system as a **transaction.**

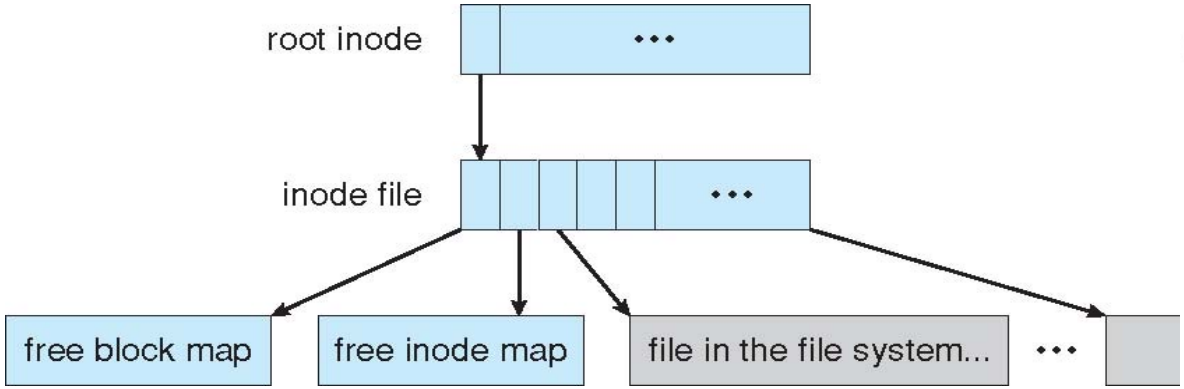
- Example of journaling FS: ext3/4, ntfs, jfs, xfs, yaffs, f2fs, btrfs, zfs, refs, apfs.
- All transactions are written to a log
  - A transaction is considered committed once it is written to the log (sequentially)
  - Sometimes to a separate device or section of disk
  - However, the file system may not yet be updated
- The transactions in the log are asynchronously written to the file system structures
  - When the file system structures are modified, the transaction is removed from the log
- If the file system crashes, all remaining transactions in the log must still be performed
- Faster recovery from crash, removes chance of inconsistency of metadata.

- Every modern file system supports journaling.
- Journaling does slow disk write performance down a tiny bit, but it's well-worth it on a desktop or laptop.
- File systems that don't offer journaling are available for use on high-performance servers.

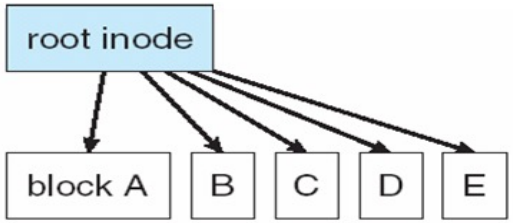
Advanced FS: WAFL FS

- WAFL - Write-anywhere file layout.
- Used on NetApp "Filers" – distributed file system appliances (2001).
- Similar to Berkeley ffs, with extensive modifications
  - Used RAID 6DP
  - Used Snapshots - data is never modified but copied.
    a)Time R/W block << Time R/W file; b)small data → used NVRAM
  - Automatic support versioning.
- Random I/O optimized, write optimized
  - Used NVRAM for write caching
- Serves up NFS, CIFS, http, ftp.

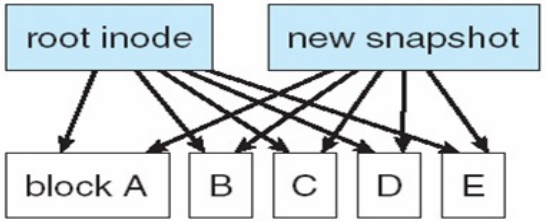
- The WAFL File Layout:



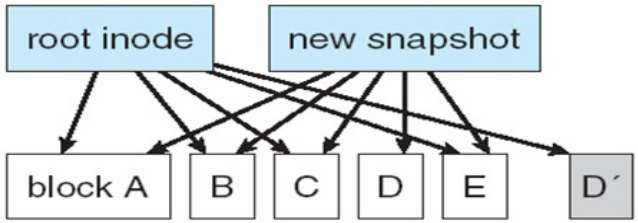
- Snapshots in WAFL



(a) Before a snapshot.



(b) After a snapshot, before any blocks change.



(c) After block D has changed to D´.

Advanced FS: ZFS

ZFS - Zettabyte File System advanced OS from Sun/Oracle (2005)

Using the ideas behind WAFL and developed in NetApp, Sun Microsystems created ZFS.

- 128-bit file system.
- Running with (10^{24} x 3)TB hard drives.
- Not really included in Linux due to licensing issues (CDDL vs GPL2)
- Acts as both the file system *and* the volume manager (RAID array)
- Goals to be super reliable.
- Can take snapshots and can roll back after problem.
- Each file has a checksum, so ZFS can tell if a file is corrupted or not.
- Checksums. Stored in parent. Other fs stores metadata with file, so if that lost then checksum also lost.
- Supports encryption.
- Limitations: needs big of RAM and big of free disk space (due to copies and snapshots). If less than 80% free space then switch from high-performance mode to space-conserve mode.

Advanced FS: BtrFS

BtrFS – advanced FS for Linux (pronounced “Butter” or “Better” FS) (2013)

- **Btrfs**, an abbreviation for **B-Tree File System**, is a FS based on the copy-on-write (COW) principle.
- Initially designed at Oracle Corporation for use in Linux.
- The development of Btrfs began in 2007, and since November 2013 the file system's marked as stable.
- BtrFS supports a lot of advanced features including drive pooling, snapshots, and dynamic disk striping (ZFS will bring many of these features to Linux by default).

- BtrFS is designed to be a clean break from the Ext series of file systems.
- Ted Ts’o, the maintainer of the Ext4 file system, considers Ext4 a short-term solution and believes BtrFS is the way forward.
- Expect to see BtrFS become the default in both enterprise server and consumer desktop Linux distributions in the next few years as it’s further tested!

Advanced FS: ReFS

ReFS - advanced OS from Microsoft (2012)

- Resilient FS - Microsoft's answer to btrfs and zfs.
- Designed to integrate data protection, snapshots, and silent high-speed background removing of corruption and data errors.
- Next generation file system after NTFS.
- Added on Windows Server 2012, Windows 8.1 and later.
- All structures 64-bit
- Windows cannot be booted from ReFS.

Advanced FS: APFS

- History Mac's FS:
  - MFS - Macintosh File System (1984, Old Apple)
  - HFS - Hierarchical File System (1985, 16bit-address)
  - HFS+ (1998, +32bit-address)
  - HFSJ (2002, +journal)
  - HFSX or APFS (2017, +more features)

APFS - new Apple FS for iOS 10.3 and later (2017)

- Fix core problems of parents HFS+, HFSJ (Hierarchical FS)
- Optimized for SSD (solid-state drive)
- Primary focus on Encryption
- 64-bit i-node numbers
- Data Integrity (Checksums)
- Clones – allow make efficient file copies on the same volume without occupying additional storage space. Changes to a cloned file are saved as deltas, required for document revisions and copies.
- Crash protection: instead of overwriting metadata, creates new metadata, points to it, and only then removes old (replaces journaling technology)

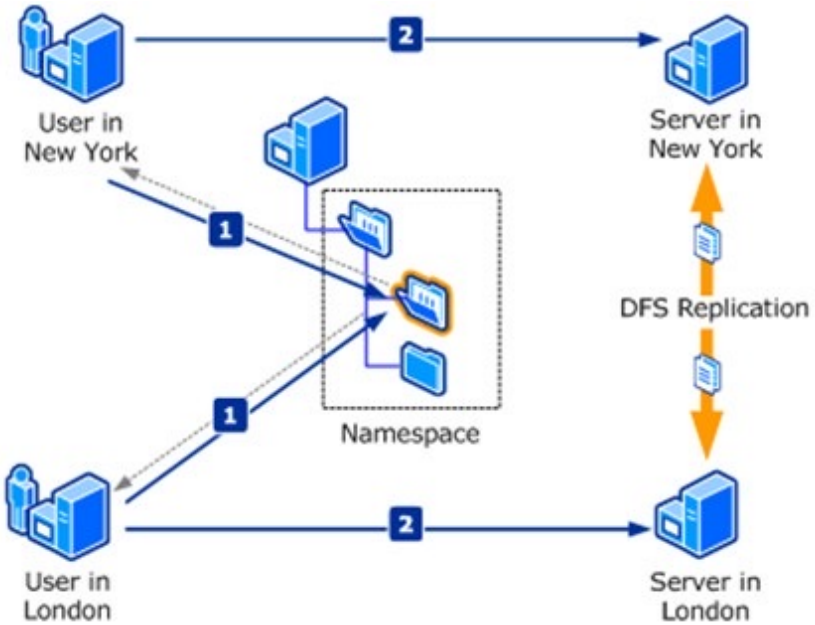
Distributed (or Networked) File Systems

Distributed (Networked) File Systems

- Allow a centralized file server to export a file-system to multiple clients.
- Redirecting user to the right copy of data.
- Provide file level access, and raw blocks access.
- Clustered file-systems also exist, where multiple servers work in together.

DFS Architectures

- as Client-Server Architecture (Centralized)
  - NFS (Network File System)
  - CIFS/SMB
    (Windows Common Internet FS/Samba protocol)
  - Andrew FS
- as Cluster-Based Arch (Less Centralized)
  - GFS (Global File System, Google FS)
- as Symmetric Arch (Fully Distributed)
  - DHT-based (Distributed Hash Table)



- Comparison of distributed file systems

https://en.wikipedia.org/wiki/Comparison_of_distributed_file_systems

END