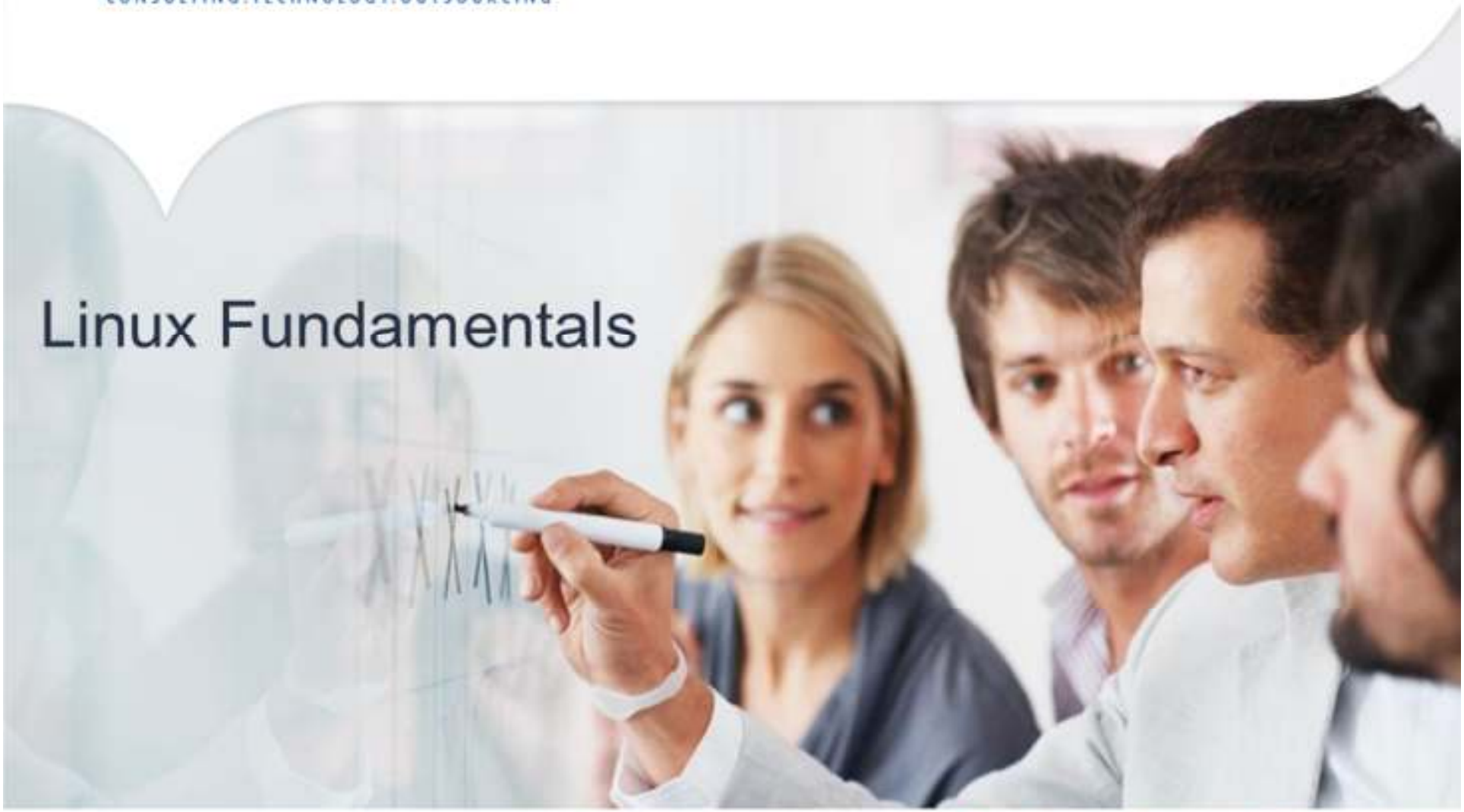
Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

Linux Fundamentals

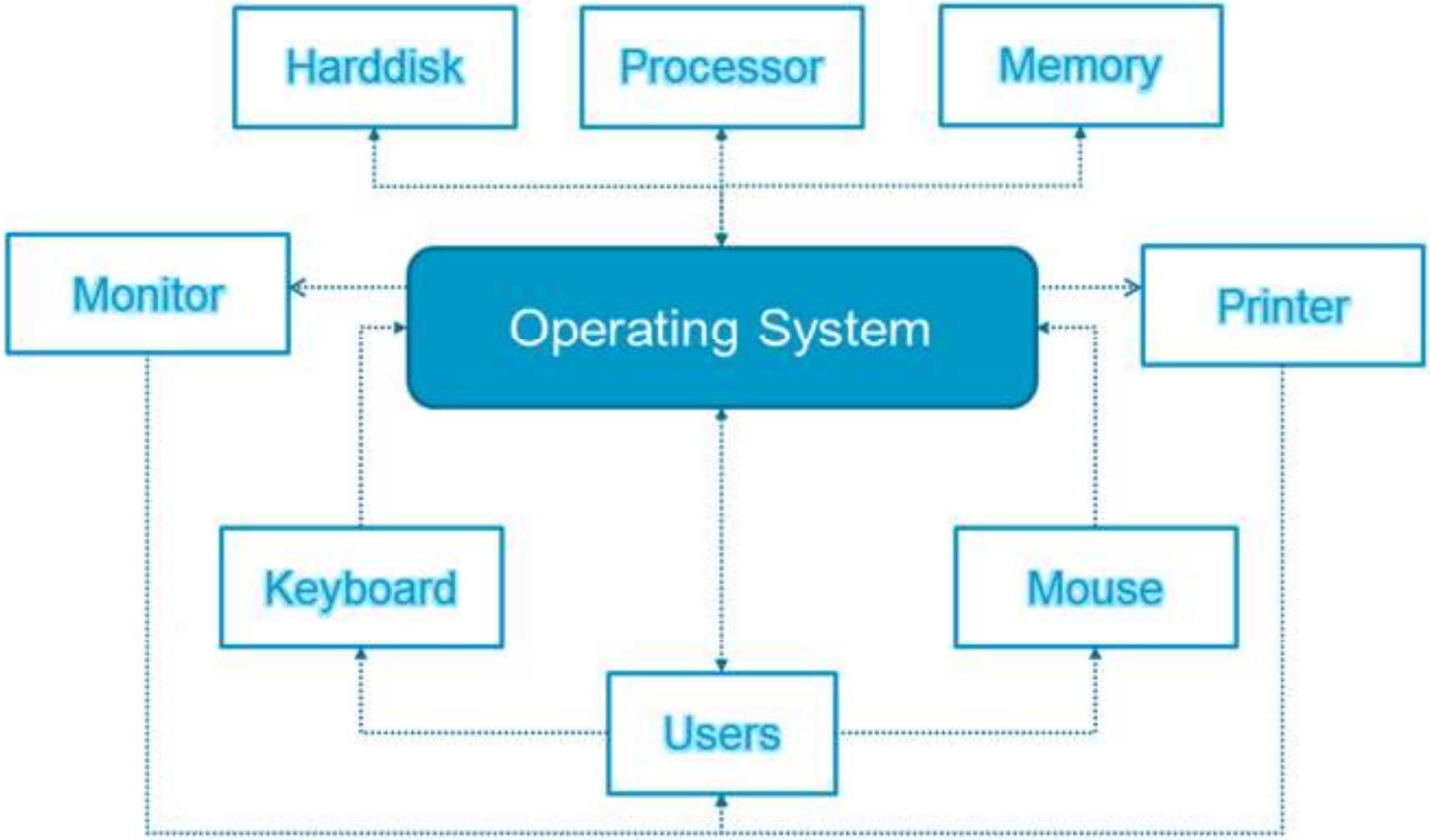


People matter, results count.

Contents

- Open source
- Introduction to Linux
- Linux Subsystem
- Booting Process
- Shell
- Process Management
- File System
- Device Management
- Memory Management

Operating System - Recall



Software's and Licensing

Freeware

Free with Limited Features

Shareware

Free for limited time

Commercial

Full / Partial features on cost

Open Source

Free / Cost with source code

Open Source

1950's and 60's – Source was available

1960's – Software's became cost

1983 – Free software movement, GNU Project

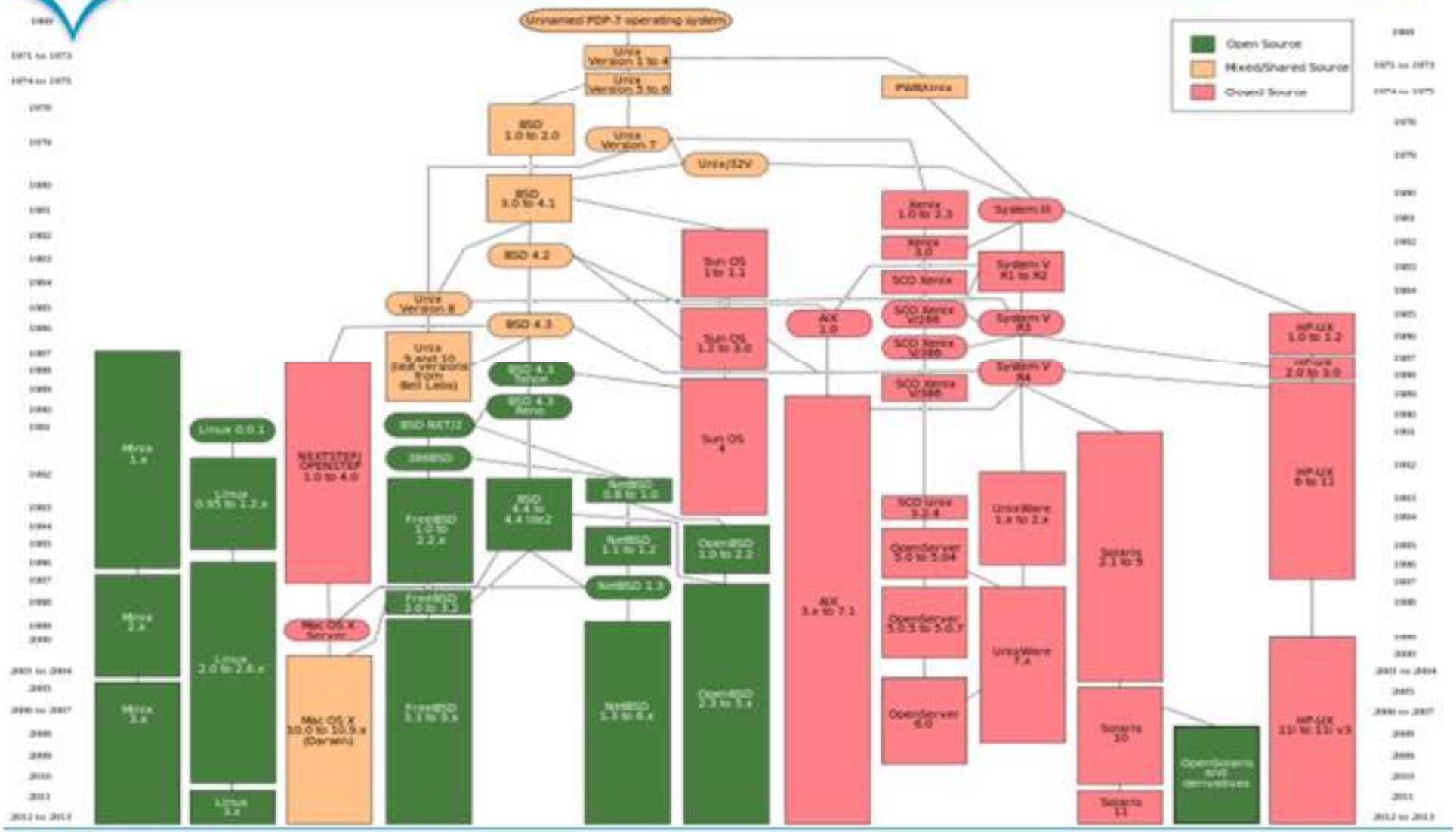
1990's – Linux, Open source initiative (1997)

Open source licenses

License	License & Copyright Notice	Commercial Use	Disclosure of source	State changes	Distribution	Patent Use
ASL2.0	☑	☑	☒	☑	☑	☑
GNU GPL3.0	☑	☑	☑	☑	☑	☑
MIT	☑	☑	☒	☒	☑	☒

Reference : http://choosealicense.com/licenses/

Unix



Capgemini

Unix (contd.,)

Created in Bell Laboratories – 1960s

Re-Written in C Language - 1972

Multiple flavors evolved (HP-UX, Solaris, AIX) – 1980s

Linux, Open source BSD released – 1990s

Linux

MINIX was created – Andrew - 1987

Linus Torvalds developed a kernel – Linux - 1991

GNU Project

Free Software collaboration – 1983 – Richard Stallman

Initially – More software for Unix are developed

Shells, compilers, libraries are created for Linux

GNU GPL license evolved to 3.0

Linux Distributions

redhat

openSUSE

fedora

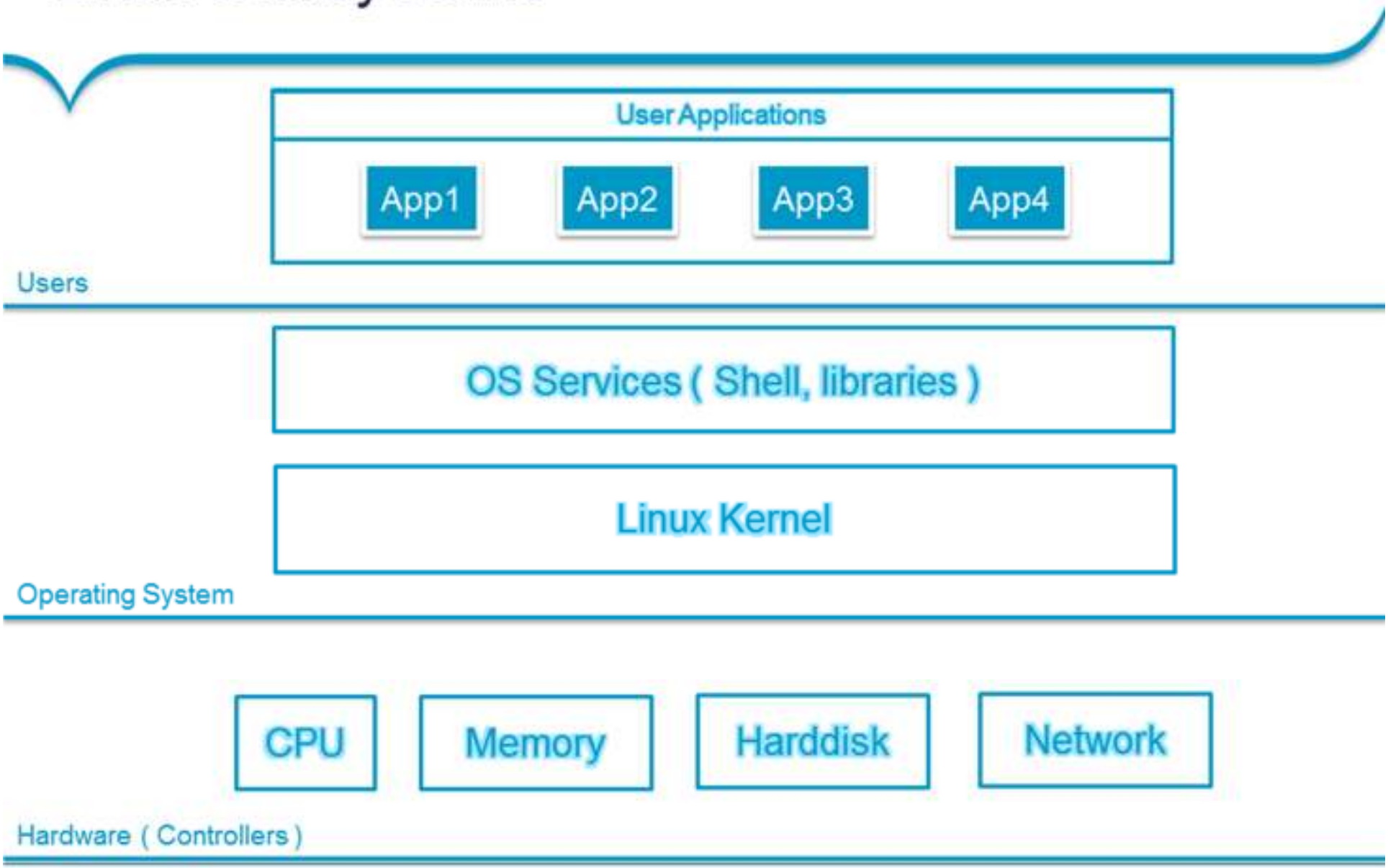
CentOS

debian
GNU/Linux

ubuntu

Linux Subsystems

Linux Subsystems

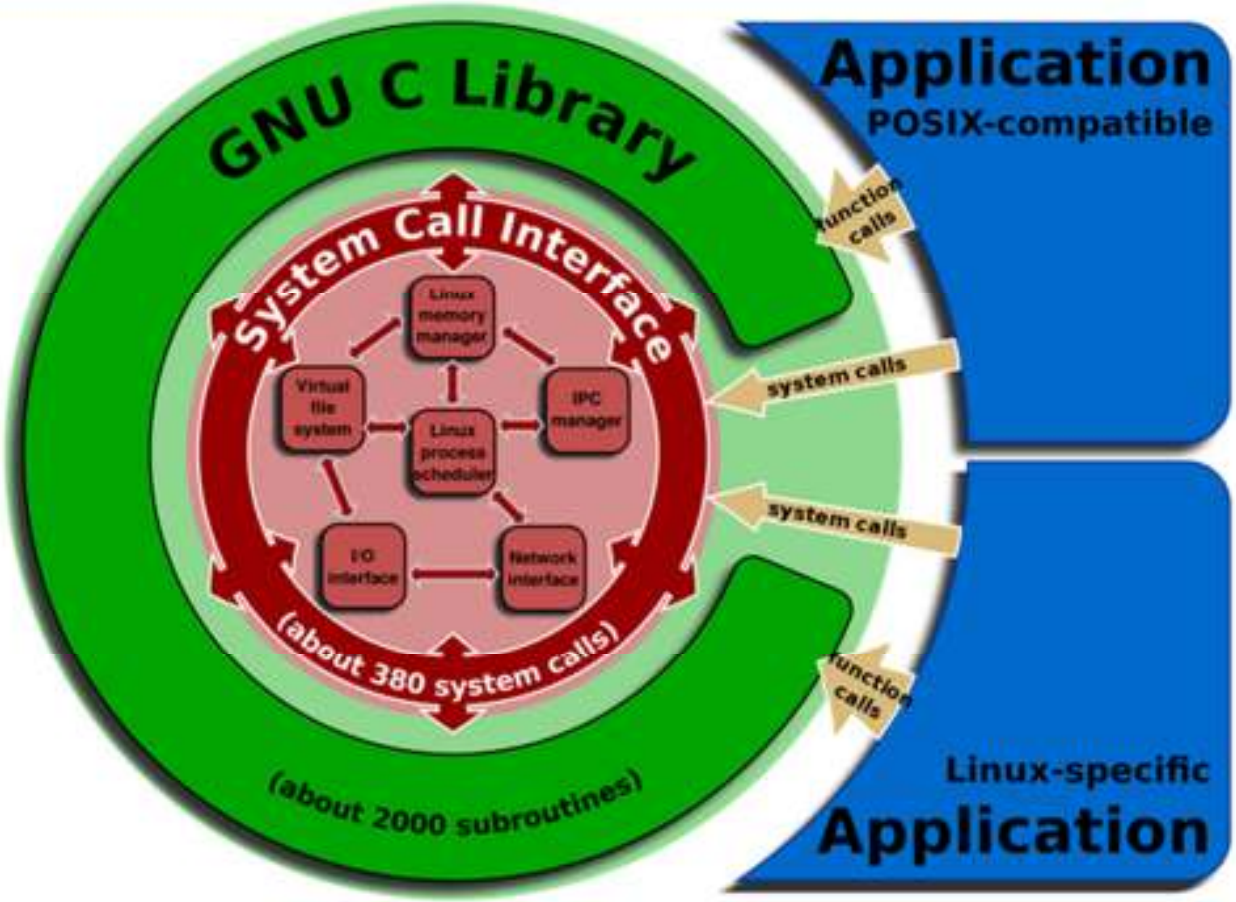


User Applications

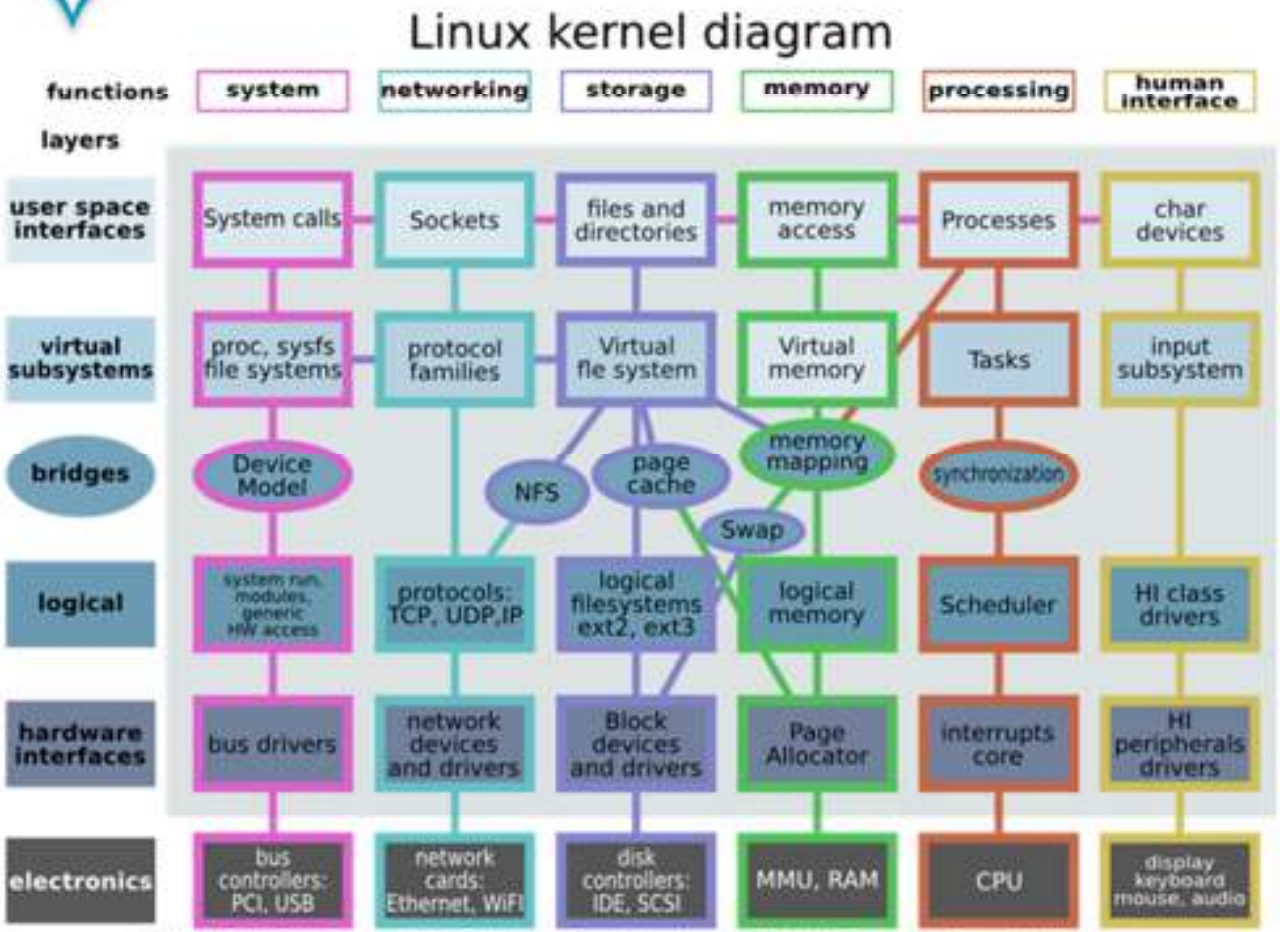
Applications used by end users for processing

OS Services

Services which enables user applications to interact with kernel

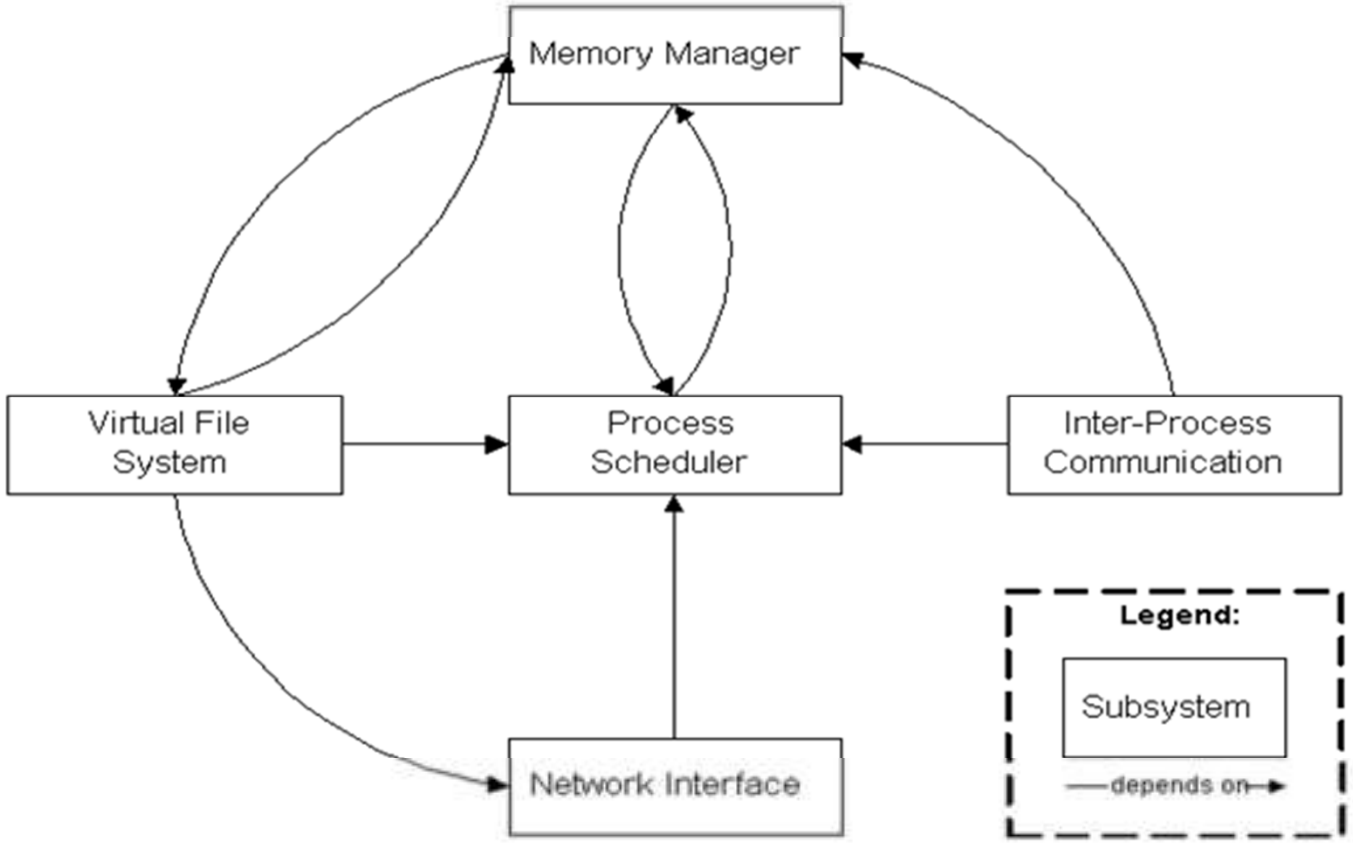


Linux Kernel



http://www.makelinux.net/kernel_map/

Linux Kernel (contd.,)



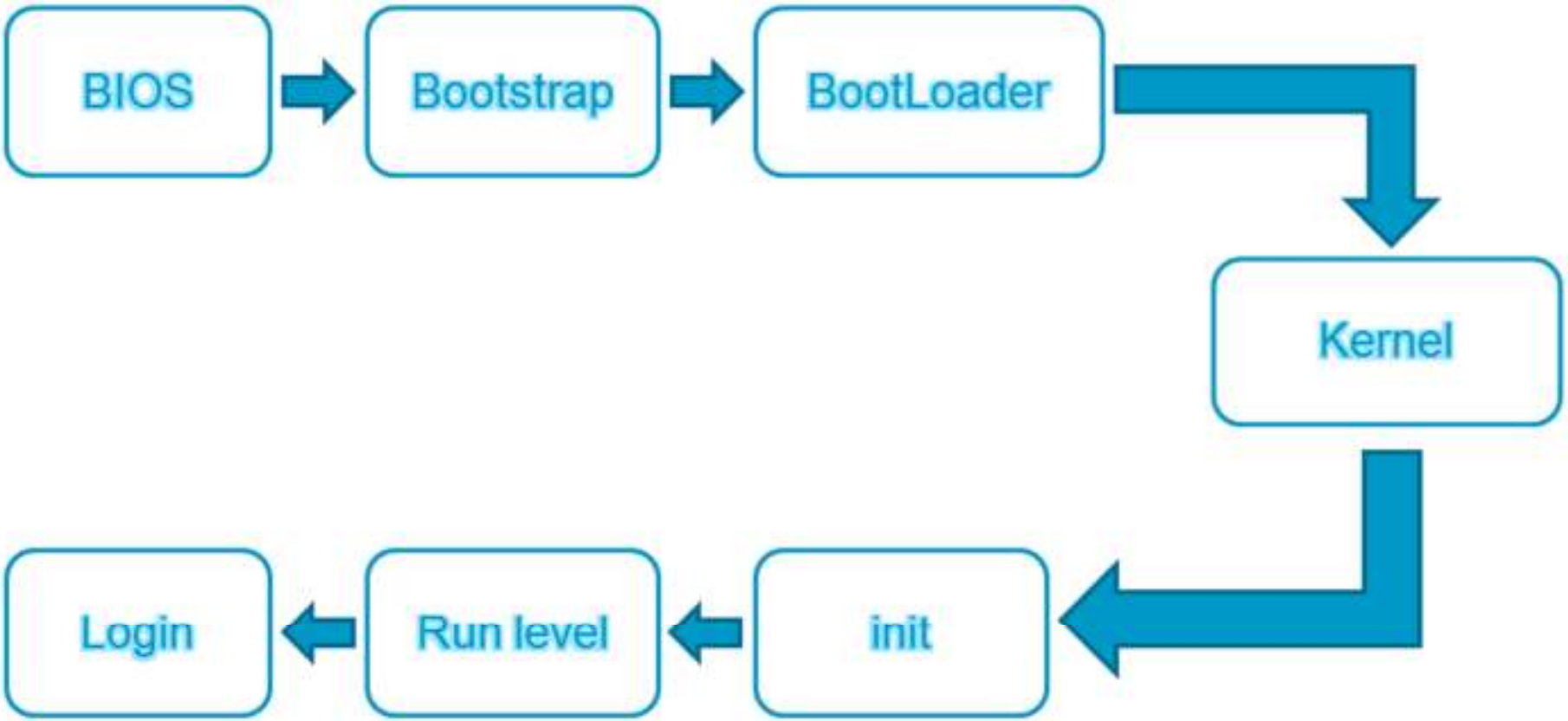
Hardware Controller

Devices – Controllers to take input from kernel

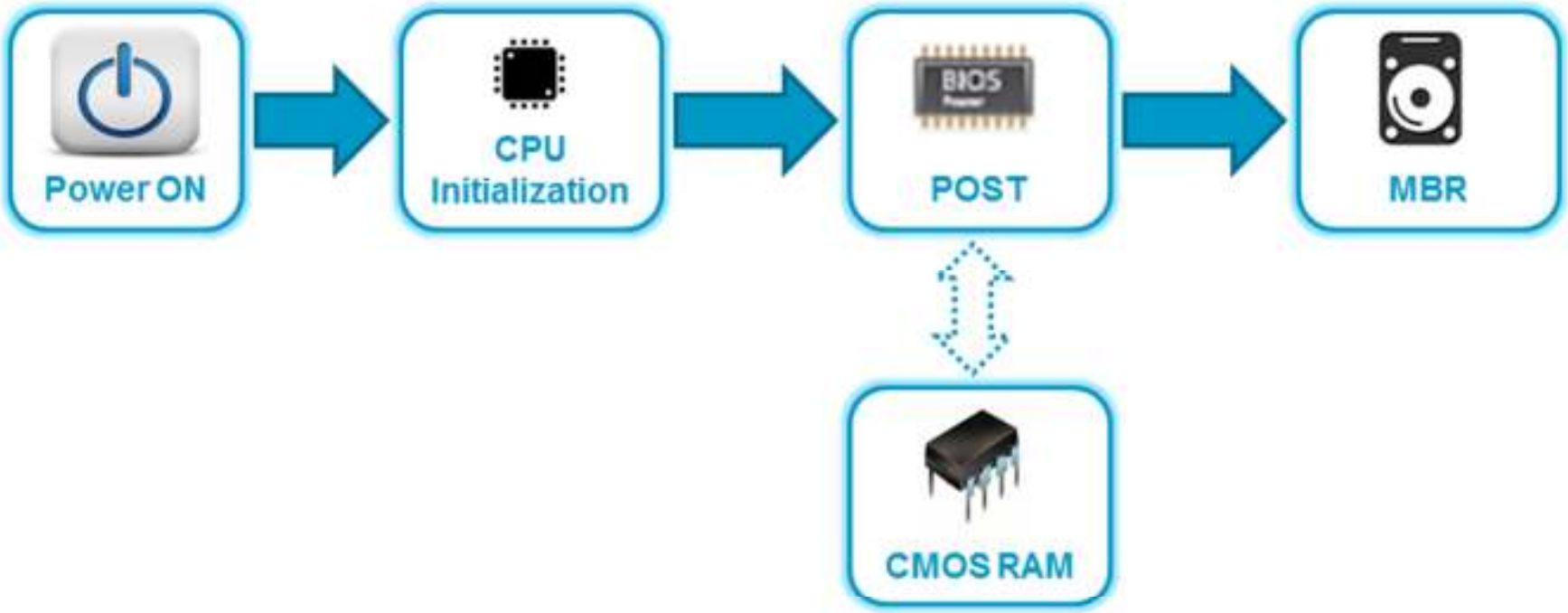
Kernel – Interfaces to connect and send commands

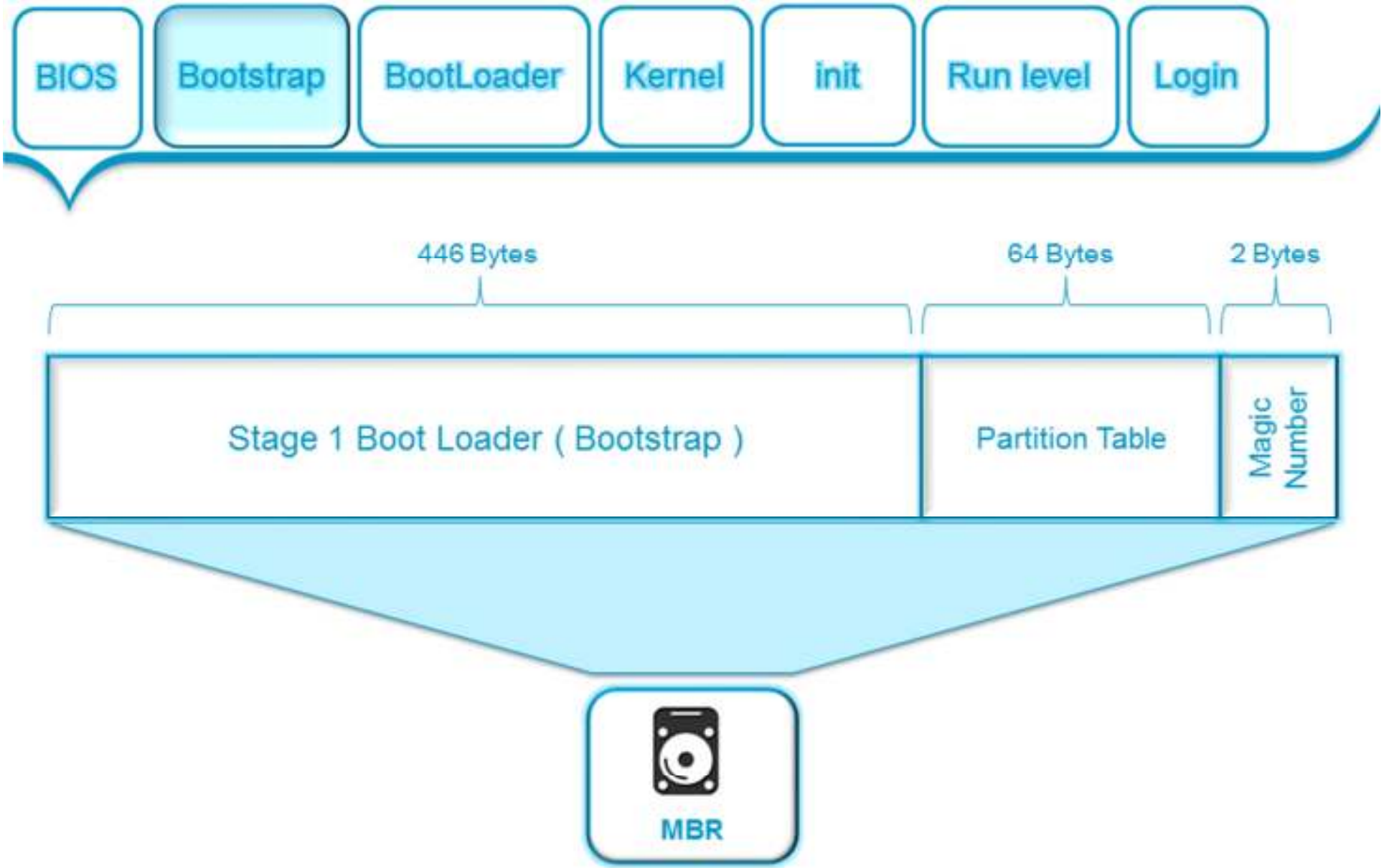
Booting Process

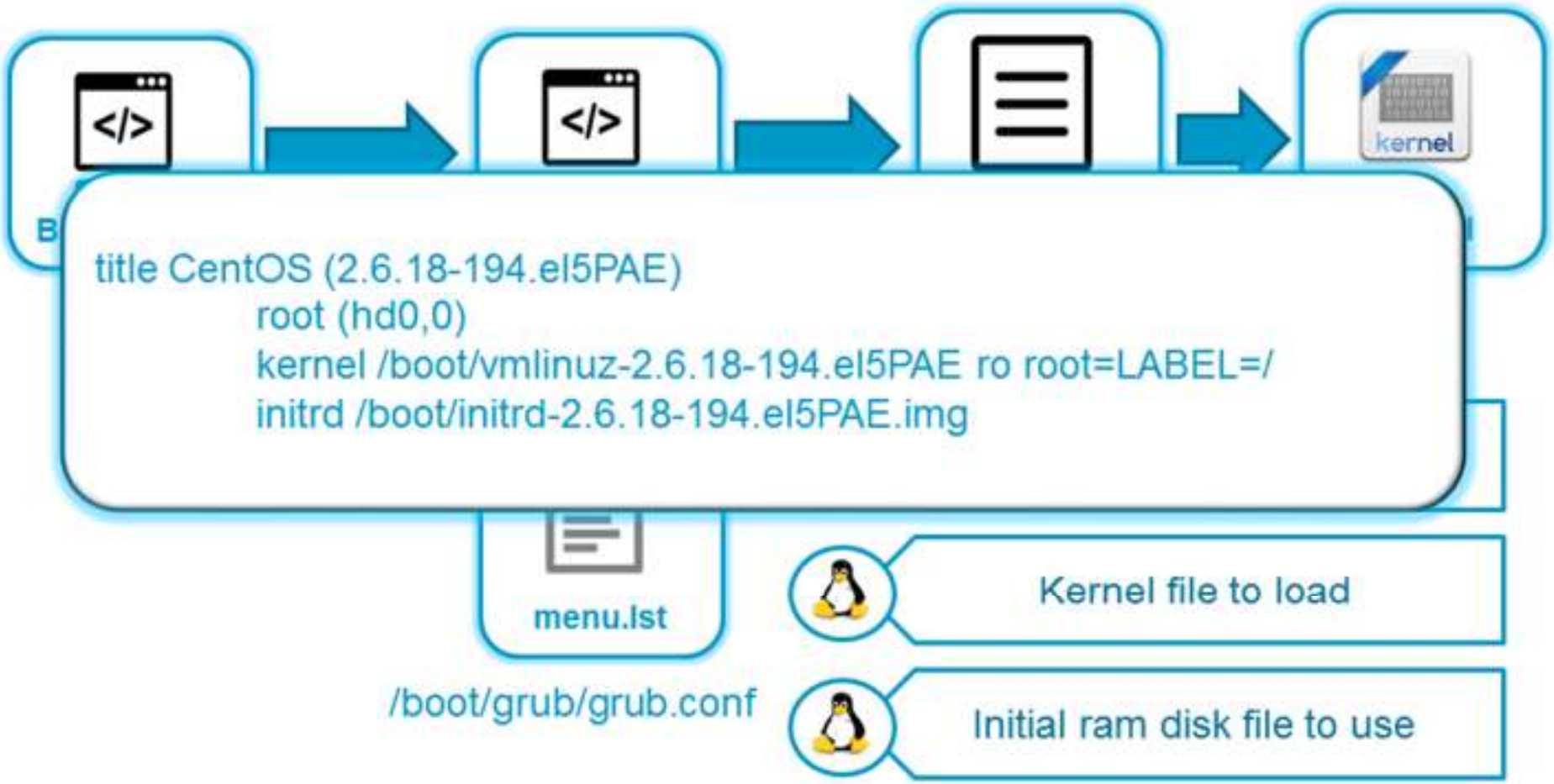
Booting Process

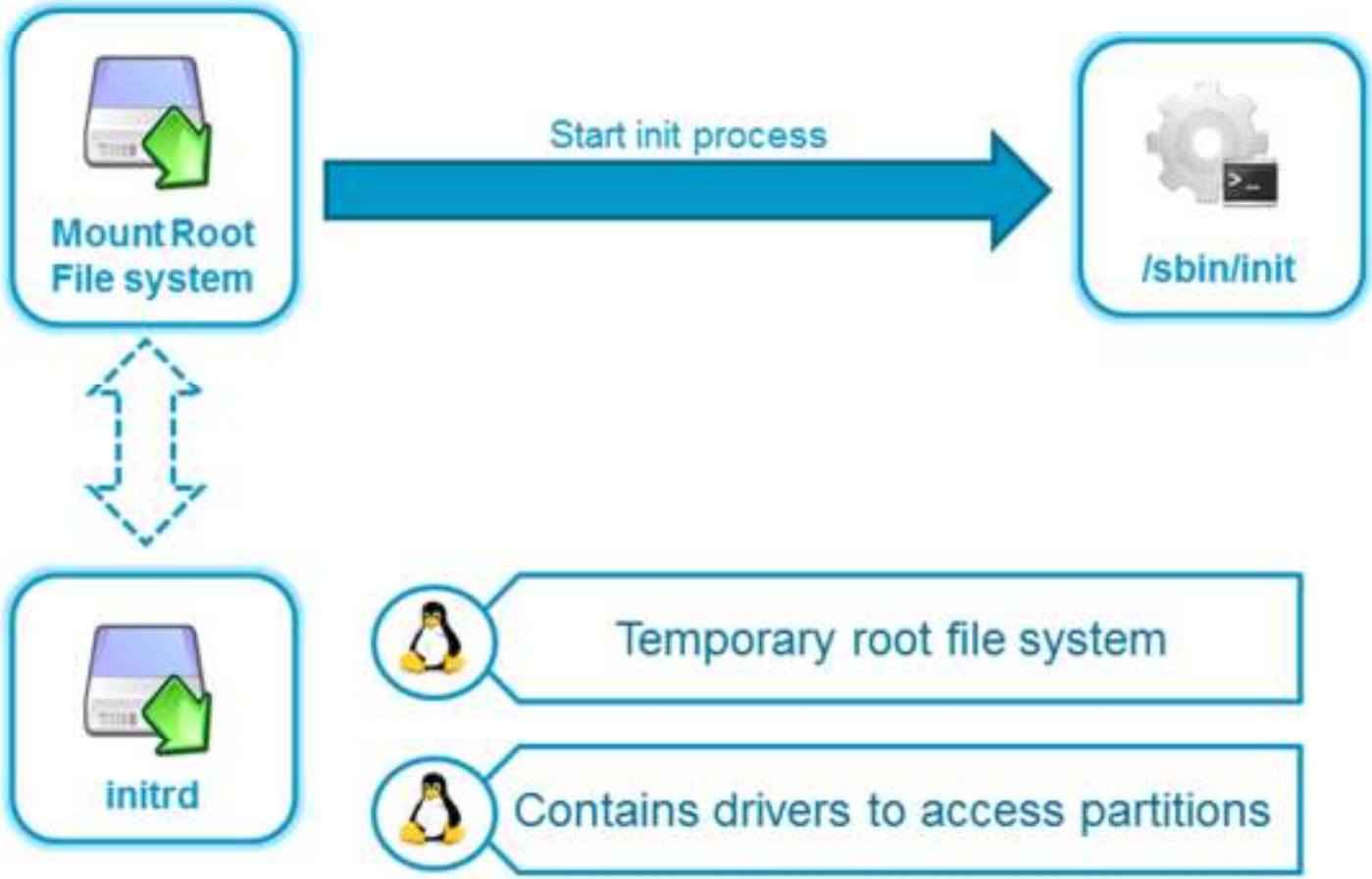


BIOS | Bootstrap | BootLoader | Kernel | init | Run level | Login





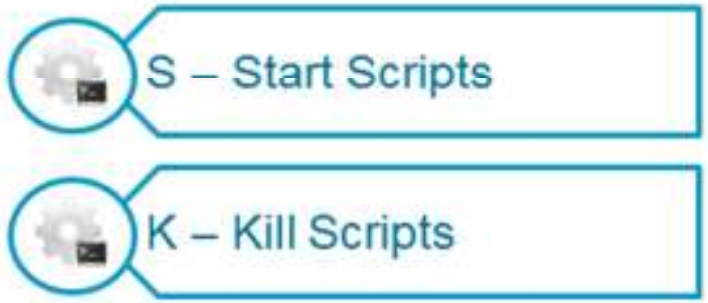




BIOS | Bootstrap | BootLoader | Kernel | **init** | Run level | Login

- First process started with pid 1
- Uses /etc/inittab file to identify runlevel

BIOS | Bootstrap | BootLoader | Kernel | init | **Run level** | Login



BIOS | Bootstrap | BootLoader | Kernel | init | Run level | Login

/sbin/ini

```
VMware ESX Server 3 (Dali)
Kernel 2.4.21-37.0.2.ELvmnix on an i686

localhost login: root
Password:
Last login: Tue Apr 17 22:06:17 on tty1
[root@localhost root]#
```

/etc/passwd
/etc/shadow

Shell

Linux Shell

- **Unix Shell :**
  - A Unix shell is a command-line interpreter .
  - It provides a traditional user interface for the Unix operating system and for Unix-like systems.
  - Whatever you type at the command line is understood and interpreted by a program and then that program gives you an output after executing your command. This program that understands what you type is called the shell.
- **Types of Shells :**
  - Ash
  - Bash
  - Korn
  - T - Shell
  - C- Shell
  - Z- Shell

Linux Shell – Environment Variables

- Environment variables are the built-in variables which contains a data that can be shared by 2 or more applications/processes.

```
HOSTNAME=puppet
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
XDG_SESSION_COOKIE=71e301d6aadadae31b5adedc0000001d-1450963092.780047-163783744
GTK_RC_FILES=/etc/gtk/gtkrc:/root/.gtkrc-1.2-gnome2
WINDOWID=44040195
USER=root
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=01;05;37;41:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=3
4;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=
01;31:*.gz=01;31:*.lz=01;31:*.xz=01;31:*.bz2=01;31:*.tbz=01;31:*.tbz2=01;31:*.bz=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.rar=01;31:*.ace=01;31
:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:
*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;3
5:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35
:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.axv=01;35:*.anx=01;35:*.ogv=01;35:*.ogx=01;35:*.
aac=01;36:*.au=01;36:*.flac=01;36:*.mid=01;36:*.midi=01;36:*.mka=01;36:*.mp3=01;36:*.mpc=01;36:*.ogg=01;36:*.ra=01;36:*.wav=01;36:*.axa=01;36:*.oga=01;36:*.s
px=01;36:*.xspf=01;36:
GNOME_KEYRING_SOCKET=/tmp/keyring-2PUt4L/socket
SSH_AUTH_SOCK=/tmp/keyring-2PUt4L/socket.ssh
SESSION_MANAGER=local/unix:@/tmp/.ICE-unix/2236,unix/unix:/tmp/.ICE-unix/2236
USERNAME=root
PATH=/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/root/bin
MAIL=/var/spool/mail/root
DESKTOP_SESSION=gnome
PWD=/root
GDM_KEYBOARD_LAYOUT=us
GNOME_KEYRING_PID=2226
LANG=en_US.UTF-8
GDM_LANG=en_US.UTF-8
GDMSESSION=gnome
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
HISTCONTROL=ignoredups
HOME=/root
SHLVL=2
```

Linux Shell – Environment Variables

- Some of the important environment variables are:
  - PWD

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin/X1
```

  - SHELL

```
sh-4.3$ echo $SHELL
/bin/bash
```

  - PATH

```
root@tuxworld:~# echo $PWD
/root
```

- SETTING ENVIRONMENT VARIABLES:

```
sh-4.3$ export PATH=$PATH:/home
sh-4.3$ echo $PATH
/home/cg/root/GNUstep/Tools:/usr/GNUstep/Local/Tools:/usr/GNUstep/System/Tools:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/h
i/bin/extra:/usr/local/fantom/bin:/usr/local/dart/bin:/usr/bin:/usr/local/bin:/usr/local/sbin:/usr/sbin:/opt/mono/bin:/opt/mono/lib
7.0_75/bin:/opt/jdk1.7.0_75/jre/bin:/opt/pash/Source/PashConsole/bin/Debug/:/home
```

Commands for Practice

1 alias
2 aspell
3 at
4 atq
5 cal
6 cat
7 cd
8 cp
9 cut
10 date
11 dd
12 df
13 diff
14 dmesg
15 echo
16 eject
17 emacs
18 fdisk
19 find
20 get
21 grep
22 Halt
23 head
24 History
25 hostname
26 ifconfig
27 ll
28 ln
29 locate
30 login
31 logout
32 ls
33 mkdir
34 mke2fs
35 more
36 mount
37 mv
38 poweroff
39 put
40 Pwd
41 reboot
42 rm
43 scp
44 sed
45 sh
46 shutdown
47 sort
48 su
49 tail
50 touch
51 umount
52 uname
53 uniq
54 uptime
55 vi
56 vim
57 wc
58 whereis
59 which
60 who
61 whoami

Process Management

Linux Process Management

- PROCESS:
  - An instance of a program is called a Process. In simple terms, any command that you give to your Linux machine starts a new process.

```
$ps
PID       TTY       TIME        CMD
18358     ttyp3     00:00:00    sh
18361     ttyp3     00:01:31    abiword
18789     ttyp3     00:00:00    ps
```

- The operating system tracks processes through a five digit ID number known as the **pid** or process ID . Each process in the system has a unique pid.

Linux Process Management (Contd…)

TYPES OF PROCESS:

- **Child Process:** A process that is created by some other process during run-time. Usually child processes are created to execute some binary from within an existing process.
- **Daemon Process:** These are special processes that run in background. They are system related process that have no associated terminal.
- **Orphan Process:** When parent process gets killed the child processes become orphan and then taken under by the init process. Though the init process takes the ownership of the orphan process but still these process are called as orphan as their original parents no longer exists.
- **Zombie Process :** A zombie process is one that should have closed, but is still active in the process table. This is usually caused when a parent process that spawned the process has not yet realized that it has completed, or wants to create another process of the same name without using the same process ID.
- **Interactive process :** These interact constantly with their users, and therefore spend a lot of time waiting for key presses and mouse operations.
- **Batch or Automatic process :** These do not need user interaction, and hence they often run in the background.
- **Real time process :** These have very strong scheduling requirements. They should have a short response time and, most important, such response time should have a minimum variance. Typical real-time programs are video and sound applications.

Linux Process Management

Process Scheduling :

- The scheduler is the component of the kernel that selects which process to run next.
- Scheduling refers to the way processes are assigned to run on the available CPUs

Functionality of Scheduler :

- **CPU utilization**: To keep the CPU as busy as possible.
- **Throughput** : Number of processes that complete their execution per time unit.
- **Turnaround** : Total time between submission of a process and its completion.
- **Waiting time** : Amount of time a process has been waiting in the ready queue.
- **Response time**: Amount of time it takes from when a request was submitted until the first response is produced.
- **Fairness** : Equal CPU time to each thread.

Linux Process Management (Contd…)

Types of Scheduling :

- **Normal** : Referred to as other, this is the scheduling type set for normal programs
- **FIFO :** This is a real time scheduling priority. The FIFO term means the first started (first in) will be the first done (first out).
- **RR :** This is a round robin type of scheduling, where each task gets a certain amount of time then it must exit, yield control to the next task and get back into the task queue. This is a real time scheduling priority.
- **Priority Based Scheduling** : Assigns each process a priority, and scheduler always chooses process of higher priority over one of lower priority .
- **Shortest Job First(SJF) :** Itis a non-preemptive discipline in which waiting job (or process) with the smallest estimated run-time-to-completion is run next

Linux Process Management (Contd…)

Inter Process Management :

- It is a set of techniques for the exchange of data among multiple threads in one or more processes.

Types of Inter Process Management :

- **Signals :** Sent by other processes or the kernel to a specific process to indicate various conditions.
- **Pipes :** Unnamed pipes set up by the shell normally with the "|" character to route output from one program to the input of another.
- **Sockets:** A **socket** is one endpoint of a two-way communication link between two programs running on the **network**.

Linux Process Management (Contd…)

- **Message queues :** Message queues are a mechanism set up to allow one or more processes to write messages that can be read by one or more other processes.
- **Semaphores :** Counters that are used to control access to shared resources. These counters are used as a locking mechanism to prevent more than one process from using the resource at a time.
- **Shared memory :** The mapping of a memory area to be shared by multiple processes

File System

Linux File System

- **What is a File?**

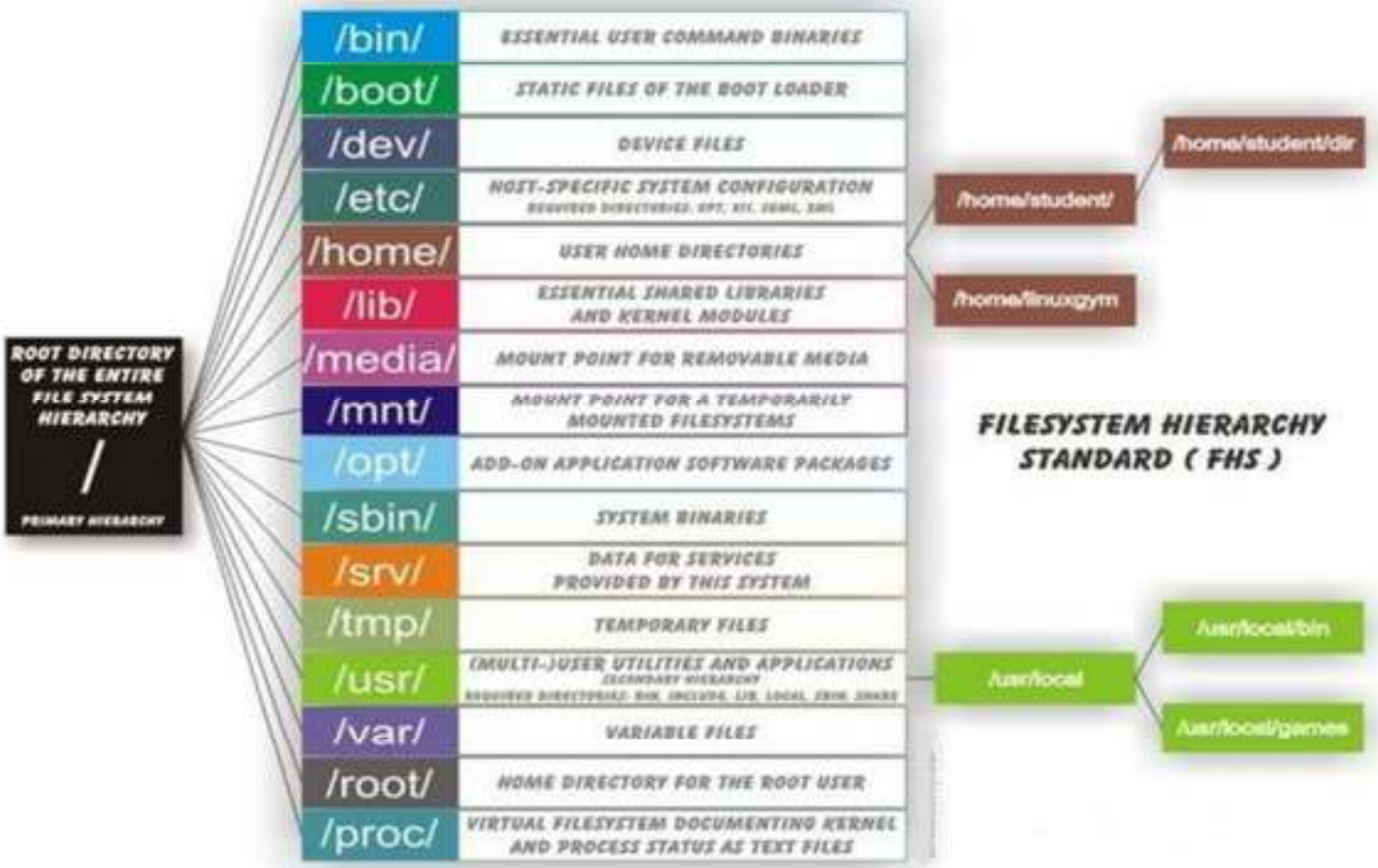
  File is a collection of data items stored on disk. Or it's device which can store the information, data, music (mp3), picture, movie, sound, book etc. In fact what ever you store in computer it must be inform of file. Files are always associated with devices like hard disk ,floppy disk etc. File is the last object in your file system tree.

- **What is a directory?**

  Directory is group of files

  - Root directory - It is root of your entire file system and can not be renamed or deleted which is denoted by / (forward slash)
  - Sub directory - Directory under root (/) directory is subdirectory which can be created, renamed by the user.

Linux Directory Structure



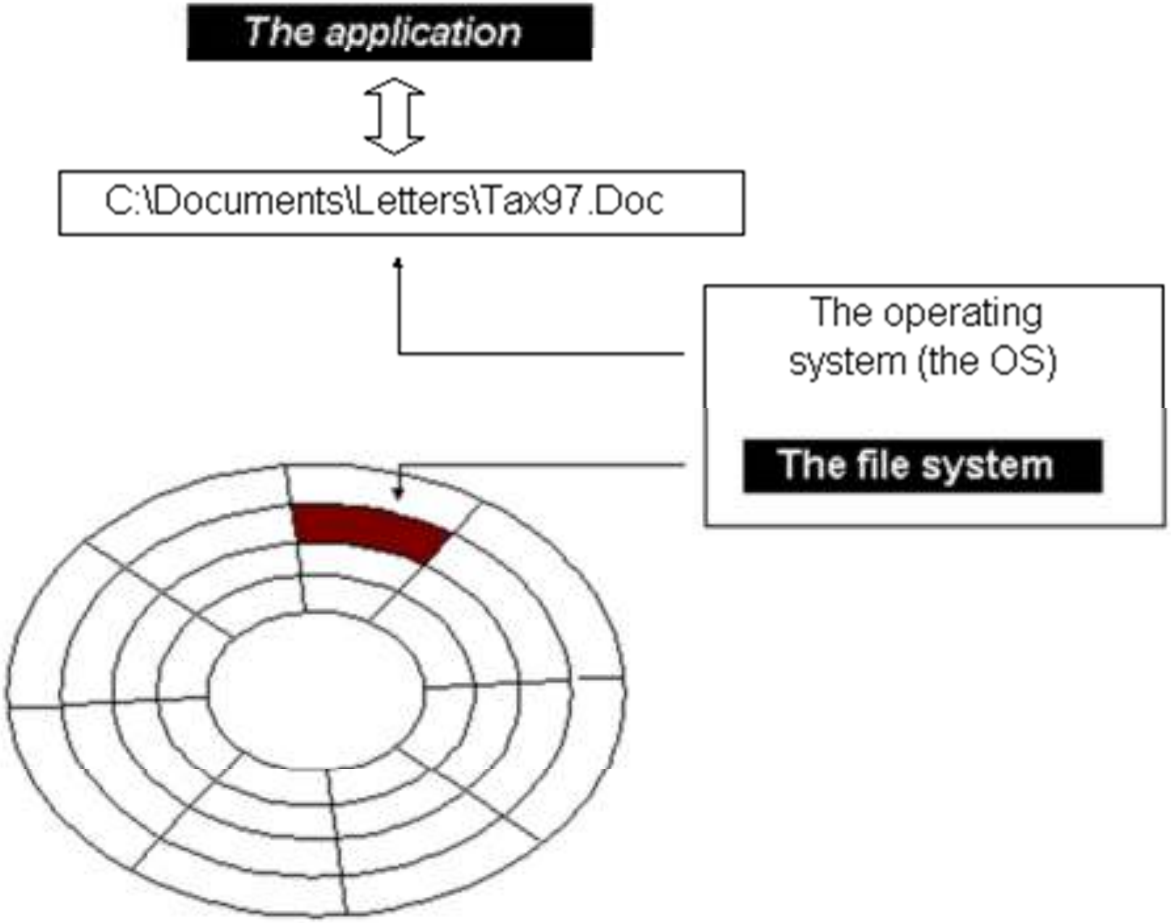
Types of Files

- Regular Files
- Directory Files
- Special Files
  - Block file
  - Character Device File
  - Named Pipe File
  - Symbolic File
  - Socket File

```
[root@localhost /]# ls -l
total 98
dr-xr-xr-x.   2 root root  4096 Dec 25 19:03 bin
dr-xr-xr-x.   5 root root  1024 Dec 25 19:07 boot
drwxr-xr-x.  10 root root  4096 Dec 25 19:17 cgroup
drwxr-xr-x.  17 root root  3600 Dec 25 19:18 dev
drwxr-xr-x. 149 root root 12288 Dec 25 19:18 etc
drwxr-xr-x.   2 root root  4096 Dec  4  2009 home
dr-xr-xr-x.  22 root root 12288 Dec 25 19:03 lib
drwx------.   2 root root 16384 Dec 25 18:21 lost+found
drwxr-xr-x.   2 root root  4096 Dec  4  2009 media
drwxr-xr-x.   2 root root     0 Dec 25 19:18 misc
drwxr-xr-x.   2 root root  4096 Dec  4  2009 mnt
```

```
[root@localhost ~]# ls -a
.   anaconda-ks.cfg  .bash_profile  .cshrc
..  .bash_logout     .bashrc        install.log
```

Basics of File System



Types of File System

- EXT
  - The Extended file system is used to on the storage media like hard disks and default file system in linux
- CDFS :
  - A file system that is used on compact disks to provide access to individual data and audio tracks
- UFS
  - The Unix File System is used various versions of unix like BSD and Solaris

Disk Partitioning

- Disk partitioning is dividing the total storage of a drive into different small parts called partitions
- The partitions will be formatted with the specified file system so it can be used for storing the data

- **Advantages**:
  - Multiple File Systems
  - Partition Size
  - Multiple Operating Systems
  - Separate system files from users files

Disk Partitioning

- We can perform the disk partitioning by using the “fdisk” command

```
[root@localhost ~]# fdisk -l /dev/sdb

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x407889db

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        1044     8385898+   5  Extended
[root@localhost ~]#
```

```
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

Formatting the Partition

- Formatting is the process of preparing a data storage device such as a hard disk drive, solid-state drive, floppy disk or USB flash drive for initial use

- Types of Formatting:

Hard Formatting	Soft Formatting
• Forming the tracks and sectors on the device • Done at manufacturer	• Creating the file system on the device to store the data • Done at OS level

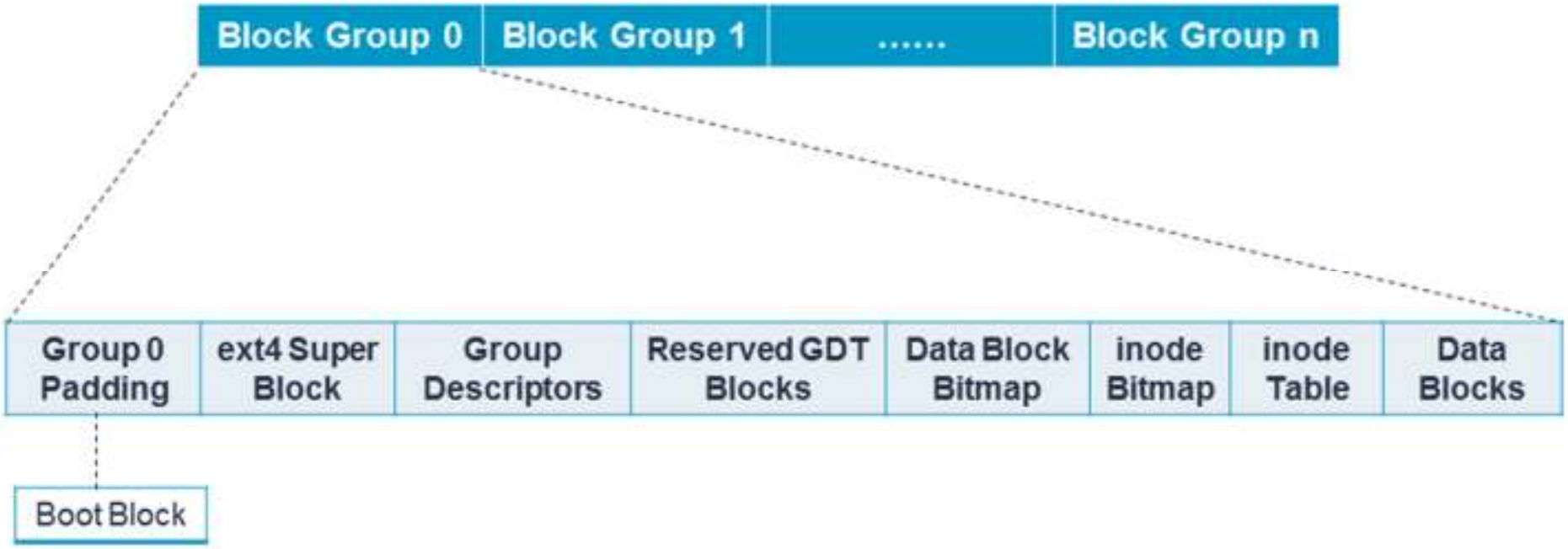
Formatting a drive using EXT4 file system

```
[root@localhost ~]# mkfs.ext4 /dev/sdb5
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
524288 inodes, 2096466 blocks
104823 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2147483648
64 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 30 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[root@localhost ~]#
```

EXT4 File System

- Fourth extended file system(EXT4) developed as successor of Ext3
- An Ext4 file system is split into a series of block groups
- Ext4 uses extents which improves performance when using large files and reduces metadata overhead for large files

Structure of an EXT4 File System



Reference : https://ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout

Blocks

- A block is a group of sectors between 1KiB and 64KiB
- The number of sectors must be an integral power of 2
- Default block size is 4096 bytes
- By default a file system can contain
  - if 32-bit : 2^32 blocks
  - if 64-bit : 2^64 blocks

- We can set the block size while creating the file system

  $ mkfs.ext4 -b 4096 /dev/sdb1

```
[root@localhost ~]# blockdev --getbsz /dev/sda3
4096
```

Check the Block size

Block Groups

- Blocks are grouped into larger units called block groups
- The block allocator tries very hard to keep each file's blocks within the same group, which reduces seek times

- Number of blocks per Block Group = 8 * block_size in bytes

- Block Group size = (Number of blocks) * block_size

- Ex: By considering the default block size 4096 bytes

(8 * 4096) * 4096 = 128 MiB

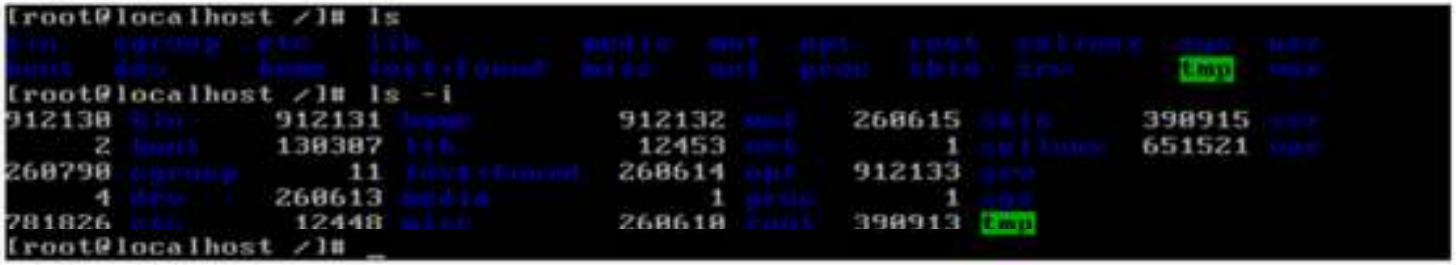
Block Groups

- **Boot Block:**
  - The boot block will maintain the boot sectors information
- **Super Block:**
  - The super block records various information about the enclosing file system, such as block counts, inode counts, supported features, maintenance information, and more
- **Group Descriptors:**
  - The standard configuration is for each block group to contain a full copy of the block group descriptor table
  - The group descriptor records the location of both bitmaps and the inode table
- **Data Block Bitmap:**
  - The data block bitmap tracks the usage of data blocks within the block group
  - One bit represents the usage status of one data block

Block Groups

- **Inode Bitmap:**
  - **inode** is a data structure used to represent a file system object
  - The inode bitmap records which entries in the inode table are in use
  - One bit represents the usage status of one inode table entry

- **Inode Table:**
  - Inode table will have a list of inodes. The inodes are placed in several tables, each of which contains the same number of inodes and is placed at a different blocks group
  - Each inode table is accessed from the group descriptor of the specific blocks group



- **Data Blocks:**
  - The data blocks will contain the actual contents of files

Allocation of Blocks

- Ext4 first uses multi-block allocator
  - When a file is first created, the block allocator allocates 8KiB of disk space to the file
  - When the file is closed, the unused allocations are freed, but if in case the space is fully used then the file data gets written out in a single multi-block extent
- Ext4 uses Delayed allocation
  - When a file needs more blocks for file writes, the file system defers deciding the exact location on the disk until all the dirty buffers are being written out to disk
  - Not committing to a particular location on disk until it's absolutely necessary is that the file system can make better location decisions
- Ext4 tries to keep a file's data blocks in the same block group as its inode. This cuts down the I/O operations time
- All the inodes in a directory are placed in the same block group as the directory, when feasible
- The disk volume is cut up into 128MB block groups these mini-containers are used to try to maintain data locality

Extents

- An extent is simply a set of blocks which are logically contiguous in a file system
- In Ext4 the file to logical block map has been replaced with an extent tree
- The inode must have the extents flag set for this feature to be used
- Storing the file structure as extents will result in significant compression of the file's metadata, since a single extent can replace a large number of block pointer
- The reduction in metadata will enable faster access

Journaling

- A journaling file system is a file system that maintains a special file called a journal that is used to repair any inconsistencies that occur during a system crash
- Journaling file systems store metadata or data or both based on the changes done, before writing the actual data to the hard disk

```
[root@localhost /]# debugfs -R features /dev/sdb5
debugfs 1.41.12 (17-May-2010)
Filesystem features: has_journal ext_attr resize_inode dir_index filetype extent
 flex_bg sparse_super large_file huge_file uninit_bg dir_nlink extra_isize
[root@localhost /]#
```

Checking whether the journal is enabled

Types of Journaling

- WriteBack
  - Only the metadata is journaled and data is written to the file on the disk
  - File system recovery is the worst, but the performance is the best
- Ordered
  - This type is the reverse of writeback. The physical data is written first before the metadata is journaled
  - File system recovery is medial.
- Journal
  - The metadata and file contents are journaled.
  - System performance can be poorer than the other two modes, but the fault tolerance is much better.

Device Management

Linux Device Management

What is Device Files?

- Under Linux each and every hardware device treated as a file. A device file allows to accesses hardware devices so that end users do not need to get technical details about hardware.
- In short, a device file (also called as a special file) is an interface for a device driver that appears in a file system as if it were an ordinary file. This allows software to interact with the device driver using standard input/output system calls, which simplifies many tasks.

Types of Device Files:

- **Block Device :** A block device is one that is designed to operate in terms of the block. A block device has an associated block device driver that performs I/O by using file system block-sized buffers from a buffer cache supplied by the kernel

- **Character Device:** A character device is any device that can have streams of characters read from or written to it. A character device has a character device driver associated with it that can be used for a device such as a line printer that handles one character at a time.

Linux Device Management

In Linux all devices were considered as files

File	Device
/dev/fd0	Floppy disk
/dev/hda0	IDE Hard drive 1, partition 0
/dev/hdb3	IDE Hard drive 2, partition 3
/dev/sda	First SCSI hard drive
/dev/cdrom	CD ROM drive This device may be on the secondary controller as a master (/dev/hdc) or slave (/dev/hdd).
/dev/mouse	May be a pointer to /dev/psaux which is the ps2 device or /dev/cua which is a serial device or /dev/ttyS0
Disk Drives	
/dev/hda	primary IDE master
/dev/hdb	primary IDE slave
/dev/hdc	secondary IDE master
/dev/hdd	secondary IDE slave

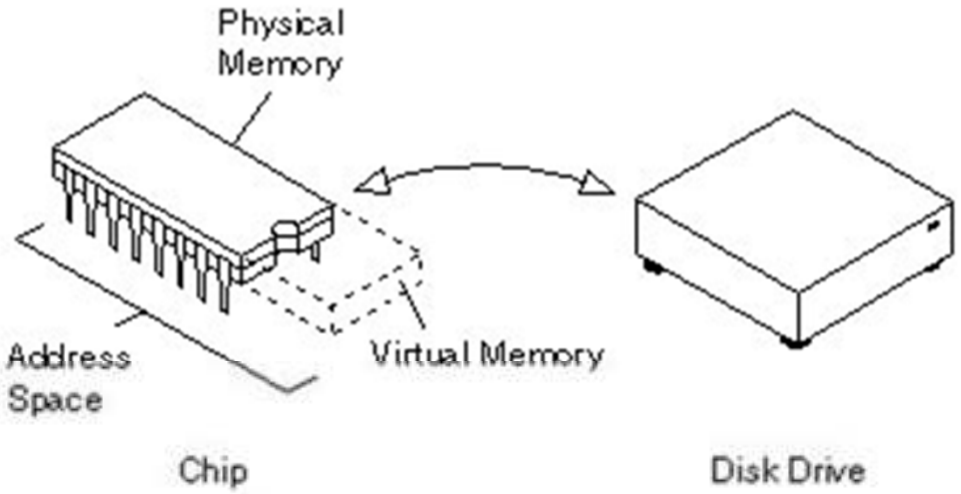
Linux Device Management

- **IRQ (Interrupt Service Request):**
  - IRQs are hardware lines over which devices can send interrupt signals to the microprocessor.
  - It is important to assign different IRQ addresses to different hardware devices is because the interrupt request signals run along single IRQ lines to a controller. This interrupt controller assigns priorities to incoming IRQs and sends them to the CPU.
  - The interrupt controller can control only one device per IRQ line, so if we assign the same IRQ address to multiple devices, we are likely to get an IRQ conflict

Memory Management

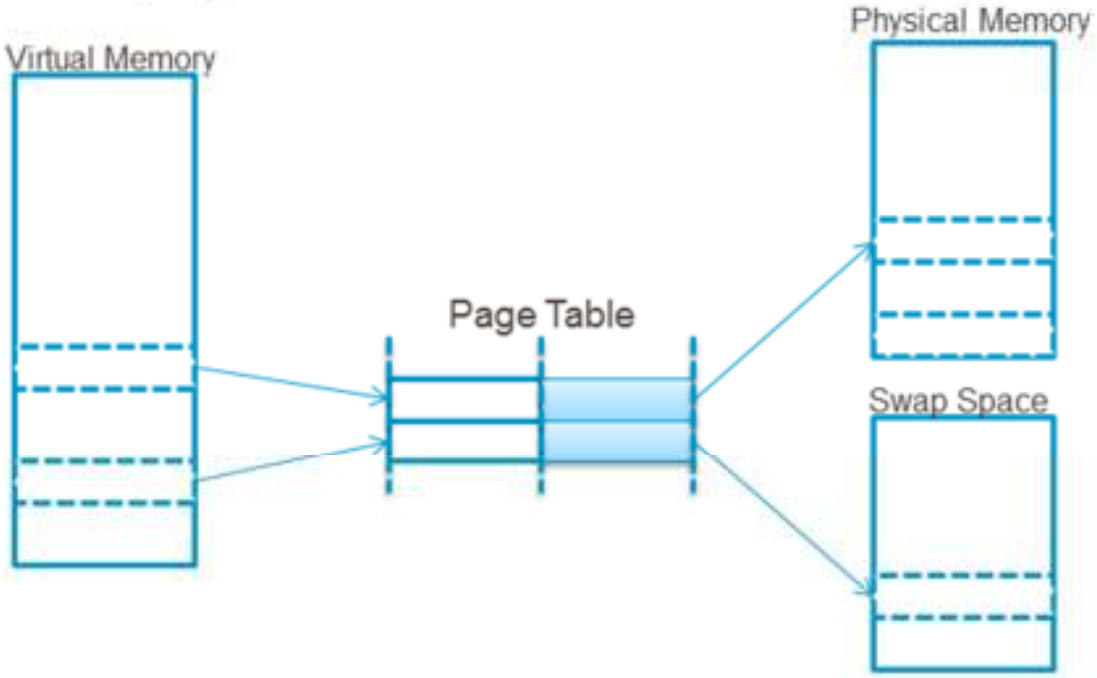
Memory Management

- Physical memory refers to the RAM installed in the system
- Virtual Memory extends the available memory of the computer by storing the inactive parts of the content RAM on a disk

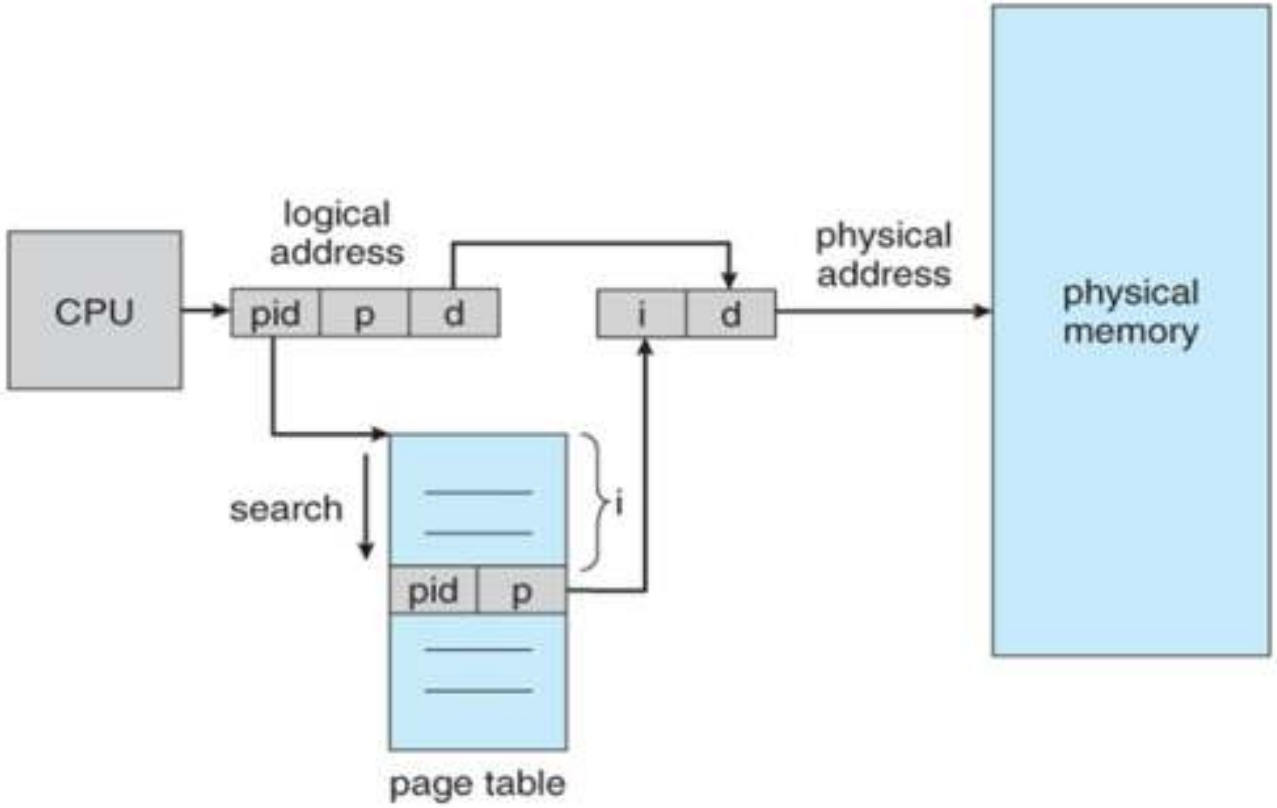


Memory Management

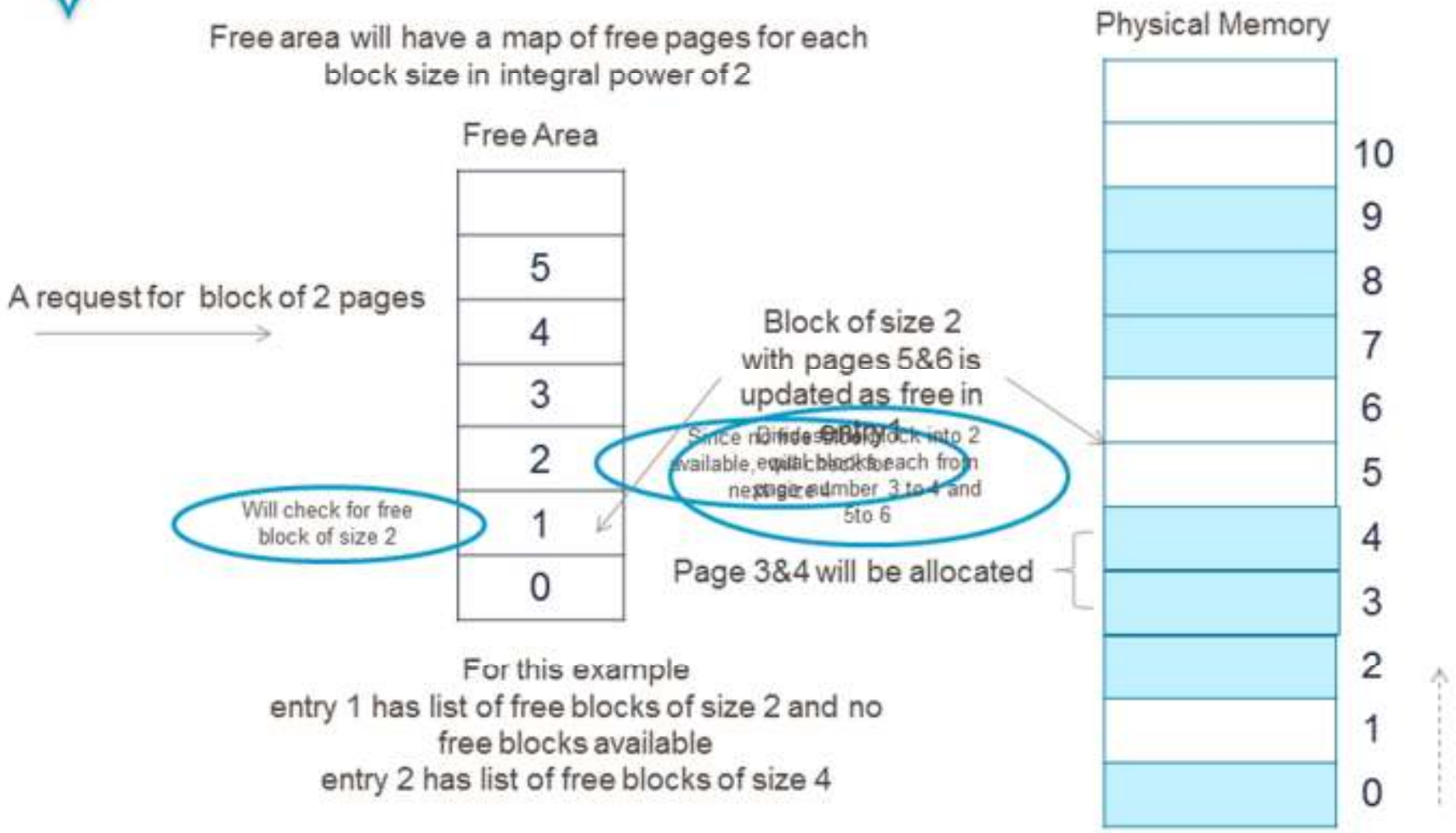
- Page : Memory is divided into chunks of equal size called pages
- Page Table: Stores the mapping between virtual addresses and physical addresses



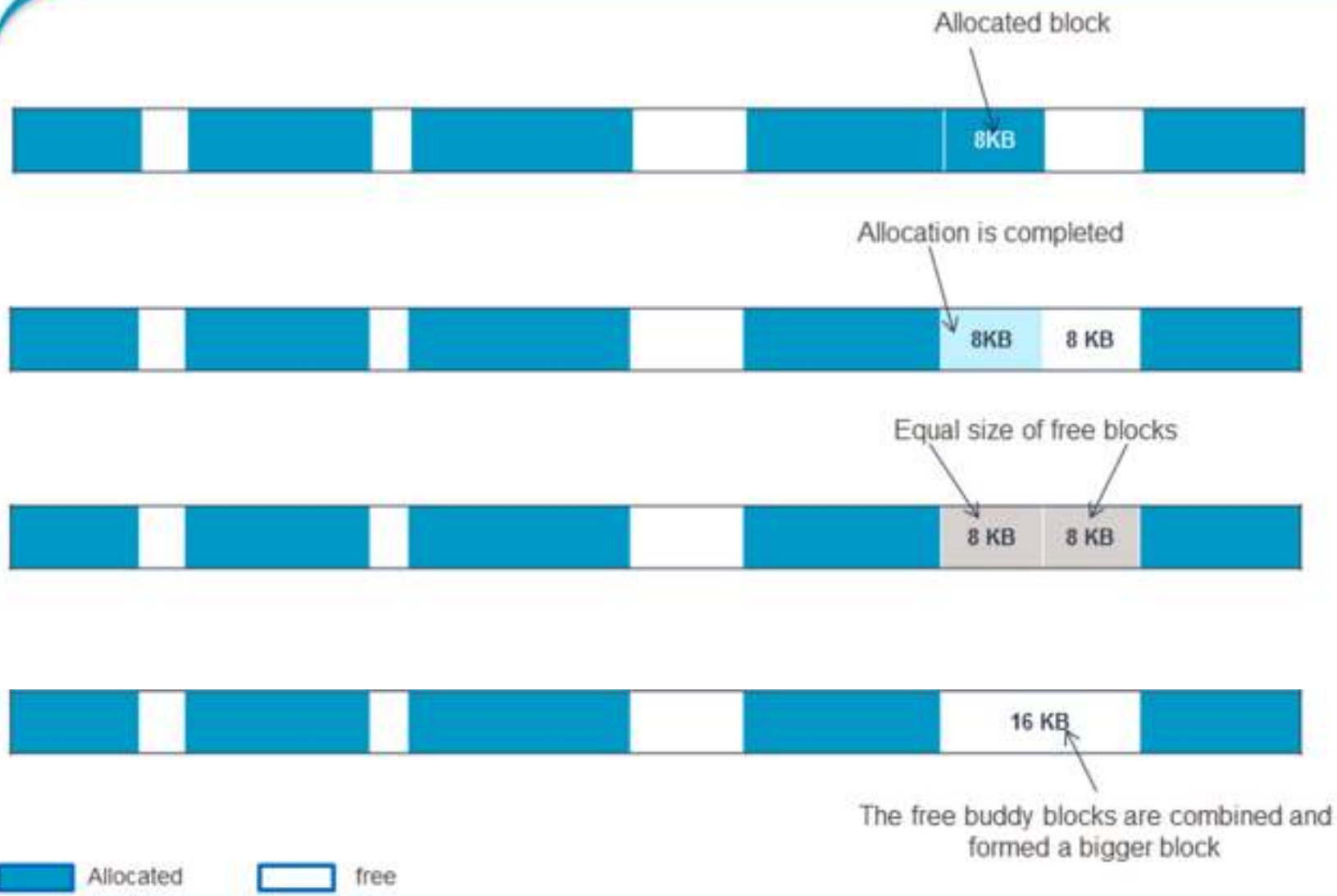
Mapping between physical and virtual address



Page Allocation



Page De-allocation



Swapping

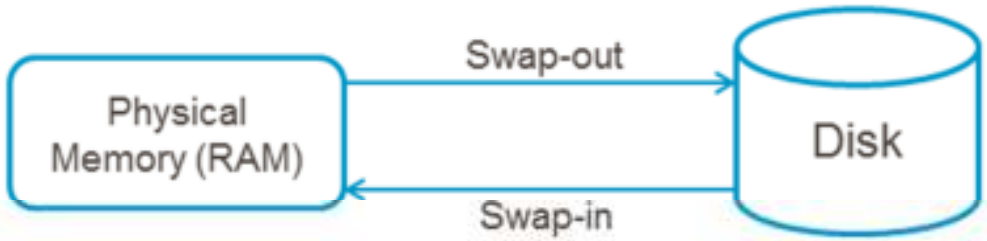
- Interchanging the pages between physical memory and disk is called swapping
- When a process requires more memory than available then swapping occurs by moving the in-active pages into disk
- When a Page fault occurs then the required page will be swap-in from disk to physical memory

```
[root@localhost ~]# grep SwapTotal /proc/meminfo
SwapTotal:       2097144 kB
[root@localhost ~]# vmstat
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 792632  69840  74484  438    0     0  1096  232  243  3  5 90  2  0
[root@localhost ~]# _
```

Check Swap Usage

Swapping

- Swap-Out:
  - The process of writing pages out from physical memory to disk to free memory is called swap-out
- Swap-In:
  - The process of loading pages in from disk to physical memory is called swap-in



Advantages of Virtual Memory

- More memory than physical RAM
- Multi Programming Environment
- More active processes
- Allocation of memory is cost-effective

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

People matter, results count.