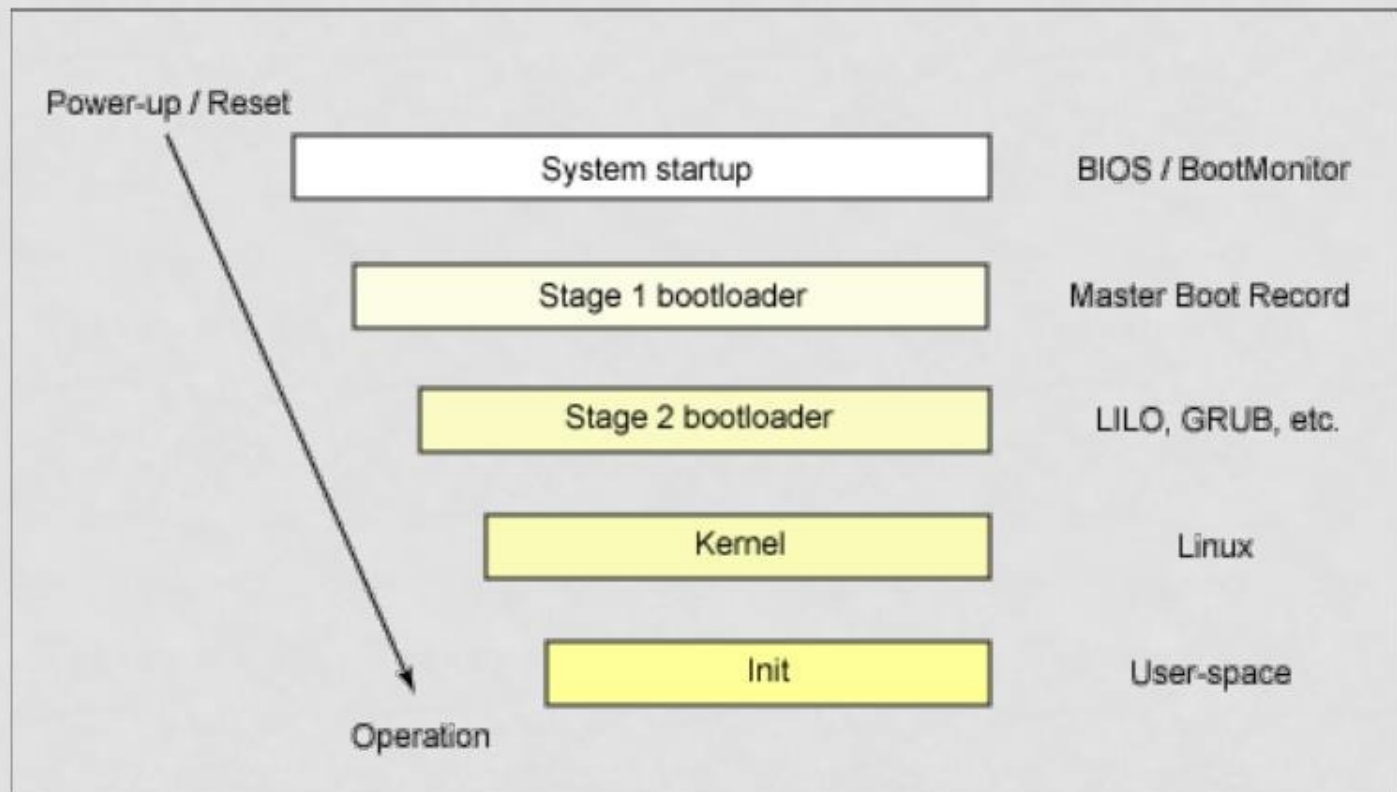


HOW LINUX BOOT?



SYSTEM STARTUP



HOW COMPUTER STARTUP?

- Booting is a bootstrapping process that starts operating systems when the user turns on a computer system
- A boot sequence is the set of operations the computer performs when it is switched on that load an operating system

BOOTING SEQUENCE

1. Turn on
2. CPU jump to address of BIOS (0xFFFF0)
3. BIOS runs POST (Power-On Self Test)
4. Find bootable devices
5. Load and execute boot sector from MBR
6. Load OS

BIOS (BASIC INPUT/OUTPUT SYSTEM)

- BIOS refers to the software code run by a computer when first powered on
- The primary function of BIOS is code program embedded on a chip that recognises and controls various devices that make up the computer.



BIOS on board



BIOS on screen



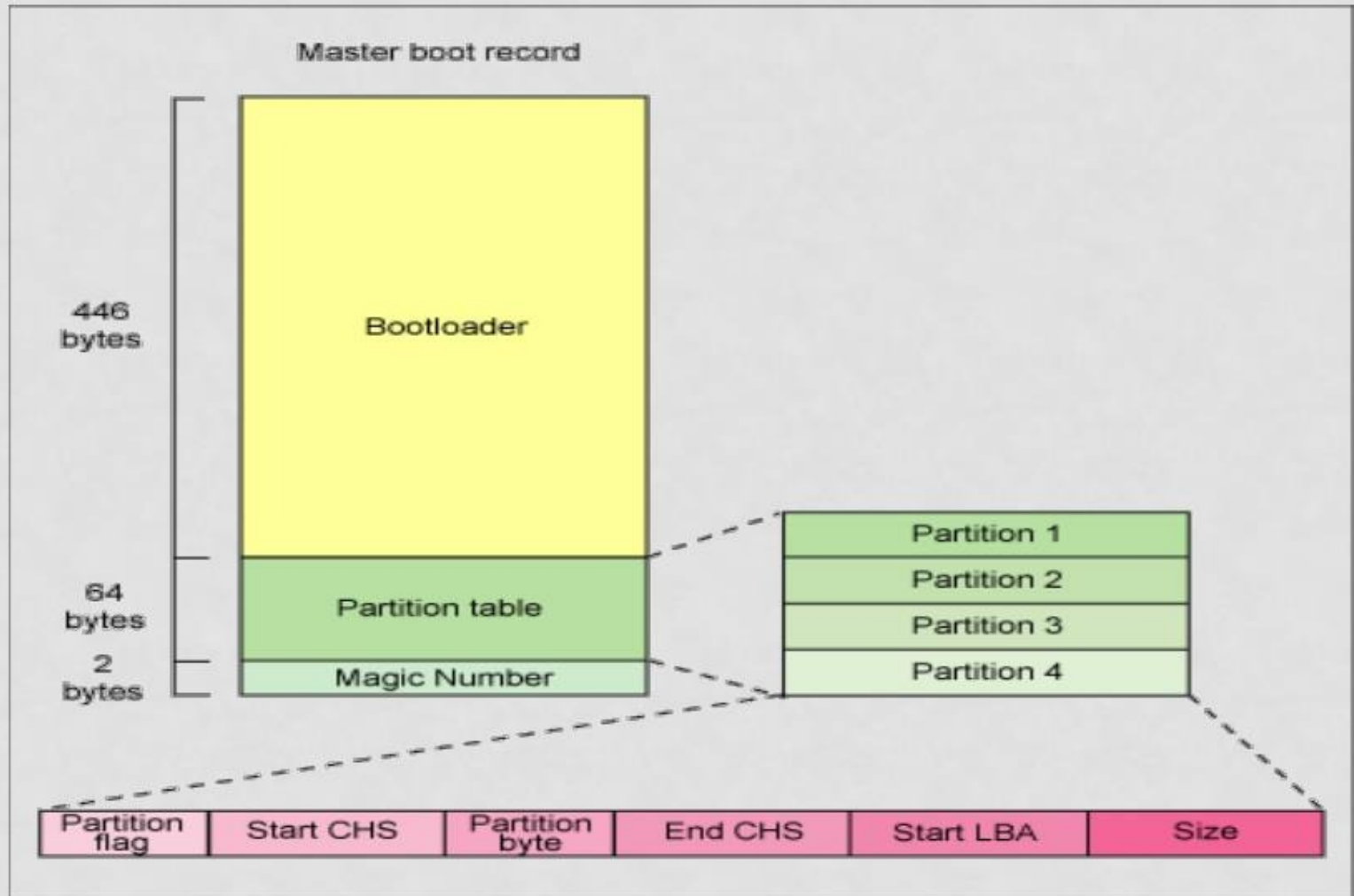
BOOT LOADER

The image shows a stylized representation of a floppy disk. The main body of the disk is a light gray rectangle. Inside this, there is a white rectangular area. Within the white area, the text 'BOOT LOADER' is centered in a dark gray, serif font. Below the text, there is a horizontal bar that is partially filled with a dark gray color, indicating a progress or status. To the right of the main white area, there is a vertical yellow rectangle, which represents the floppy disk's label or a specific data field.

MBR (MASTER BOOT RECORD)

- OS is booted from a hard disk, where the Master Boot Record (MBR) contains the primary boot loader
- The MBR is a 512-byte sector, located in the first sector on the disk (sector 1 of cylinder 0, head 0)
- After the MBR is loaded into RAM, the BIOS yields control to it.

MBR (MASTER BOOT RECORD)



MBR (MASTER BOOT RECORD)

- The first 446 bytes are the primary boot loader, which contains both executable code and error message text
- The next sixty-four bytes are the partition table, which contains a record for each of four partitions
- The MBR ends with two bytes that are defined as the magic number (0xAA55). The magic number serves as a validation check of the MBR

EXTRACTING THE MBR

- To see the contents of MBR, use this command:
 - `# dd if=/dev/hda of=mbr.bin bs=512 count=1`
 - `# od -xa mbr.bin`
- **The dd command, which needs to be run from root, reads the first 512 bytes from /dev/hda (the first Integrated Drive Electronics, or IDE drive) and writes them to the mbr.bin file.
- **The od command prints the binary file in hex and ASCII formats.

BOOT LOADER

- Boot loader could be more aptly called the kernel loader. The task at this stage is to load the Linux kernel
- Optional, initial RAM disk
- GRUB and LILO are the most popular Linux boot loader.

OTHER BOOT LOADER (SEVERAL OS)

- bootman
- GRUB
- LILO
- NTLDR
- XOSL
- BootX
- loadlin
- Gujin
- Boot Camp
- Syslinux
- GAG

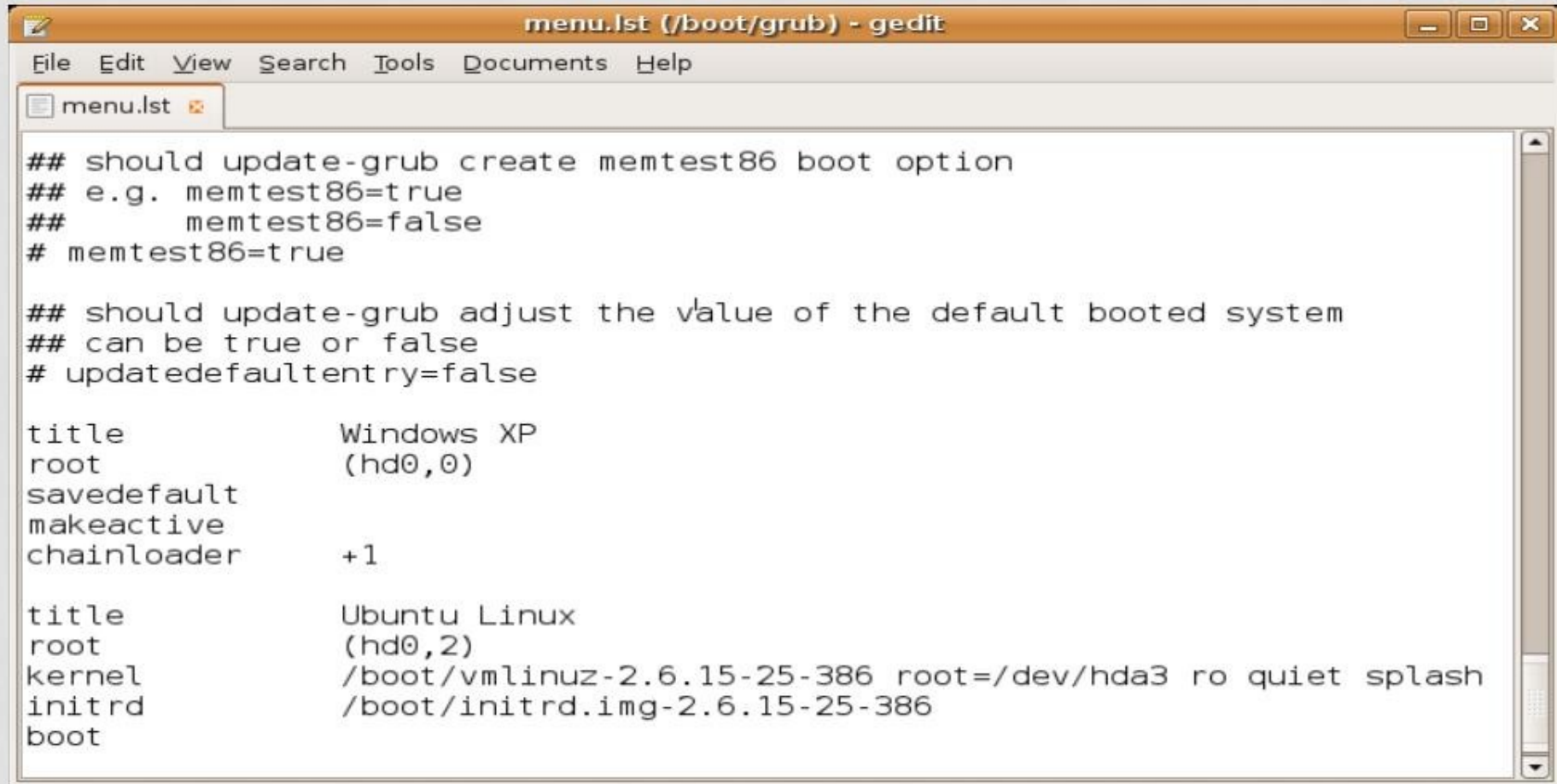
GRUB: GRAND UNIFIED BOOTLOADER

- GRUB is an operating system independent boot loader
- A multiboot software packet from GNU
- Flexible command line interface
- File system access
- Support multiple executable format
- Support diskless system
- Download OS from network
- Etc.

GRUB BOOT PROCESS

1. The BIOS finds a bootable device (hard disk) and transfers control to the master boot record
2. The MBR contains GRUB stage 1. Given the small size of the MBR, Stage 1 just load the next stage of GRUB
3. GRUB Stage 1.5 is located in the first 30 kilobytes of hard disk immediately following the MBR. Stage 1.5 loads Stage 2.
4. GRUB Stage 2 receives control, and displays to the user the GRUB boot menu (where the user can manually specify the boot parameters).
5. GRUB loads the user-selected (or default) kernel into memory and passes control on to the kernel.

EXAMPLE GRUB CONFIG FILE



The image shows a screenshot of a gedit text editor window. The title bar reads "menu.lst (/boot/grub) - gedit". The menu bar includes "File", "Edit", "View", "Search", "Tools", "Documents", and "Help". The file "menu.lst" is open in the editor. The content of the file is as follows:

```
## should update-grub create memtest86 boot option
## e.g. memtest86=true
##      memtest86=false
# memtest86=true

## should update-grub adjust the value of the default booted system
## can be true or false
# updatedefaultentry=false

title          Windows XP
root           (hd0,0)
savedefault
makeactive
chainloader    +1

title          Ubuntu Linux
root           (hd0,2)
kernel         /boot/vmlinuz-2.6.15-25-386 root=/dev/hda3 ro quiet splash
initrd         /boot/initrd.img-2.6.15-25-386
boot
```

LILO: LINUX LOADER

- Not depend on a specific file system
- Can boot from haddisk and floppy
- Up to 16 different images
- Must change LILO when kernel image file or config file is changed



The diagram illustrates a system architecture on a light gray background. A large, light gray rounded rectangle contains two main components. On the left is a white rectangular box with a thin gray border, which is divided into a top section and a bottom dark gray section. The word 'KERNEL' is centered in the top section. To the right of this box is a vertical olive-green rectangle, also with a thin gray border.

KERNEL

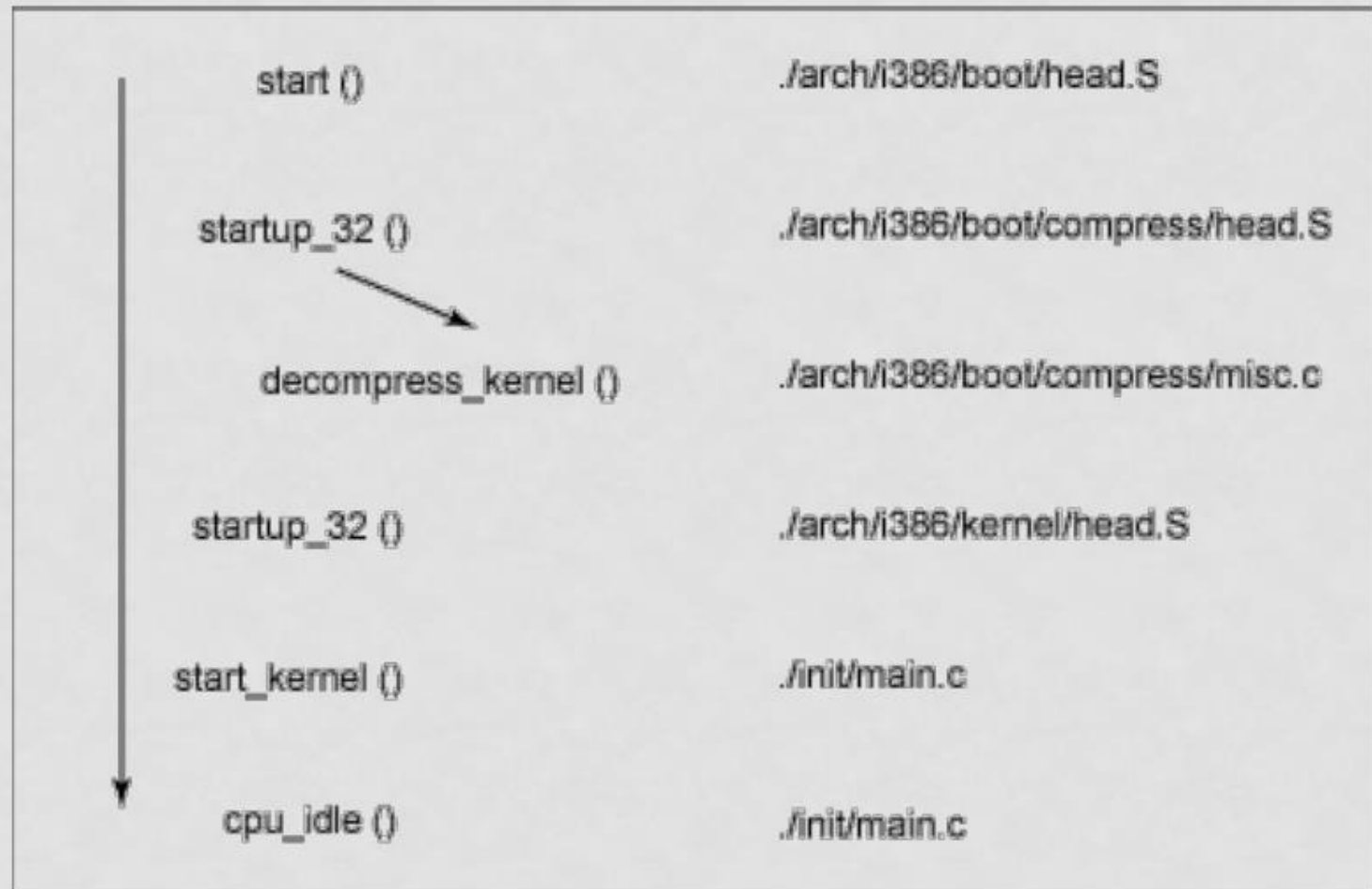
KERNEL IMAGE

- The kernel is the central part in most computer operating systems because of its task, which is the management of the system's resources and the communication between hardware and software components
- Kernel is always store on memory until computer is tern off
- Kernel image is not an executable kernel, but a compress kernel image
- zImage size less than 512 KB
- bzImage size greater than 512 KB

TASK OF KERNEL

- Process management
- Memory management
- Device management
- System call

MAJOR FUNCTIONS FLOW FOR LINUX KERNEL BOOT



INIT PROCESS

- The first thing the kernel does is to execute init program
- Init is the root/parent of all processes executing on Linux
- The first processes that init starts is a script `/etc/rc.d/rc.sysinit`
- Based on the appropriate run-level, scripts are executed to start various processes to run the system and make it functional

THE LINUX INIT PROCESSES

- The init process is identified by process id "1"
- Init is responsible for starting system processes as defined in the `/etc/inittab` file
- Init typically will start multiple instances of "getty" which waits for console logins which spawn one's user shell process
- Upon shutdown, init controls the sequence and processes for shutdown

SYSTEM PROCESSES

Process ID	Description
0	The Scheduler
1	The init process
2	kflushd
3	kupdate
4	kpiod
5	kswapd
6	mdrecoveryd

INITTAB FILE

- The inittab file describes which processes are started at bootup and during normal operation
 - /etc/init.d/boot
 - /etc/init.d/rc
- The computer will be booted to the runlevel as defined by the initdefault directive in the /etc/inittab file
 - id:5:initdefault:

RUNLEVELS

- A runlevel is a software configuration of the system which allows only a selected group of processes to exist
- The processes spawned by init for each of these runlevels are defined in the `/etc/inittab` file
- Init can be in one of eight runlevels: 0-6

RUNLEVELS

Runlevel	Scripts Directory (Red Hat/Fedora Core)	State
0	/etc/rc.d/rc0.d/	shutdown/halt system
1	/etc/rc.d/rc1.d/	Single user mode
2	/etc/rc.d/rc2.d/	Multiuser with no network services exported
3	/etc/rc.d/rc3.d/	Default text/console only start. Full multiuser
4	/etc/rc.d/rc4.d/	Reserved for local use. Also X-windows (Slackware/BSD)
5	/etc/rc.d/rc5.d/	XDM X-windows GUI mode (Redhat/System V)
6	/etc/rc.d/rc6.d/	Reboot
s or S		Single user/Maintenance mode (Slackware)
M		Multiuser mode (Slackware)

RC#.D FILES

- rc#.d files are the scripts for a given run level that run during boot and shutdown
- The scripts are found in the directory /etc/rc.d/rc#.d/ where the symbol # represents the run level

INIT.D

- Deamon is a background process
- init.d is a directory that admin can start/stop individual demons by changing on it
 - `/etc/rc.d/init.d/` (Red Hat/Fedora)
 - `/etc/init.d/` (S.u.s.e.)
 - `/etc/init.d/` (Debian)

START/STOP DEAMON

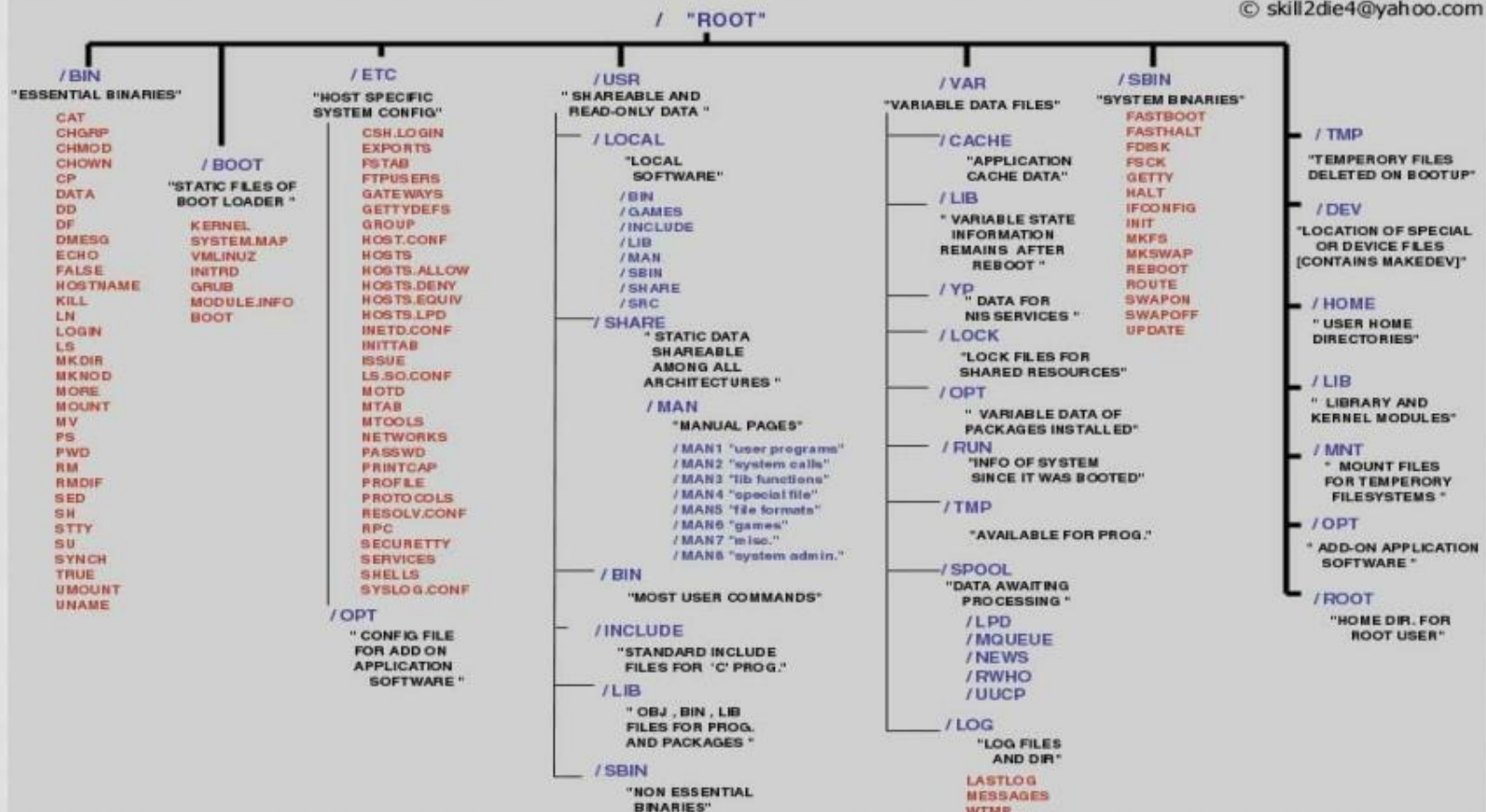
- Admin can issuing the command and either the start, stop, status, restart or reload option
- i.e. to stop the web server:
 - `cd /etc/rc.d/init.d/`
 - (or `/etc/init.d/` for S.u.s.e. and Debian)
 - `httpd stop`

LINUX FILES STRUCTURE



LINUX FILES STRUCTURE

© skill2die4@yahoo.com



FSSTND : (FILESYSTEM STANDARD)

- All directories are grouped under the root entry "/"
- **root** - The home directory for the root user
- **home** - Contains the user's home directories along with directories for services
 - ftp
 - HTTP
 - samba

FSSTND : (FILESYSTEM STANDARD)

- **bin** - Commands needed during booting up that might be needed by normal users
- **sbin** - Like bin but commands are not intended for normal users. Commands run by LINUX.
- **proc** - This filesystem is not on a disk. It is a virtual filesystem that exists in the kernels imagination which is memory
 - 1 - A directory with info about process number 1. Each process has a directory below proc.

FSSTND : (FILESYSTEM STANDARD)

- **usr** - Contains all commands, libraries, man pages, games and static files for normal operation.
 - **bin** - Almost all user commands. some commands are in /bin or /usr/local/bin.
 - **sbin** - System admin commands not needed on the root filesystem. e.g., most server programs.
 - **include** - Header files for the C programming language. Should be below /usr/lib for consistency.
 - **lib** - Unchanging data files for programs and subsystems
 - **local** - The place for locally installed software and other files.
 - **man** - Manual pages
 - **info** - Info documents
 - **doc** - Documentation
 - **tmp**
 - **X11R6** - The X windows system files. There is a directory similar to usr below this directory.
 - **X386** - Like X11R6 but for X11 release 5

FSSTND : (FILESYSTEM STANDARD)

- **boot** - Files used by the bootstrap loader, LILO. Kernel images are often kept here.
- **lib** - Shared libraries needed by the programs on the root filesystem
- **modules** - Loadable kernel modules, especially those needed to boot the system after disasters.
- **dev** - Device files
- **etc** - Configuration files specific to the machine.
- **skel** - When a home directory is created it is initialized with files from this directory
- **sysconfig** - Files that configure the linux system for devices.

FSSTND : (FILESYSTEM STANDARD)

- **var** - Contains files that change for mail, news, printers log files, man pages, temp files
 - **file**
 - **lib** - Files that change while the system is running normally
 - **local** - Variable data for programs installed in /usr/local.
 - **lock** - Lock files. Used by a program to indicate it is using a particular device or file
 - **log** - Log files from programs such as login and syslog which logs all logins and logouts.
 - **run** - Files that contain information about the system that is valid until the system is next booted
 - **spool** - Directories for mail, printer spools, news and other spooled work.
 - **tmp** - Temporary files that are large or need to exist for longer than they should in /tmp.
 - **catman** - A cache for man pages that are formatted on demand

FSSTND : (FILESYSTEM STANDARD)

- **mnt** - Mount points for temporary mounts by the system administrator.
- **tmp** - Temporary files. Programs running after bootup should use /var/tmp