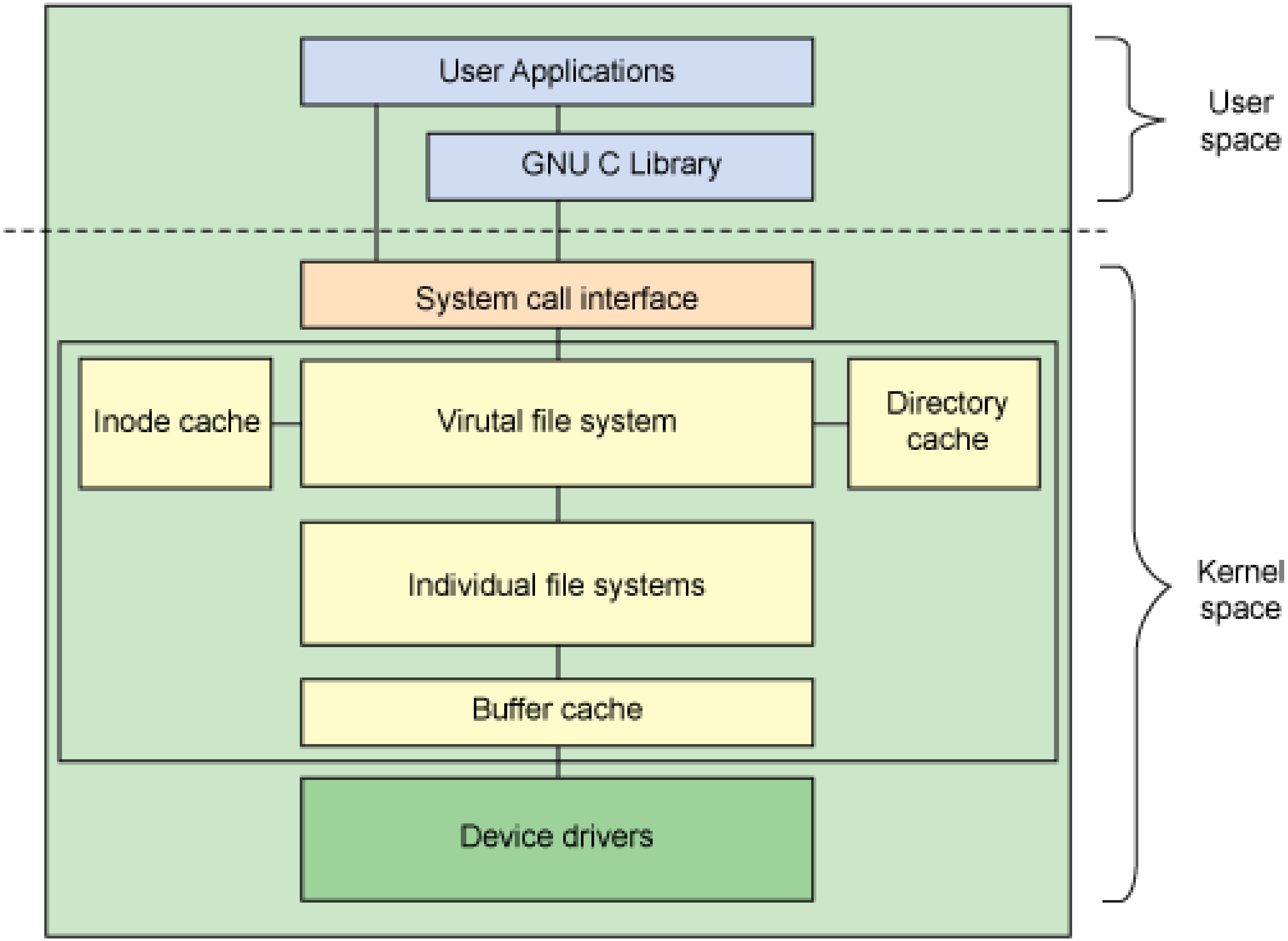
COMP 175

System Administration and Security

The File System

001000001 001000001 01010010 01010010 01010010

File System Architecture

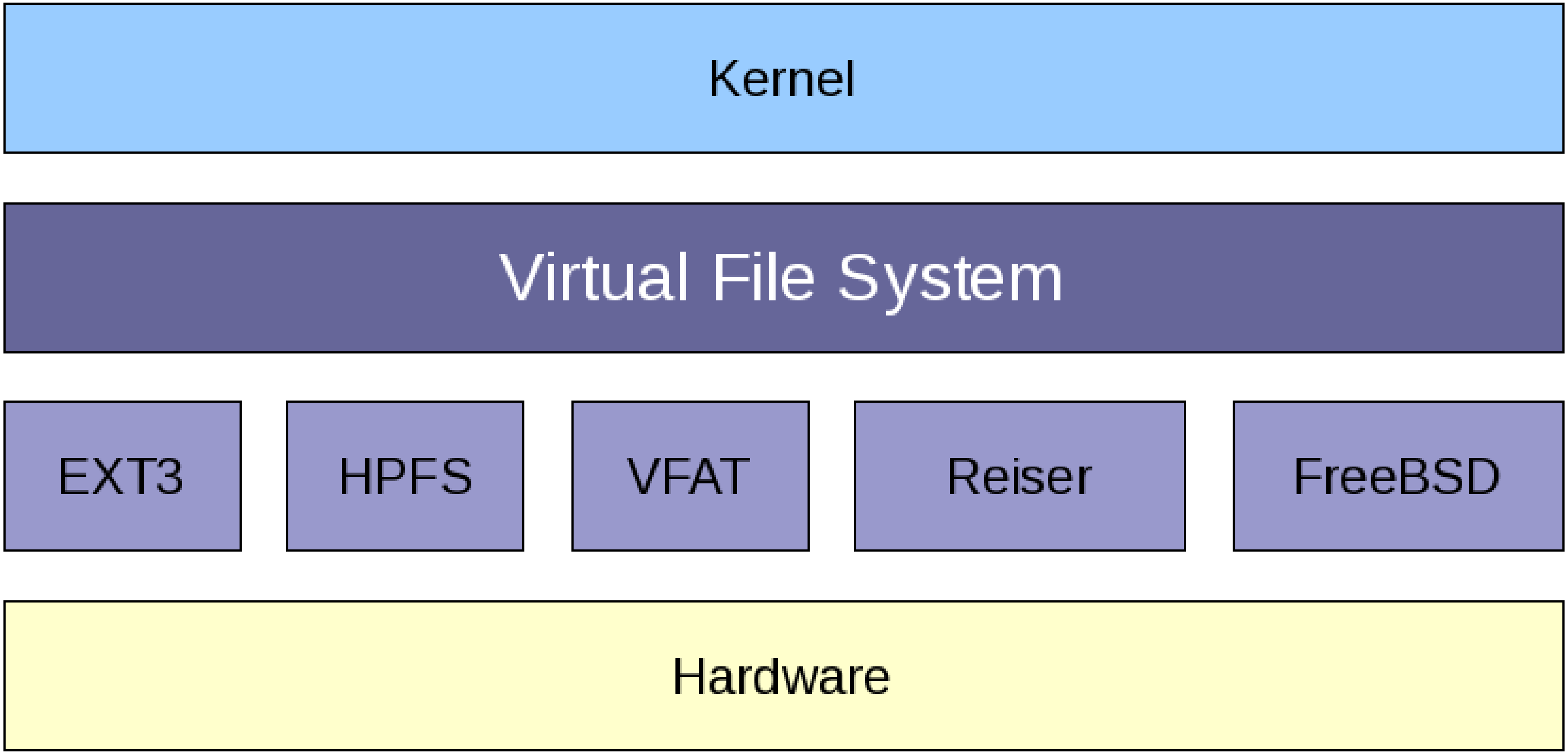


Structural view of the file system

File System Architecture

- User space contains the applications (the user of the file system) and the GNU C Library (glibc), which provides the user interface for the file system calls (open, read, write, close). The system call interface acts as a switch, funneling system calls from user space to the appropriate endpoints in kernel space.
- The VFS is the primary interface to the underlying file systems. This component exports a set of interfaces and then abstracts them to the individual file systems, which may behave very differently from one another.
- Two caches exist for file system objects (inodes and dentries – directory entries). Each provides a pool of recently-used file system objects.

File System Architecture



Kernel

Virtual File System

EXT3

HPFS

VFAT

Reiser

FreeBSD

Hardware

Simplified further it looks like the above

With ~100 file systems supported – many legacy systems
one way to support legacy file systems

Virtual File System Layer

- VFS acts as the root level of the file system interface. The VFS keeps track of the currently-supported file systems, as well as those file systems that are currently mounted
- File systems can be dynamically added/removed
- Kernel keeps list of currently-supported file systems, which can be viewed from user space through the /proc file system.
- /proc virtual file also shows the devices currently associated with the file systems

We will get back to /proc

Buffer Cache

Buffer cache keeps track of r/w requests from the:

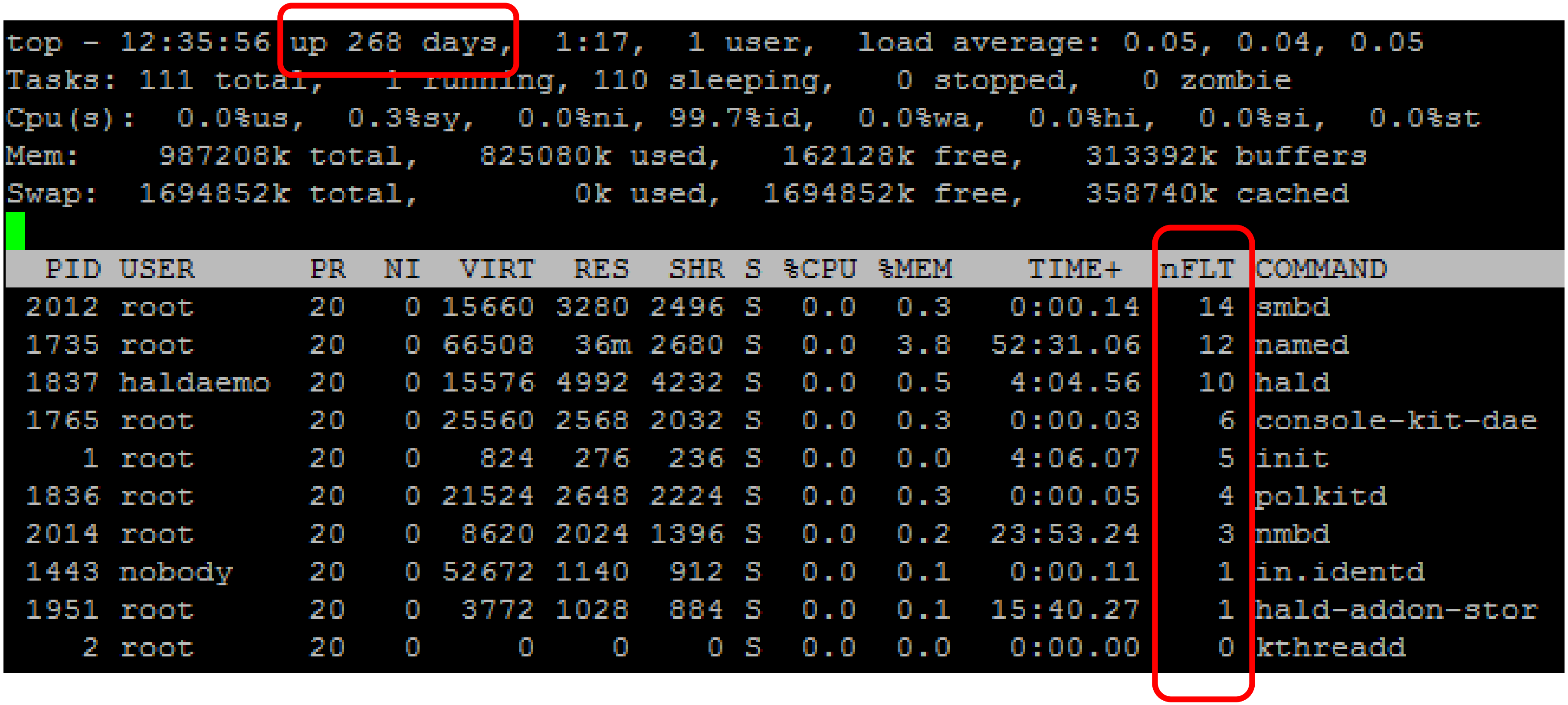
- Individual file system implementations
- Physical devices (through device drivers)
  - Linux maintains a cache of the requests
  - Avoids going back to physical device for all requests - efficient
  - The most-recently used buffers (pages) are cached thus can be quickly provided back to the individual file systems
    - Not found in cache? – Page fault

Page Faults

- The thread experiencing the page fault is put into a Wait state while the operating system finds the specific page on disk and restores it to physical memory
  - Takes time
    - Sends trap to the OS
    - Save user registers and process state
    - Determine location of the page on the disk
- Large numbers of page faults are an indication of insufficient RAM
- Also cause page reads (see disk counters)

Page Faults

- top   add fault stats via f nMaj (not as flags)



```
top - 12:35:56 up 268 days,  1:17,  1 user,  load average: 0.05, 0.04, 0.05
Tasks: 111 total,   1 running, 110 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    987208k total,   825080k used,   162128k free,   313392k buffers
Swap:  1694852k total,        0k used,  1694852k free,   358740k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  nFLT COMMAND
 2012 root      20   0 15660 3280 2496 S  0.0  0.3   0:00.14   14 smbd
 1735 root      20   0 66508  36m 2680 S  0.0  3.8  52:31.06   12 named
 1837 haldaemo  20   0 15576 4992 4232 S  0.0  0.5   4:04.56   10 hald
 1765 root      20   0 25560 2568 2032 S  0.0  0.3   0:00.03    6 console-kit-dae
    1 root      20   0   824  276  236 S  0.0  0.0   4:06.07    5 init
 1836 root      20   0 21524 2648 2224 S  0.0  0.3   0:00.05    4 polkitd
 2014 root      20   0  8620 2024 1396 S  0.0  0.2  23:53.24    3 nmbd
 1443 nobody    20   0 52672 1140  912 S  0.0  0.1   0:00.11    1 in.identd
 1951 root      20   0  3772 1028  884 S  0.0  0.1  15:40.27    1 hald-addon-stor
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00    0 kthreadd
```

Block & Character Devices

Block devices move data to/from that occur in blocks (such as disk sectors)

Supports buffering and random access behavior (is not required to read blocks sequentially, but can access any block at any time). Block devices include hard drives, CD-ROMs, RAM disks.

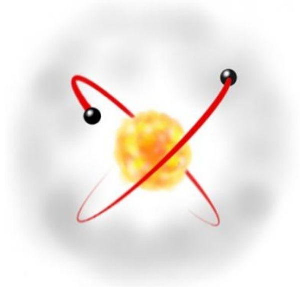
Character devices differ in that they do not have a physically-addressable media. Character devices include serial ports and tape devices, in which data is streamed character by character.

Journaling

- Journaling keeps track of the major steps taken during last file sessions
- If system crashes, it can boot and back up to the last known good configuration and recover to the point of the crash
- Hard drives may have their own write caches, journaling thus forces the device to flush its cache at certain points in the journal (called barriers in ext3 and ext4)
- ACID (atomicity, consistency, isolation, durability) a set of properties that guarantee database transactions are processed reliably

Vocabulary

- Atomicity (database systems): a property of database transactions which are guaranteed to either completely occur, or have no effects.
- Atomicity states that database modifications must follow an “all or nothing” rule. Each transaction is said to be “atomic.” If one part of the transaction fails, the entire transaction fails. It is critical that the database management system maintain the atomic nature of transactions in spite of any DBMS, operating system or hardware failure



Linux File Systems

Linux has several standard file systems

- ext2 - legacy, general purpose, based on UFS
- ext3 – journaling, based on UFS
  - avoids long file system checks after system crash
- ext4 – journaling successor to ext2
- ReiserFS – journaling file system (infamous)
- XFS – IRIX’s file system (good for streaming media)
- ZFS – Sun’s combined file system & volume mgr
- procfs – interface to internal kernel structures
- swap – used to support virtual memory

Additional File Systems

Several foreign file systems are supported

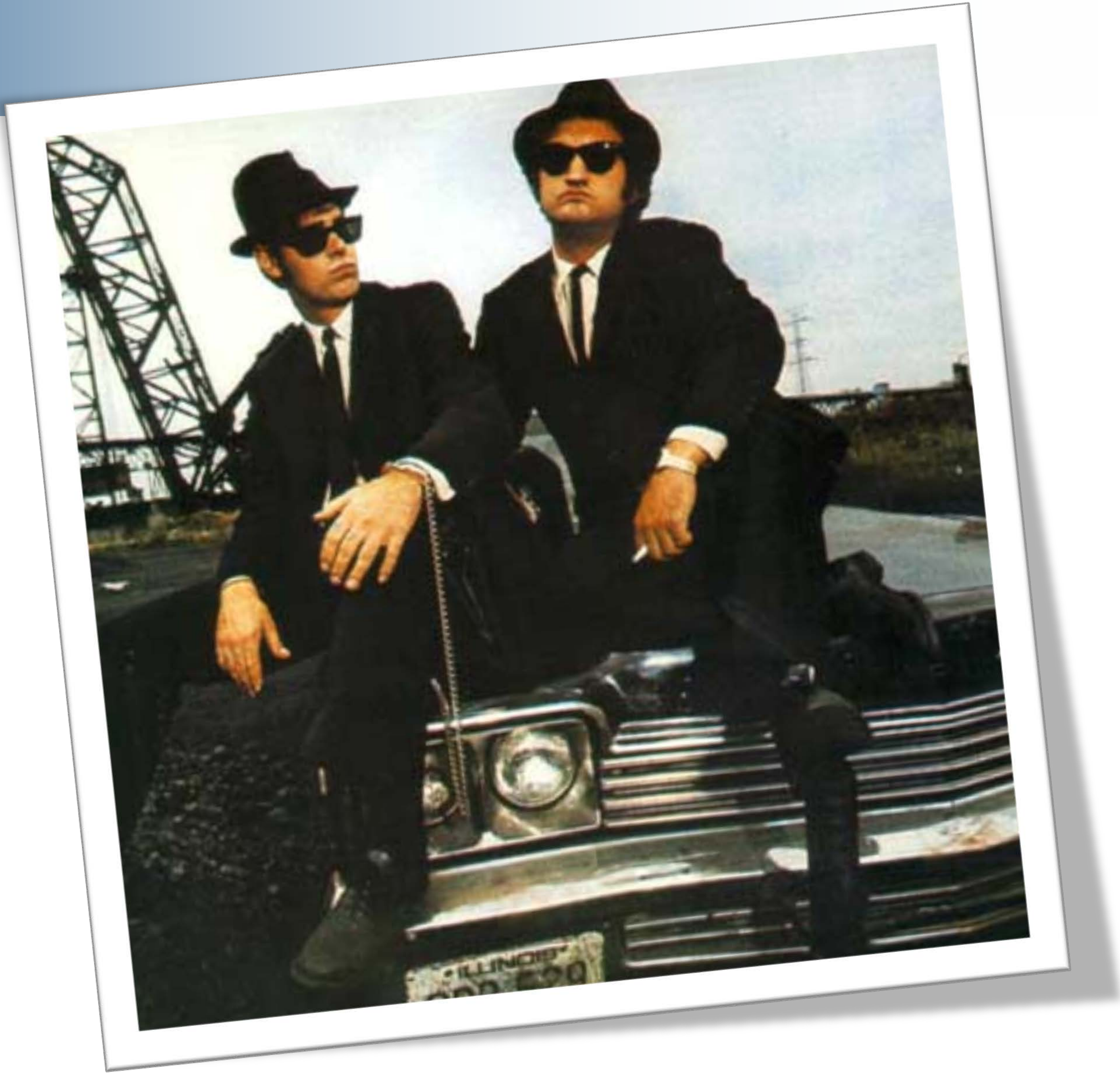
- Easier to exchange files with another OS
- Work just like native ones, except:
  - May lack some usual UNIX feature
    - Long File Name support
    - UNIX permissions
  - Have curious limitations/oddities

CDROM file systems supported

- isofs – iso9660 CDROM file system
- Joliet - Microsoft CDROM filesystem extensions
  - Why is it called Joliet, you ask?

Joliet ISO 9660 Extension

- At the time of Windows NT 4.0 and Windows 95/98, the file system of choice to record files with names of 128 characters was called Romeo.
- Romeo didn’t contain the ISO 9660 file system and broke backward compatibility with DOS
- Joliet combined Romeo and ISO 9660
- Joliet.doc metadata file in MS Win95 Driver Development Kit comment: "Joliet is a small town just outside of Chicago, where a man named Jake did some time in The Blues Brothers."



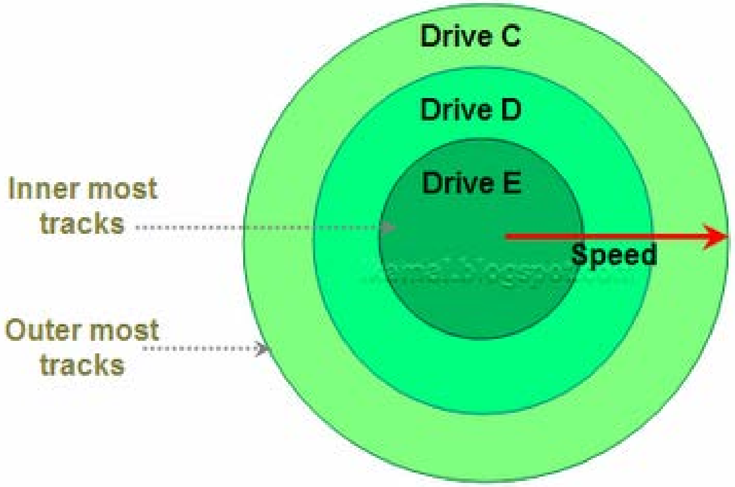
Elwood J. Blues, “Joliet" Jake B. Blues, Bluesmobile

Additional File Systems

- NFS - Network file system allows multiple users or hosts to share the same files using a client/server methodology
- NTFS – preferred Microsoft file system since NT provides ACL’s and journaling
- FAT, FAT32 – potential 8.3 vs. Long Filenames (LFN), permissions complications
  - Used to mount usb flash drives
  - vfat – LFN compatible

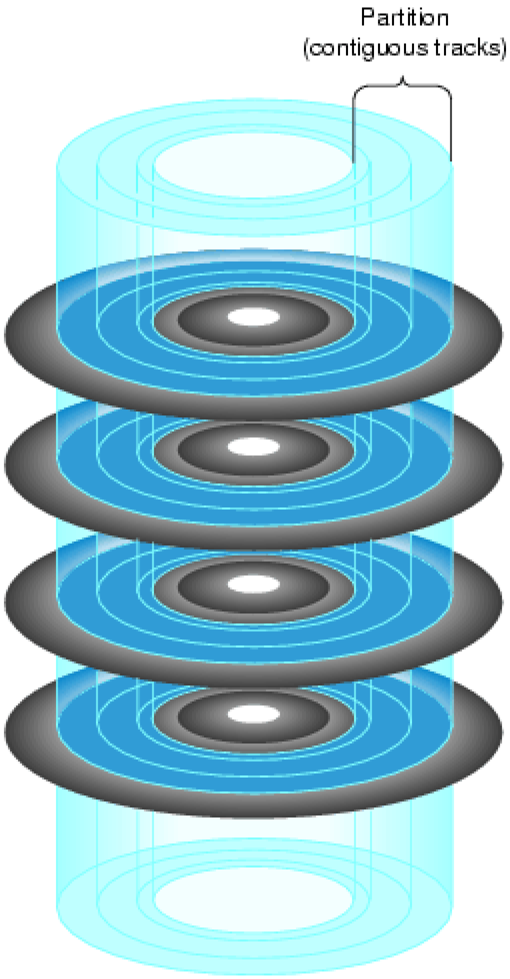
Partitioning

- Partitioning is dividing a single hard drive into several logical drives.
- A partition is a contiguous set of blocks on a drive, treated as an independent disk.
- A partition table is an index that relates sections of the hard drive to partitions.



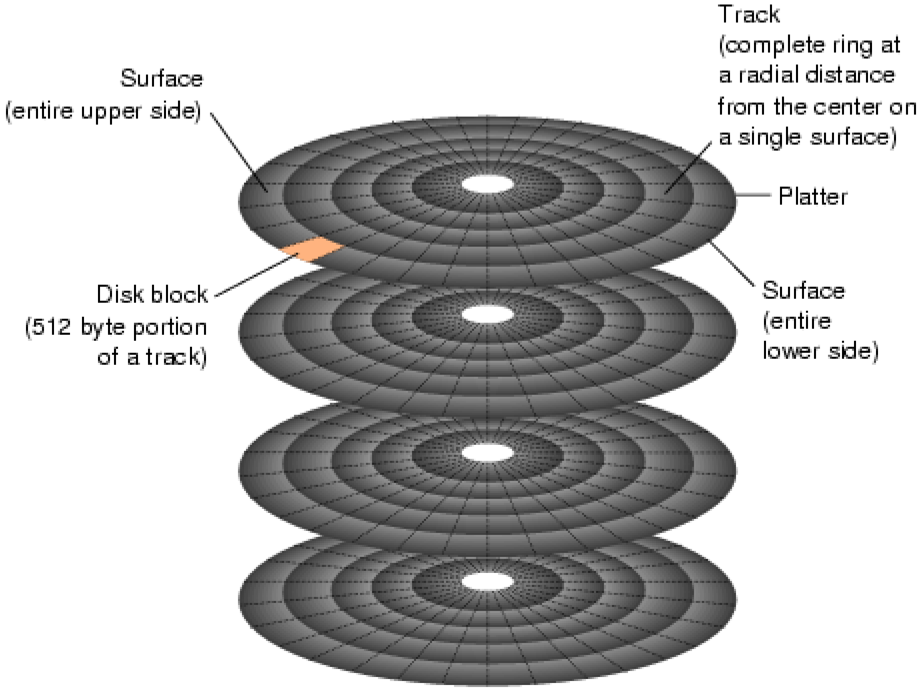
Multiple Partitions

- Multiple partitions reduce risk of system failure should a partition becomes full
- Segregating OS and user data space protects operating system if allocated disk space is exhausted
- Partitions can contain different **operating systems**
- Partitions can contain different **file systems**
- Called a *slice* in BSD, Solaris, and GNU Hurd



Physical Disk Structure

- 1 or more circular platters
- Platter has upper & lower oxide-coated surface
- Heads - min 1 per surface
  - Mounted on arms
- Heads float very close to platter surfaces
  - never touching them
  - disk crash


Track
(complete ring at a radial distance from the center on a single surface)
Surface
(entire upper side)
Platter
Disk block
(512 byte portion of a track)
Surface
(entire lower side)

Partition Fields

- Device: the partition's device name
- Start: drive sector where partition begins
- End: drive sector where partition ends
- Size: partition's size (in MB)
- Type: partition type (e.g. ext2, ext3, or vfat)
- Mount Point: where partition will be mounted within directory hierarchy (e.g. /,/var,/usr)

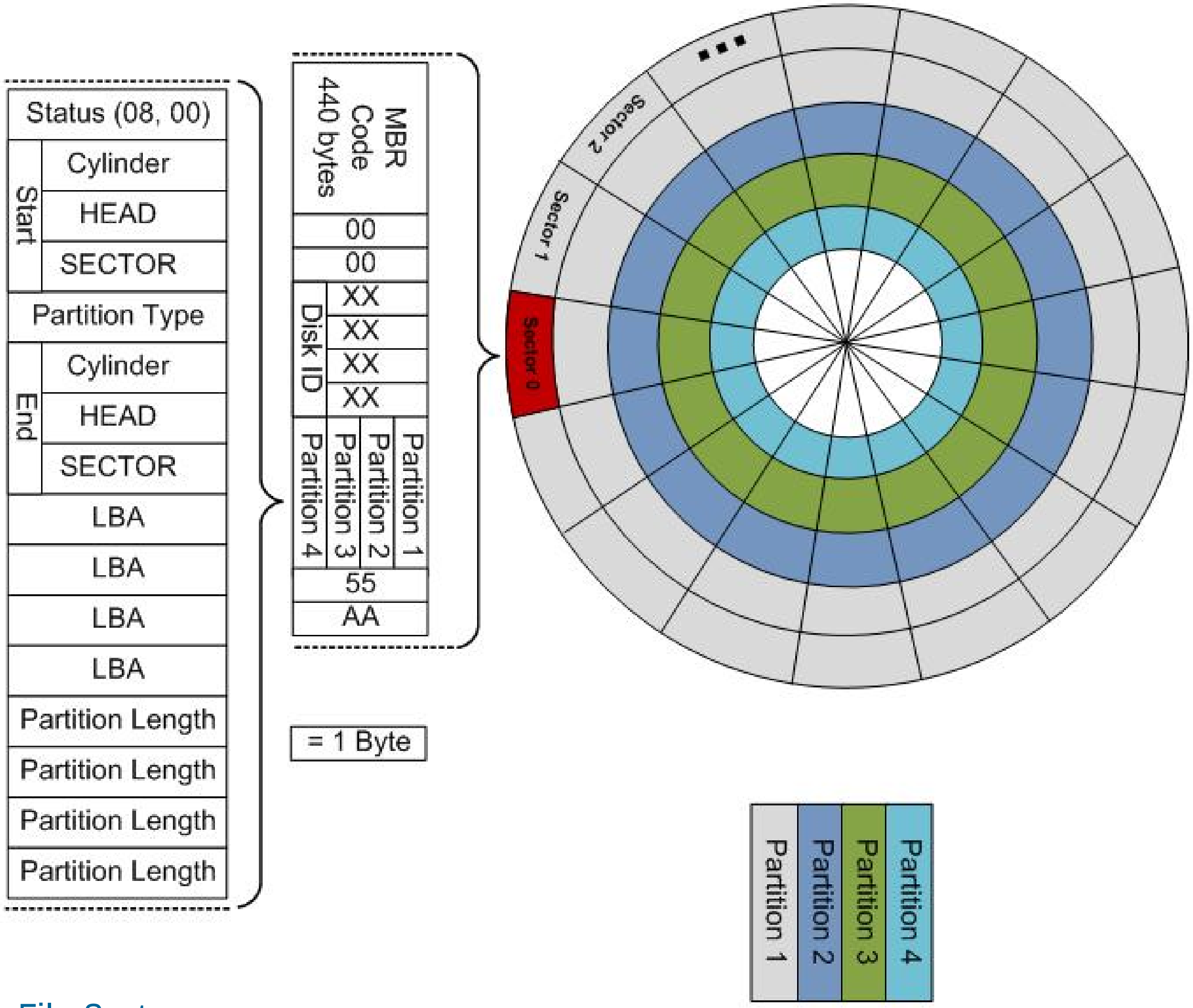
- Tools - fdisk, cfdisk
- Always document settings – hard copy

Master Boot Record

- MBR – 512 byte sequence at first sector of drive
- MBR used for one or more of:
  - Holds a partition (thus called a partition sector)
  - Bootstrapping an operating system.
    - PC BIOS loads the MBR from disk and passes execution to machine code instructions at the beginning of the MBR
  - Identify disk with a 32-bit disk signature
- MSDOS `fdisk /mbr` rewrites MBR – undocumented
- GRUB and LILO can write to the MBR

Master Boot Record

MBR in more detail



Status (08, 00)
Start: Cylinder, HEAD, SECTOR
Partition Type
End: Cylinder, HEAD, SECTOR
LBA
LBA
LBA
LBA
Partition Length
Partition Length
Partition Length
Partition Length

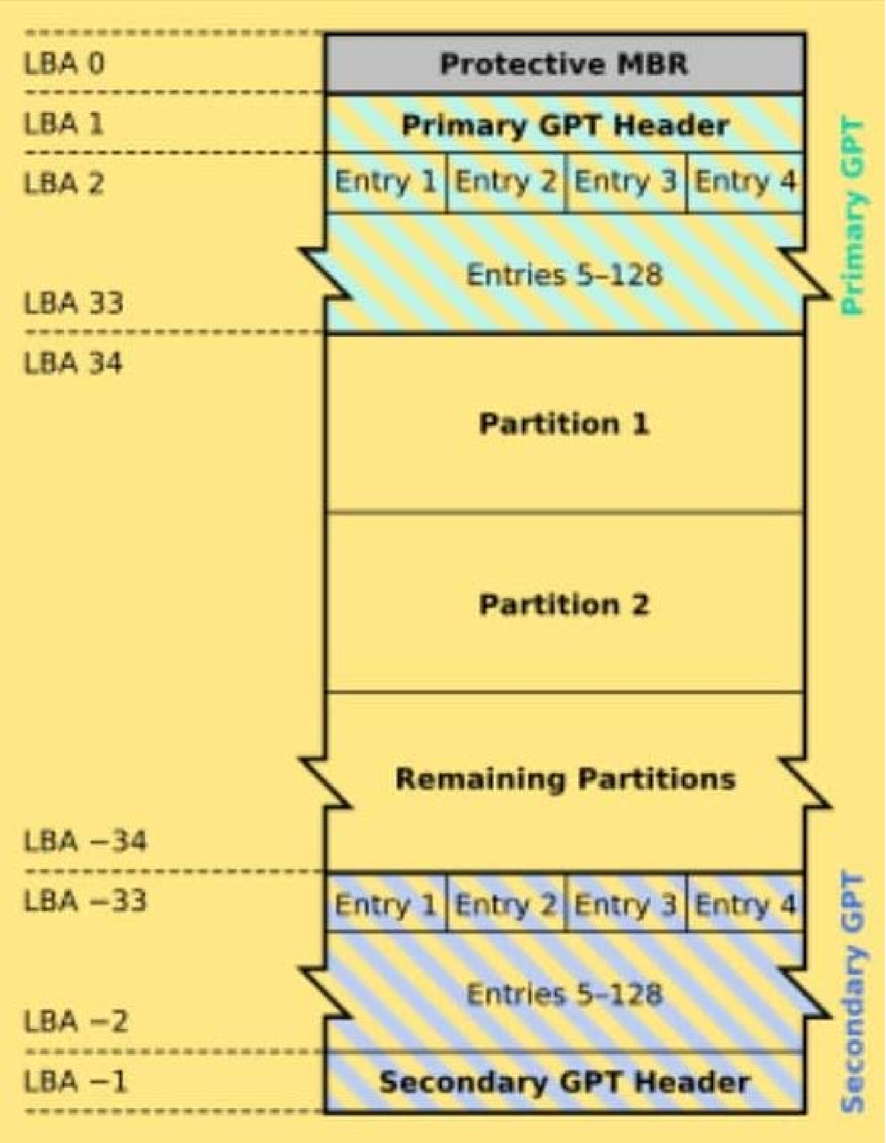
MBR Code 440 bytes
00
00
Disk ID: XX XX XX XX
Partition 1 | Partition 2 | Partition 3 | Partition 4
55
AA

□ = 1 Byte

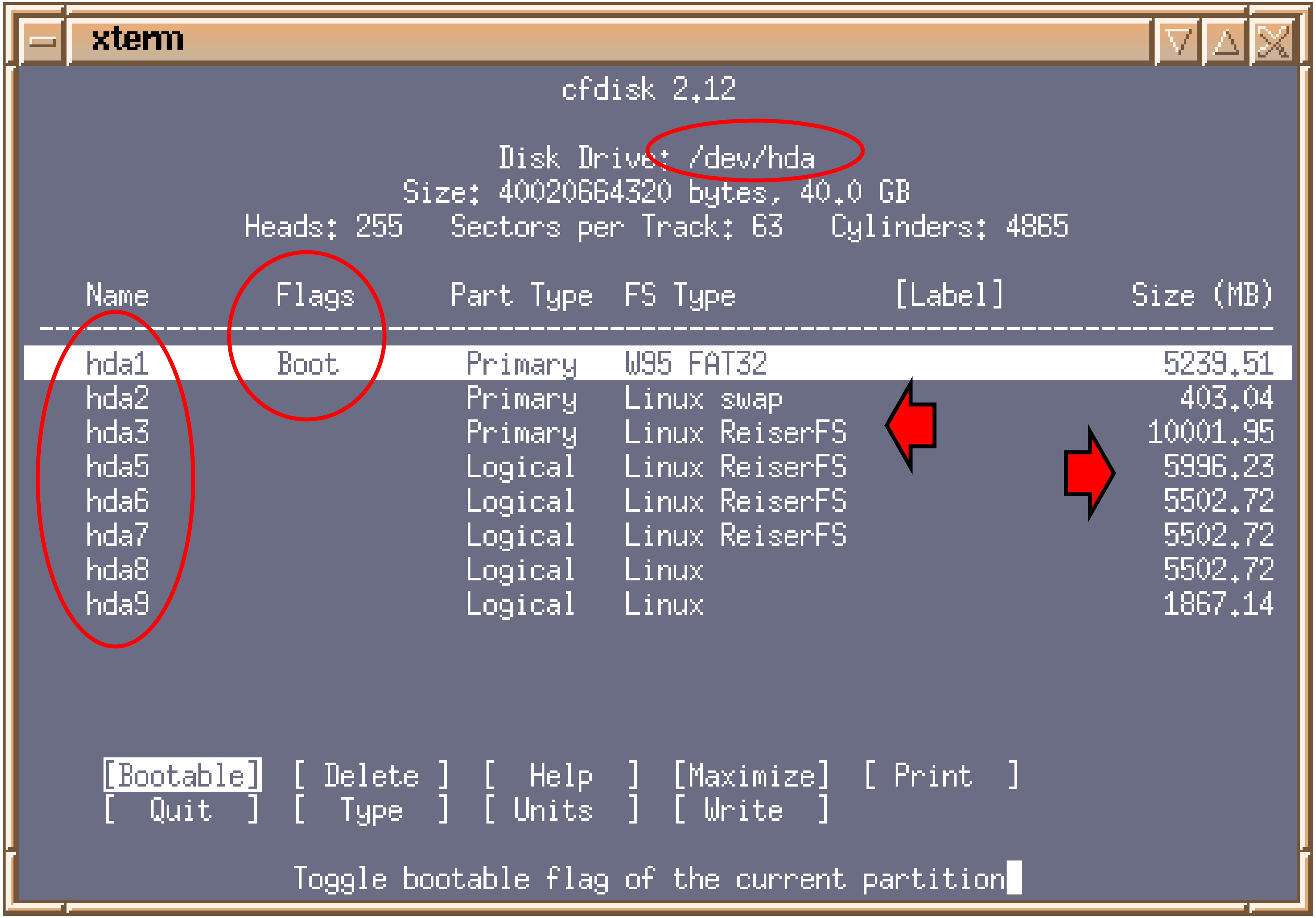
The partition definition section of the MBR could put them anywhere, they do not need to be outer to innermost

Master Boot Record

- The organization of the partition table in the MBR limits the maximum addressable storage space of a partitioned disk to 2 TiB (232 × 512 bytes).
- MBR-based partitioning scheme is in the process of being superseded by the GUID (Globally Unique Identifiers) Partition Table (GPT). Can co-exist with an MBR.
- GPT is part of UEFI standard



cfdisk



```
                              cfdisk 2.12

                          Disk Drive: /dev/hda
                    Size: 40020664320 bytes, 40.0 GB
           Heads: 255   Sectors per Track: 63   Cylinders: 4865

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
    hda1        Boot        Primary   W95 FAT32                         5239.51
    hda2                    Primary   Linux swap                         403.04
    hda3                    Primary   Linux ReiserFS                   10001.95
    hda5                    Logical   Linux ReiserFS                    5996.23
    hda6                    Logical   Linux ReiserFS                    5502.72
    hda7                    Logical   Linux ReiserFS                    5502.72
    hda8                    Logical   Linux                             5502.72
    hda9                    Logical   Linux                             1867.14







     [Bootable]  [ Delete ]  [  Help  ]  [Maximize]  [ Print  ]
     [  Quit  ]  [  Type  ]  [ Units  ]  [ Write  ]

                 Toggle bootable flag of the current partition
```

Disk Partition

IDE Disk Partition Example

Note: Physical drives can contain max. 4 primary partitions

- /dev/hda (Primary Master Disk)
  - /dev/hda1 (First Primary Partition)
  - /dev/hda2 (Second Primary Partition)
- /dev/hdb (Primary Slave Partition)
  - /dev/hdb1
- /dev/hdc (Secondary Master/Slave Partition)
  - /dev/hdc1

Disk Partition

SATA and SCSI Disk Partition Example

Note: Physical drives can contain max. 4 primary partitions

- /dev/sda (Primary Master Disk)
  - /dev/sda1 (First Primary Partition)
  - /dev/sda2 (Second Primary Partition)
- /dev/sdb (Primary Slave Partition)
  - /dev/sdb1
- /dev/sdc (Secondary Master/Slave Partition)
  - /dev/sdc1

Warning

- MultiBoot with Windows & Linux can be tricky
- Partitioning can be complicated
- Be careful altering partitions
  - Assume it will go bad
- Backup data prior to making any changes
- Read the documentation
- Research what others have done
- Check with other students
- Have a backout plan

Linux Disk Utilities

fdisk /dev/hda	Linux/DOS drive partitioning tool
cfdisk /dev/hda	Easier to use
sfdisk –l	Lists the partition tables
parted /dev/hda	Partition manipulation tool
fsck -t ext2 /dev/hda2	Check & repair file system

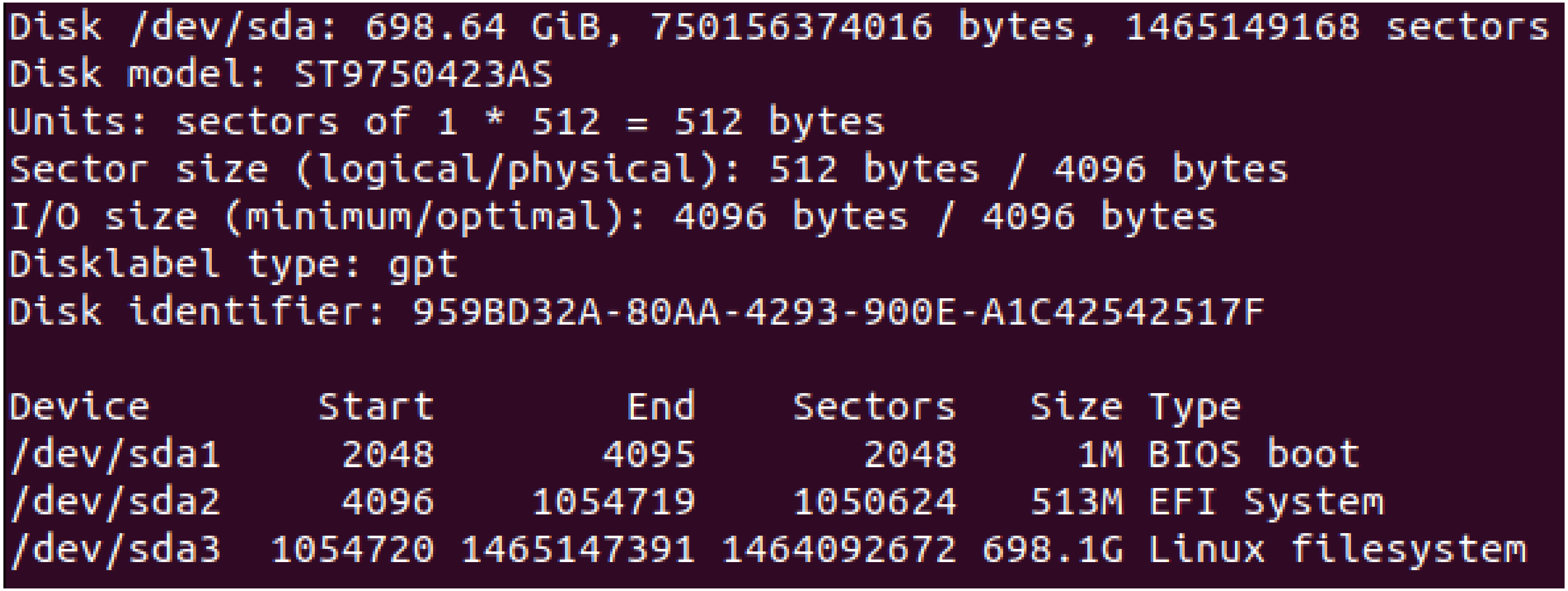
- fsck runs automatically at boot if OS detects files system wasn’t properly shut down. Run when file system is unmounted or mounted read-only.
- Similar to MS scandisk or chkdsk

sfdisk

```
root@tea:~# sfdisk –l
Disk /dev/hda: 1027 cylinders, 255 heads, 63 sectors/track
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0

   Device Boot Start     End   #cyls    #blocks      Id  System
/dev/hda1   *      0+    995     996-   8000338+  83  Linux
/dev/hda2        996    1026      31     249007+   82  Linux swap
/dev/hda3          0       -       0          0     0   Empty
/dev/hda4          0       -       0          0     0   Empty
```

sfdisk

- sudo sdfisk – l   (edited)   Ubuntu
  - gpt
  - UEFI



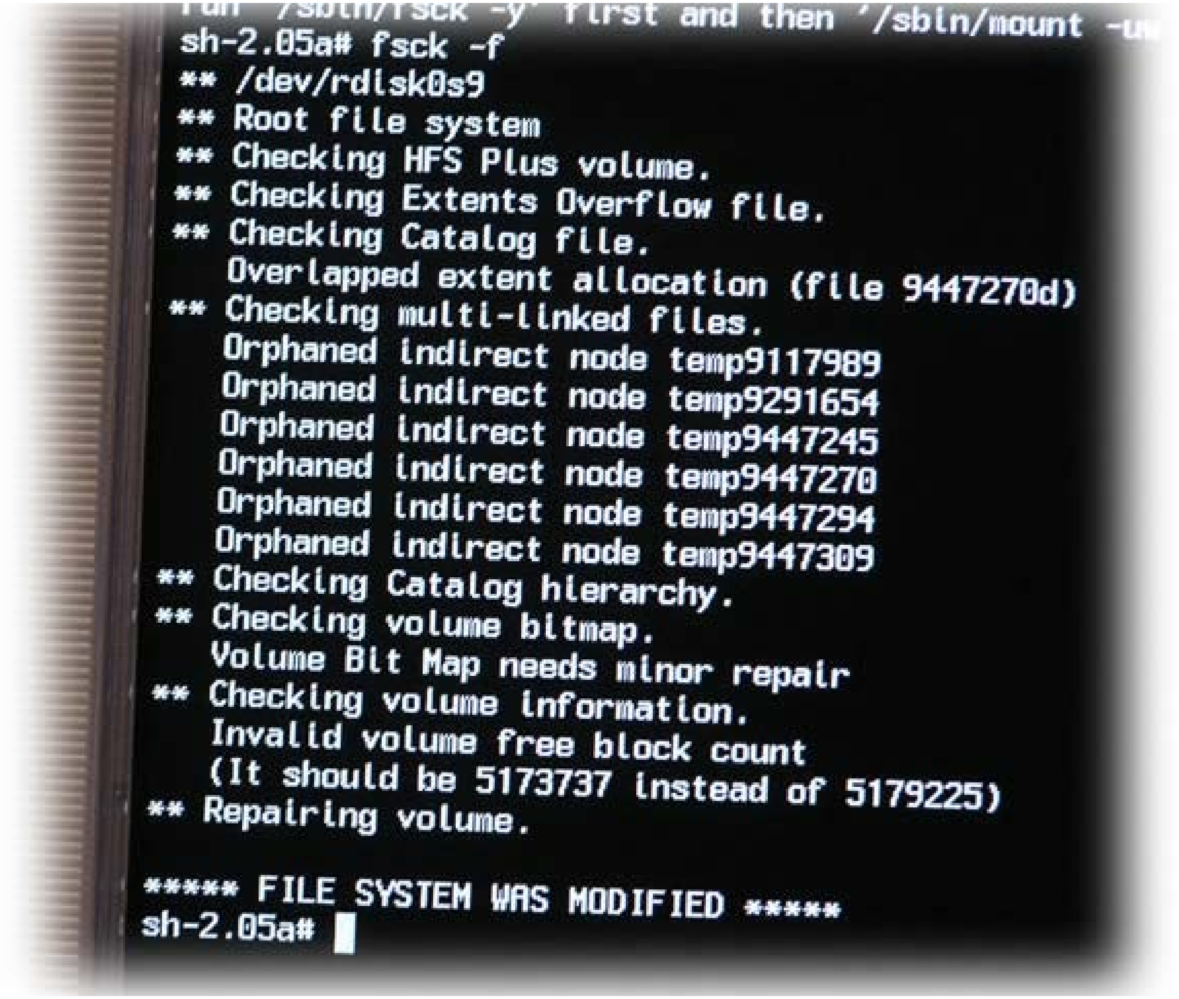
```
Disk /dev/sda: 698.64 GiB, 750156374016 bytes, 1465149168 sectors
Disk model: ST9750423AS
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: 959BD32A-80AA-4293-900E-A1C42542517F

Device         Start        End    Sectors   Size Type
/dev/sda1       2048       4095       2048     1M BIOS boot
/dev/sda2       4096    1054719    1050624   513M EFI System
/dev/sda3    1054720 1465147391 1464092672 698.1G Linux filesystem
```

Inconsistent State

- A non-graceful shutdown (crash, power loss) can leave a file system in an Inconsistent State. Prior to journaling file systems, it was common for an improperly shut-down Unix system's file system to develop a corrupted superblock.
- Running fsck to fix this could take minutes to hours (volume size and disk I/O throughput)
- The consequences of fsck not being able to fix the error are not good
- Hence: “fsck” and “fscked”

fsck on OS-X

Note: Spelling auto-correct doesn't play well with UNIX terms



```
run '/sbin/fsck -y' first and then '/sbin/mount -uw
sh-2.05a# fsck -f
** /dev/rdisk0s9
** Root file system
** Checking HFS Plus volume.
** Checking Extents Overflow file.
** Checking Catalog file.
   Overlapped extent allocation (file 9447270d)
** Checking multi-linked files.
   Orphaned indirect node temp9117989
   Orphaned indirect node temp9291654
   Orphaned indirect node temp9447245
   Orphaned indirect node temp9447270
   Orphaned indirect node temp9447294
   Orphaned indirect node temp9447309
** Checking Catalog hierarchy.
** Checking volume bitmap.
   Volume Bit Map needs minor repair
** Checking volume information.
   Invalid volume free block count
   (It should be 5173737 instead of 5179225)
** Repairing volume.

***** FILE SYSTEM WAS MODIFIED *****
sh-2.05a# █
```

Remember

- MBR: Master Boot Record
- Superblock is a boot record
- Big-endian machine stores most significant byte first
- Little-endian machine stores least significant byte first

Ponder this

- Journaling file systems
- Raid
- What if you don't have ECC?
- Error correction code memory (ECC memory) uses an error correction code (ECC) to detect and correct n-bit data corruption which occurs in memory. ECC memory is used in most computers where data corruption cannot be tolerated, like industrial control applications, critical databases, and infrastructural memory caches.