```
<<------------------------------------------------------------------------------------
------------>>
                              Chapter 1

UEFI - Unified Extensible Firmware Interface
BIOS - Basic Input and Output System

- files in the /etc control startup behaviour especially those in the
  /etc/default and /etc/sysconfg

OBJECTIVES:
   -Explain the boot process
        The basic steps are:
           1. The BIOS/UEFI locates and executes the boot program, or boot
loader.
           2. The boot loader loads the kernel.
           3. The kernel starts the 'init' process(pid=1).
           4. init manages system initialization, using conventional 'SysVinit'
startup scripts, or
              using 'Upstart' or sysmd.
                                                         Checkmemory and hardware
                                                        /
           STEPS: BIOS ---> Power On Self Test(POST) -->
                                                        \
                                                         Locates boot program in
MBR(Master Boot Record)
                                                                                |
                                                                                V

GRUB(GRand Unified Bootloader)

  |

        V

      Kernel

        |

      Hardware check

        |

      init processes start

       /|\

      / | \

SysVinit   |  systemd
```

init
scripts |

Upstart

-Identify several types of boot loaders.
 Types include:
  GRUB
  LILO - Linux Loader (obsolete)
  efilinux - designed for the UEFI mechanism
  Das U-Boot - most popular loader for embedded Linux systems; others include
  bareboot.
-Describe what BIOS does.
 Checks memory and hardware, then locates Boot loader in MBR
-Identify the relevant configuration files.
 they are: /etc/sysconfig for RHEL, and /etc/default for Debian systems
-Describe how the system shuts down and reboots.
 commands examples:

```
sudo shutdown -h +1 "Power Failure imminent"
sudo shutdown -h now
sudo shutdown -r now
sudo shutdown now
FORMAT: shutdown [OPTIONS...] [TIME] [WALL...]
find more help using 'shutdwon --help'
```

Basic Steps For Computer Startup:
 1 - Boot loader
 2 - Linux kernel and initrd or initramfs loaded into memory and kernel executes
 3 - init process starts
 4 - Additional kernel modules(including device drivers) are loaded, system services started.

<<--->>

Chapter 3

OBJECTIVES:
- Explain what the role of the GRUB is.
 1 - handles the early phases of system startup
 2 - makes it possible to choose alternatives OS
 3 - makes it possible to choose alternative kernels and/or initial ramdisks at boot time
 4 - Boot paramters can be easily changed at boot time without having to edit configuration
 files, etc. in advance.
- Understand the differences btween the GRUB 1 and GRUB 2 versions.
 GRUB 2 :

```
                  - file read at boot are
                       /boot/grub/grub.cfg or /boot/grub2/grub.cfg
                       This file is auto-generated by update-grub (or
grub2-mkconfig on RHEL 7)
                       based on configuration files in the /etc/grub.d directory
and on /etc/default/grub
              GRUB 2 :
                  - file read at boot are
                       /boot/grub/grub.conf or /boot/grub/menu.lst.
                       In RHEL 5-7, it can be edited using the 'grubby utility'.
Any changes made will be preserved
                       but in GRUB 2 any changes to grub.cfg is lost when next it
is auto-generated.
      - Be familiar with the interactive selections you can make at boot.
              - on entering the GRUB environment after BIOS setup menu appears
              - menu offers the ffl.:
                           ~ list of bootable images
                           ~ interactive shell - for altering the available
stanzas
                           ~ enter pure shell command
                           ~ reinstall GRUB

      - Know how to install GRUB
  GRUB 1 using a program called grub -> $ sudo grub or grub-install

 /                                        > root (hd0, 0)
              - install grub program and associated utilities in proper locations
-                                         > setup (hd0)

 \                                        > exit

  \

   GRUB 2 uses a bunch of utilities like grub2-* or grub-*

      -> $ sudo grub2-install /dev/sda

              - installing files GRUB needs to operate at boot time, either under
/boot/grub or /boot/grub2
                            : Linux kernel files vmlinux-*, initramfs-* which need
to be in the /boot directory
              - installing GRUB as the boot loader in the system

      - Explain how the configuration files that GRUB needs are used and
modified.
                     - the two locations that are used in the reconstruction of
the /boot/grub2/grub.cfg are:
                           /etc/default/grub, and
                           /etc/grub.d

--------------------------------------------------------------------------------
```

```
--------------------------------<<
                   GRUB Device Nomenclature

     - sda1 is (hd0,1) in GRUB 2 but (hd0,0) in GRUB 1
     - sdc4 -s (hd2,4) in GRUB 2 but (hd1,3) in GRUB 1

  Note: In the configuration file, each stanza has to specify what the root
partition is i.e. the partition that contains
        the 'kernel' itself(in the /boot directory), say /boot had its own
directory /dev/sda1, then
        For GRUB 1,
                title 3.17.3
                       root (hd0,0)
                       kernel vmlinuz-3.17.3 ro root=/dev/sda2 quiet
                       intrd intramfs-3.17.1.img

        If /boot is not in its own partition, it might look like
                  title 3.17.3
                        root (hd0,0)
                        kernel /boot/vmlinuz-3.17.3 ro root=/dev/sda1 quiet
                        intrd /boot/intramfs-3.17.3.img

       it is also fine to do kernel (hd0,0)/vmlinuz

<<--------------------------------------------------------------------------------
-------------------------------------------->>
```

Chapter 4 init: SystemV, Upstart, Systemd

```
Steps:
        Device recognition and initialization
        launch system services
        filesystems made available
        start important management systems
        make system available


OBJECTIVE:
        - Understand the importance of the 'init' process.
        - Explain the traditional SysVinit method works and how it incorporates
'runlevels' and
          what happens in each one.
        - know how to use chkconfig and service to start and stop services or
make them persistent across
          reboots.
        - Understand the alternative 'Upstart' and 'systemd'
        - Use 'systemctl' to configure and control 'systemd'.


The mother process controller is the /sbin/init or simple called 'init'
```

System Runlevels

Runlevel	Meaning
S,s	Same as 1
0	Shutdown system and turn off
1	Single User Mode
2	Multiple user, no NFS, only text login
3	Mutiple user, with NFS and network, only text login
4	Not used
5	Multiple user, with NFS and network, graphical login with X
6	Reboot

telinit can be used to change from one level to another

```
$ sudo /sbin/telinit 5
```

On start the 'init' process reads the /etc/inittab
- Here the scripts to be run are mentioned along with other parameters.
- Format: id:runlevel(s):action:process
    - where:
        id - a unique 1-4 character identification for the entry
        runlevel(s) - zero or more single characters or digit indicating runlevel the action will be taken for.
        action - describes the action to be taken.
        process - specifies the process to be executed.

The 'init' run steps:

```
                                                                  1                     2
        (init)/sbin/init -----> /etc/inittab ===> (script)rc.sysinit ===>
(script)rc
                            |                             ^
^
                            |                                                     |
                       |
                            |                                     [start LVM, mount
fs, etc.]      |
                            |
|
            <Reads from for scripts to be run>
|

         |
                                                 <makes system to go to
rc.d/rc[0-6].d and run all scrips there>
```

```
Note: All runlevel directory link back to the /etc/init.d directory where all the
scripts actually reside.
      Start scripts start with S in name
      Kill scripts start with K in name
      --Controlling which initialization scripts are run on entry to each
runlevel involves managing the symbolic links,
        this can be done manually but the 'chkconfig' utility is used to do this
efficiently.
Note: Ubuntu uses update-rc.d inplace of chkconfig

        chkconfig useage:
                - check service to see if it is set to run in the current level
                'chkconfig service_name'

                - see what services are configured to run in each of the run
levels
                'chkconfig --list [service names]

                - Turn on a certain service next time the system boots
                'sudo chconfig somme_service on'

                - Do not turn on a service next time the system boots
                'chkconfig some_service off

                - Change a currently running service
                'sudo chkconfig service_name [stop | start]

The chkconfig utility process explained:
  Syntax in scripts:
     #  chkconfig: 2345 10 90
                --meaning runlevel 2, 3, 4, 5
                --start script -> S10
                --stop script -> K90


UPSTART:
         is 'event driven', rather than being a set of serial procedures. Event
notifications are sent to the 'init' process to tell it to execute
        certain commands at the right time after pre-requisites have been
fulfilled. 'Upstart' is being superseded by 'systemd'.

        Upstart configuration files are:
                - /etc/init/rcS.conf
                - /etc/rc-sysinit.conf
                - /etc/inittab
                - /etc/init/rc.conf
                - /etc/rc[0-6].d

        Upstart events are found in the /etc/event.d or (in Ubuntu)
```

/etc/apm/event.d

        Using 'initctl' you can view, start, stop jobs in much the same way as that 'service' does.
                syntax: initctl options command
                        - options includes: start, stop, restart, reload, status, list, emit

              More info: www.//upstart.ubuntu.com/cookbook

UPSTART STEPS:

```
             (/sbin/init)
                |
Kernel -----> init
                  \
                   \
                    \
                    V
                    rcS.conf(script)
                    |
                    |
                    |
                    V
                    rc-sysinit.conf(script)
                   /
                  /
                 V
   -LVM loaded
   -Mount filesystems
   -Execute all runlevel
    scripts specified in the
    /etc/inittab via
    the rc.conf script
   -then prefdm.conf is run
    at runlevel 5
```

SYSTEMD(systemd)
----
  features include:
      - compatible with SysVinit scripts.
      - Boosts faster.
      - provides paralization capabilities.
      - Uses socket and D-Bus activation for starting services.
      - Replaces scripts with programs.
      - Offers on-demand starting daemons
      - Keeps track of processes using cgroups(control groups).
      - snapshots capabilities and system state restoration.
      - can be a drop in replacement for SysVinit.

```
      - uses '.service' files rather than bash scripts
  Examples of new configuration files for systemd are:
      - /etc/hostname (redhat, replaces /etc/sysconfig/network)
      - /etc/HOSTNAME SUSE
      - /etc/hostname Debian
      - /etc/vconsole.conf - default keyboard mapping and console font
      - /etc/sysctl.d/*.conf - drop-in directory for kernel 'sysctl' parameters
      - /etc/os-release - distros ID


     Runlevel    Target Units                         Description
     0           runlevel0.target, poweroff.target     Shut down and power off
the system.
     1           runlevel1.target, rescue.target       Set up a rescue shell.
     2           runlevel2.target, multi-user.target   Set up a non-graphical
multi-user system.
     3           runlevel3.target, multi-user.target   Set up a non-graphical
multi-user system.
     4           runlevel4.target, multi-user.target   Set up a non-graphical
multi-user system.
     5           runlevel5.target, graphical.target    Set up a graphical
multi-user system.
     6           runlevel6.target, reboot.target       Shut down and reboot the
system.

NOTE:------------------------------------------------------------------
      SysVinit                                           systemd
         |                                                   |
         v                                                   v
   service utility                                   systemctl utility
   update-rc.d, invoke-rc.d -- Ubuntu, Debian
   sysv-rc-conf -- Ubuntu
   chkconfig -- RHEL, CentOS, Fedora
   -----------------------------------------------------------------

 'systemctl' is the main utility for managing services in 'systemd'

  Basic syntax: $ systemctl [options] command [name]


  Useage examples:
        $ systemctl --> shows status of systemd controlled services

        $ systemctl list-units -t service --all --> show all available services

        $ systemctl list-units -t service  --> show only active services

        $ sudo systemctl start foo --> start or activate foo  service
          sudo systemctl start foo.service
          sudo systemctl start /path/to/foo.service
```

$ sudo systemctl stop foo.service --> stop(deactivate) a service

$ sudo systemctl enable sshd.service --> to enable or disable a service
sudo systemctl disable sshd.service
-- equivalent to chkconfig --add/ --del and doesn't actually start the service.

<-->

Chp 5 Linux Filesystem Tree Layout

Types of file system differ by:
- purpose
- size
- ownership
- sharing

Objectives of chpter 5:
- Explain why Linux requires the organization of one big filesystem tree, and what the major considerations are for how it is done.
- Know the role played by the Filesystem Hierarcchy Standard.
- Describe what must be available at boot in the root(/) directory, and waht can available only once the system has started.
- Explain each of the main subdirectory trees in terms of purpose and contents.

File systems are:
1. Shareable vs. non-shareable
2. Variable vs static

File main directories present in FHS

Directory	In FHS?	Purpose
/	Yes	Primary directory of the entire file system hierarchy.
/bin	Yes	Essential executable programs that must be available in 'single user mode'.
/boot	Yes	Files neede to boot - kernel, initrd ot initramfs, images, boot configuration files and bootloader programs.
/dev	Yes	Device nodes, used to interact with hardware devices.
/etc	Yes	System wide configuration files.
/home	Yes	User home directories including personal settings, files, etc.
/lib	Yes	Libraries required bt executable binaries in .bin and /sbin.

```
/lib64          No              64-bit libraries requires by executable binaries
in /binand /sbin, for systems which can run both 32-bit and 64-bit programs.
/media          Yes             Mount points for removable media such as CDs,
DVDs, USB sticks etc.
/mnt            Yes             Temporarily mounted filesystems.
/opt            Yes             Optional application software packages.
/proc           Yes             Virtual pseudo-filesystem giving information
about the system and processes running on it
                                Can be used to alter system parameters.
/sys            No              Virtual psuedo-filesystem giving information
about the system and processes running on it. Can be used to
                                alter system parameters. Similar to a device tree
and is part of the Unified Device Model.
/root           Yes             Home directory of the root user.
/sbin           Yes             Essential system binaries.
/srv            Yes             Site-specific data served up by the system.
Seldom used.
/tmp            Yes             Temporary files; on many distributions lost
across reboot and may be a ramdisk in memory.
/usr            Yes             Multi-user applications, utilities and data;
theoretically read-only.
/var            Yes             Variable data that changes during system
operation.


 The three file associated with each 'bootable kernel' are:
        - vmlinuz -- compressed Linux kernel
        - initramfs or initrd -- Initial RAM Filesystem, mounted before the real
root filesystem becomes available.
        - config --configuration file used when compiling the kernel. Used mainly
for bookkeeping and reference
        - System.map -- The kernel 'symbol table', useful for debugging. Gives
the 'hexadecimal addresses' of all kernel symbols.

---------------------
/bin
---------------------
        contains executable binaries needed buy both admin and unprivileged
users.
        may not contain cubdirectories

---------------------
/boot
---------------------
        two main absolutely essential files are: vmlinuz --> comporessed kernels
                                                 initramfs --> initial RAM
Filesystem

---------------------
/dev
---------------------
```

        contains special device files(also known as device nodes) this represent devices built into or connected to the system.
        network devices do not have device nodes in Linux and are referenced by names such as eth1 or wlan0

/etc
----
        contains machine-local configuration files; there should be no executable binary programs.
        sample files and directories include:
                - /etc/sysconfig -- system configuration and directories (Red Hat)
                - /etc/default -- same as above (Debian)
                - /etc/skel -- contains skeleton files used to populate newly created home directories
                - /etc/init.d -- contains start up amd shut down scripts when using System V initialization

----
/home
----
        contains all personal configuration, data, and executable programs.

/lib
----
        contains only thise libraries needed to execute the binaries in /bin and /sbin. These are useful for booting the system
        and executing commands within the filesystem.
        kernel modules(device and filesystem drivers) are located under /lib/modules/<kernel-version-number>
        PAM(Pluggable Authentication Modules) files are stored in the /lib/security
        Systems that support both 32-bit and 64-bit libraries use /lib and /lib64 respectively.

/media
----
        used to mount filesystems on removeable media such as CDs, DVDs, and USB drives or even old floppy disks
        on SUSE and RHEL 7 removable media will pop up under /run/media/[username]/....

/mnt
----
        used to temporarily mount a filesystem when needed. Like
                - NFS

```
                - Samba
                - CIFS
                - AFS

-----------------
/opt
-----------------
        used by software packages that wish to keep all their files in one
isolated place rather than scatter themall over
        the system.
        Example: dolphy_app -- /opt/dolphy_app/bin, /opt/dolphy_app/man
        Special subdirectories of /opt are:
                /opt/bin

----------------
/proc
----------------
        mount point for a pseudo-filesystem, where information only resides in
memory, not on disk. Like /dev the /proc is
         empty on a non-running system.
        Here each active process on the system has its own subdirectory that
gives detailed information about the state of the process,
         the resources it is using, and its history.
        Important pseudo-files include:
                - /proc/interrupts -------
                - /porc/meminfo           |
                - /proc/mounts            |-------> system's hardware
                - /proc/partitions  ------

                - /proc/filesystem ------|
                - /proc/sys/-------------|-------> system configuration
information and interfaces

----------------
/sys
----------------
        mount point for sysfs pseudo-filesystem, where infromation resides only
in memory
        sysfs is used both to gather information about the system, and modify its
behaviour while running.


---------------
/root
---------------
        home directory of the root user

--------------
/sbin
--------------
        contains binaries essential for booting, restoring, recovering, and/or
```

```
repairing
        must be able to mount other filesystems on /usr, /home and other
locations
        these programs should be included here:
                - fdisk, fsck, getty, halt, ifconfig, init, mkfs, mkswap, reboot,
route, swapon, swapoff, update.
        contains binaries essential for booting, restoring, recovering, and/or
repairing in addition to those binaries in /bin.
        they must also be able to mount other filesystems on '/usr, /home' and
other locations, once the root system is known to be in good health during boot.
        The following programs shld be included in this directory, if their
subsystems are installed:
                fdisk, fsck, getty, halt, ifconfig, init, mkfs, mkswap, reboot,
route, swapon, swapoff, update.
        Note: some recent repos are merging /sbin and /usr/sbin as well as /bin
and /usr/bin.

-------------
/tmp
-------------
        store temporary files, accessed by any user
        reset /tmp behaviour on RHEL 6 using 'systemctl mask tmp.mount
        for temporary files
        regularly cleaned of its contents at regurlar basis using 'cron jobs' or
'at reboot'
        files here are stored in memory not disk
        Note: canceling the usage of /tmp for creating large files can be done
using the command: systemctl mask tmp.mount, then 'reboot'

-------------
/usr
-------------
        secondary hierarchy
        used for files that are not needed for system booting.
        may be located at location different from root directory
        software packages should not create subdirectories directly under /usr
        typically read-only data
        contains binaries which are not need in single user mode


        Directory       |          Purpose

---------------------------------------------------------------------------------
        /usr/bin        | binaries for applications not needed in single user
mode
        /usr/include    | header files for compiling applications
        /usr/lib        | Libraries for programs in /bin and /sbin.
        /usr/lib64      | 64-bit libraries for 64-bit programs in /bin and /sbin.
        /usr/sbin       | Non-essential system binaries, e.g. system daemons.
        /usr/share      | Shared dara used by applications,
architecture-independent
```

```
        /usr/src        | Source files usually for linux kernel.
        /usr/X11R6      | X Window files; generally obsolete.
        /usr/local      | Local data and programs specific to the host.
Subdirectories include bin, sbin, lib, share, include, etc.


-----------------
/var
----------------
        contains variable (or volatile) data files that change frequently during
system operation.
        Examples:
                - Log file
                - Spool directories and files for printing, mail queues, etc.
                - Admin data files
                - Transient and temporary files


                Directory       |         Purpose

--------------------------------------------------------------------------------
                /var/ftp        | ftp server base
                /var/lib        | Persistent data modified by programs as they
run.
                /var/lock       | Lock files used to control simultaneous access
to resources.
                /var/log        | Log files
                /var/mail       | User mailboxes
                /var/run        | Information about the running system since the
last boot.
                /var/spool      | Tasks spooled or waiting to be processed, such
as print queues.
                /var/tmp        | Temporary files to be persisted across reboot,
at times linked to /tmp
                /var/www        | Root for website hierarchies.


-----------------
/run
-----------------
        stores transient files: those that contain run-time information, which
may need to be written early in system startup.



<---------------------------------------------------------------------------------
---------------------->
```

Chapter 6

This lies at the heart of the Linux operating system.
It controls access to hardware, competition for resources between different

applications and other tasks, handles I/O activity and
        files and data storage, security, networking, etc.
Adding kernel command line parameters at boot time, the system can be made to behave in many different ways.

        Learning Objectives:
                - grasp the main responsibilities the kernel must fulfill and how it achieves them.
                - Explain what parameters can be set on the kernel command line and how to make them effective either for just one
                  system boot, or persistently.
                - know where to find detailed documentation on these parameters.
                - know how to use sysctl to set kernel parameters either after the system starts, or persistently across system reboots.

        kernel serves as a connection between hardware and software
        handles all connected devices using 'device drivers'

        Main kernel:
                - system initialization and boot up.
                - process scheduling
                - memory management
                - controlling access to hardware
                - I/O between applications and storage devices.
                - Implementation of local and network filesystems.
                - Security control, both locally (such as filesystem permissions) and over the network.
                - networking control.

Kernel Command Line:
        GRUB version 1
                /boot/grub/grub.conf
        GRUB version 2
                /boot/grub2/grub.cfg

        cat /proc/cmdline -- shows what cmdline a system was booted with.

-Kernel Boot Parameters:
        sources of documentation for kernel parameters includes:-
                - in the kernel source Documentation/kernel-parameters.txt
                - Online, at
http://kernel.org/doc/Documentation/kernel-parameters.txt
                - On the as kernel-doc or linux-doc
                - By typing 'man dootparam'

        parameters may be typed as an argument or in the form param=value, where value can be a 'string, integer, array of integers
        etc.

```
        Bootparameters:
                - ro -- mounts root device read-only on boot.
                - root -- root filesystem
                - rd_LVM_LV -- it activates the root filesystem in the logical
volume specified
                - rd_NO_LUKS -- disables crypto LUKS detection.
                - rd_NO_DM -- disables DM RAID detection.
                - LANG -- is the system language.
                - SYSFONT -- is the console font.
                - KEYTABLE -- is the keytable filename.
                - rhgb -- for graphical boot support on Red Hat systems.
                - quiet -- disables most log messages.

        'sysctl' interface can be used to read and tune kernel parameters at run
time. To display current values - 'sysctl -a'
                - More details: sysctl use -- man 8 sysctl
                                sysctl() use -- man 2 sysctl
                - if settings are in /etc/sysctl.conf -- man sysctl.conf
        Note:
                kernel command line allows specification of start up options
                sysctl allows specification of run time options


<--------------------------------------------------------------------------------
---------->

                Chapter 7 Kernel Modules

Objectives:
        - list the advantages of utilizing kernel modules.
        - use insmod, rmmod, and modprobe to load and unload kernel modules.
        - know how to use modinfo to find out information about kernel modules.


        Advatages of kernel modules include:
                - it facilitates development
                - kernel reboots are nor required
                -



        Module utilities:
                - lsmod -- list modules
                - insmod -- directly load modules
                - rmmod -- directly remove modules
                - modprobe -- load or unload modules, using a pre-built module
database with dependency information
                - depmod -- rebuild the module dependency database; needed by
modprobe and modinfo
                - modinfo -- display information about a module.
        Syntax for load and unloading modules
                sudo /sbin/rmmod module_name
```

```
                sudo /sbin/insmod <pathto>/module_name

                ------------OR using modprobe---------

                sudo /sbin/modprobe module_name
                sudo /sbin/modprobe -r module_name
```

information on modules can be gotten using:
modinfo module_name OR /sbin/modinfo my_module, /sbin/modinfo <pathto>/my_module.ko

A modules status can be seen in the /sys pseudo-filesystem directory tree
- e.g.:
for module e1000
/sys/module/e1000
some or if not all parameters can be 'read or written' under /sys/module/e1000/parameters

Module parameters:loading module with parameters

```
                sudo /sbin/insmod <pathto>/e1000.ko debug=2 copybreak=256
                sudo /sbin/modprobe e1000 debug=2 copybreak=256
```

Note: files in the /etc/modprobe.d control some parameters that come into play when using modprobe

<--->

Chapter 8. Devices and UDEV

Linux uses udev to discover devices (hardware and peripheral) both during boot and later on when connected to the system.
Device nodes are created automatically and then used by apps and OS subsystems to communicate with and transfer data to and from devices.

Objectives:
- Explain the role of 'device nodes and how they use major and minor numbers.
- Understand the need for the udev method and list its key components.
- Descibe how the udev device manager functions.
- Identify udev rule files and learn how to create custom rules.

Character and block devices have filesystem entries associated with them; network devices in Linux do not.
These device nodes can be used by programs to communicate with devices, using I/O system calls such as 'open(), close(), read(), and write().
Network devices work by transmitting and recieving packets of data.

Device nodes can be created with:

```
                sudo mknod [-m mode] /dev/name <type> <major> <minor>
```

```
                e.g., mknod -m 666 /dev/mycdrv c 254 1

                                        Device Nodes
                                        ------------


---------------------------------------------------------------------------------
        |                             Application
      |

---------------------------------------------------------------------------------
                                                  / \                   / \
                                                   |                     |
                                                   |                     |
                                                   |                     |
                                                   V                     |
        -------------------------------------------------------------    |
        |                        Virtual File System                |    |
        -------------------------------------------------------------    |
                     / \                / \           / \                |
                      |                  |             |                 |
                      |                  |             |                 |
                      V                  V             |                 |
        -------------------       --------------       |  -----------    |
        |character node   |       |block node  |<--------|  |socket   |    |
        -------------------       -------------           ----------     |
              / \                    / \                        / \         |
               |                      |                          |          |
               |                      |                          |          |
               V                      V                          V          V
        -------------------       --------------          -----------------
        |character driver |       |file system |          | TCP/IP        |
        |                 |       |            |          |Appletalk, etc |
        -------------------       --------------          -----------------
               / \                   / \                        / \
                |                     |                          |
                |                     |                          |
                |                     V                          V
                |                 --------------          -----------------
                |                 |block driver|          |network driver |
                |                 --------------          -----------------
                |                     / \                        / \
                |                      |                          |
                |                      |                          |
                V                      V                          V

---------------------------------------------------------------------------------
        |                              Hardware
      |

---------------------------------------------------------------------------------
```

The 'major' and 'minor' numbers identify the driver associated with the device.
I most cases (but not all) device nodes of the same type (block or character) with the same major number use the same driver.
Minor numbers are used only by the device driver to differentiate between the different devices it may control.
mknod() and stat() : return information about 'major' and 'minor' numbers.

udev:

POSIX -- Portable Operating System Interface
udev -- user device
Handles the dynamical generation of device nodes. It replaced devfs and hotplug
udev Components:
    udev runs a daemon(udevd or systemd-udevd) and monitors a netlink socket.
    mechanism: device --ADDED--> uevent kernel facility --SENDS MESSAGE--> socket --> udev [addes or removes nodes]
    - libudev -: library which allows access to information about the devices
    - udevd -: daemon that manages the /dev directory.
    - udevadm -: utility for control and diagnostics.
udev main configuration file: /etc/udev/udev.conf
udev naming rules file: /etc/udev/rules.d
udev Device Manager:-
    mechanism --: udev[receives message from kernel] --PARSES--> Rule-Setting Files[in /etc/udev/rules.d/*.rules]
        Actions taken includes:-
            - device node naming
            - device node and symbolic links creation.
            - setting file permissions and ownership for the device node.
            - taking other actions to initialize and make device available.
            These rules are completely customizable
udev rules are located under:
    /etc/udev/rules.d/<rulename>.rules, e.g. 30-usb.rules, 90-mycustom.rules

```
                                                                 Create device node in /dev
                                                                             / \
                                                                              |
                                                                              |
                                                                              |
```

```
                                                  udevd
                                                   / \
                                                  /   \
                                                 /     \
                                                /       \
                                               /         \
                                              /           \
                                      kernel module    Rules
database
```

Creating udev Rules

format: <match><op>value [, ...] <assignment><op>value [, ...]

<-->

Chapter 9. Partitioning and Formatting Disks

Schemes are selected based on the following:

- size of system
- number of users and their needs
- type of hardware available
- type of data bus to which the storage is attached

Objectives:

- Describe and contrast the most common types of hard disks and data buses.
- Explain disk geometry and other partitioning concepts.
- Understand how disk devices are named and how to identify their associated device nodes.
- Distinguish among and select different partitioning stratigies.
- Use utilities such as blkid and fdisk.
- Back up and restore partition tables

Common Disk Types:

- IDE and EIDE(Integrated Drive Electronics, and Enhanced IDE)
  - obsolete
- SATA(Serial Advanced Technology Attachment)
  - seen as SCSI devices by the OS
  - smaller cable size(7 pins) when compared to PATA, a.k.a. IDE
  - native hot swapping, and faster
  - more efficiant data transfer
  - can handle 16GB/s, but 3 GB/s and 6 GB/s are more common
- SCSI(Small Computer Systems Interface)
  - lower capacity than SATA
  - faster than SATA
  - work better in parallel, as when used in RAID configurations
  - versions may include:- Fast, Wide, Ultra, and UltraWide

```
                        more varied device drivers unlike SATA
                        disk range vary from 8 bit bus to 16 bit bus
                        transfer rate of 5 MB/s(narrow, standard SCSI) to about
160 MB/s (Ultra-Wide SCSI-3)
                        Single ended device controllers [host up to 7 devices,
max cable length 6 meters]
                        Differential controllers[host up to 15 devices, max
length 12 meters]
                - SAS(Serial Attached SCSI)
                        newer point to point serial protocol replacing the
earlier Parallel SCSI interface
                        data transfer rate similar to SATA
                        better performance
                - USB(Universal Serial Bus)
                        include pen drives and extensible USB drives
                        OS sees them as SCSI devices
                        in the same category as SSDs drives
        Disk Geometry:
                - parameters include:-
                        heads, cylinders, tracks and sectors
                - Structure:-


                                                                    DISKS
                                                                    / \
                                                                     |
                                                                     |
                                                                     |
                                                                     |

--------------------------------------------------------------------------------
----------
                        |    Platter
           Platter              |

--------------------------------------------------------------------------------
----------
                               / \
             / \
                                |
        |
                                |
        |
                                |
        |
                                |
        |

--------------------------------------------------------------------------------
-----------
```

```
                       |                   Cylinders(group of similar platters on
all sectors)                    |

-------------------------------------------------------------------------------------
-----------
                              /   \
            /    \
                             /     \
     /      \
                            /       \
          /        \
                      circular  circular
    circular   circular
                       tracks    tracks
     tracks     tracks
                               |
        |
                               |
        |
                               |
        |

-------------------------------------------------------------------------------------
----------
                       |                           Data Blocks (Sectors)
                               |

-------------------------------------------------------------------------------------
----------
                              / \
       / \
                               |
              |
                               |
              |
                               |
              |

-------------------------------------------------------------------------------------
----------
                       |                                 Read by Heads
                               |

-------------------------------------------------------------------------------------
----------
```

        Partitioning:
                disks are divided into partitions based on physically contigious groups of sectors or cylinders.
                SCSI and SATA support up to 15 partitions, where 1-4 are primary

```
and 5-15 logical partitions
                        structure:
                                                Disk
                                                 |
                                                 |
                                                 |
                                                \ /
                ------------------------------------------------------------------
                |                           Drives                               |
                ------------------------------------------------------------------
                       |                        |                        |
                       |                        |                        |
                       |                        |                        |
                ------------------------------------------------------------------
                |                        Partitions                              |
                ------------------------------------------------------------------
                        |                       |                        |
                        |                       |                        |
                       sda                     sdb                      sdc
                       /|\
                      / | \
                     /  |  \
                    /   |   \
                   /    |    \
                  sda1 sda2  sda3 ..


        Why Partition?
                - separation -- separating installation files from user files
                - sharing -- keeping shared resources like /home on central
location
                - security -- imposing quotas, permissions and settings
                - size -- preventing accumulated data from crashing OS
                - performance --
                - swap -- hibernation schemes can use this
        Partition Table:
                - The disk partition table is contained within the Master Boot
Record(MBR), and the MBR is 512bytes in length.
                  structure is defined by an operating system-independent
convention.
                - The parttion table is 64 bytes long and is placed after the 446
byte boot record
                        -: structure -
                        MBR
          ---------  -----
          |          |
          |          |
          |          | 446 bytes --- GRUB program code.
          |          |
          |          |
          |          |
```

```
        |          -----
    512 |          -----
  bytes |          | Partition 1 - 16 bytes
        |          |----
        |          | Partition 2 - 16 bytes            <----Partiton Table
        |          |----
        |          | Partition 3 - 16 bytes
        |          |----
        |          | Partition 4 - 16 bytes { 0x55AA }
        |          |
        ---------  -----
```

Note: each entry in the partition table 16 bytes long
- gives the following information:
  - Active bit.
  - Beginning address in cylinder/head/sectors (CHS) fromat (ignored by Linux)
  - Partition type code, indicating: xfs, LVM, ntf, ext4, swap, etc.
  - Ending address in CHS(also ignored by Linux).
  - Start sector, counting linearly from 0.
  - Number of sectors in partion.

Linux only uses the last two fields for addressing using the linear block addressing(LBA) method.

Naming Disk Devices and Nodes:

The Linux kernel interacts at a low level with disks through 'device nodes' normally found in the /dev directory

Device nodes are accessed only through the infrastructure of the kernel's Virtual File System

```
        SCSI and SATA:
                first hard disk is /dev/sda
                second hard disk is /dev/sdb
                        partitions:
                                /dev/sdb1
                                /dev/sdc4
        IDE:
                /dev/hda3
                /dev/hdb
```

More on SCSI Device Names:

- these are determined by the controller number/ID number combination.

```
                                       /dev/sda
                                         /   /dev/sdb
                                        /    /
        controller 0 => target ID number 1 & 3
```

```
        controller 1 => target ID number 2 & 5
                                          \   \
                                           \   \
                                            \  /dev/sdd
                                         /dev/sdc
```

```
blkid and lsblk:
----------------
        blkid is a utility to locate block devices and report on their attributes
        blkid works with libblkid library, takes as an argument a particular
device or list of devices
        Usage:
                sudo blkid /dev/sda*
        blkid will only work on devices that contain data that is
finger-printable; e.g., empty partition will not generate a block-identity UUIO
        blkid - forms of operations:
                1 - searching for a device with a specific NAME=value pair, or
                2 - displaying NAME=value pairs for one or more devices
                lsblk - will represent information in a tree format.
        Linux systems shld use a minimum of two partitions:
                - /root:
                        used for the entire logical system

                - Swap:
                        used as an extension of physical memory
                        used as virtual memory

Backing Up and Restoring Partition Tables:
------------------------------------------
        this helps to restore the former partition of disk if new partition fails
                backup - sudo dd if=/dev/sda of=mbrbackup bs=512 count=1
                restore - sudo dd if=mbrbackup of=/dev/sda bs=512 count=1


Partition Table Editors:
------------------------
        -  fdisk: menu driven partition table editor
        -  sfdisk: non-interactive partition editor program, useful in scripting
        -  parted: GNU partition manipulation program.
        -  gparted: widely used graphical interface to parted

Using fdisk:
------------
        start: sudo fdisk /dev/sdb
                the main (one-letter) commands are:
                        -m: Display the menu
                        -p: List the partition table.
                        -n: Create a new partition.
                        -d: Delete a partion.
                        -t: Change a partition type.
```

-w: Write the new partition table information and exit.
-q: Quit without making changes.

cat /proc/partitions - will show you the partition operating system is currently aware of.

<-->

Chapter 10. Encryption Disks

Linux distributions most often use the 'LUKS' method amd perform encryption-related tasks using 'cryptsetup'.

OBJECTIVES:

- provide sound reasons for using encryption and know when it is called for.
- understand 'LUKS' operations through the use of 'cryptsetup'.
- be able to setp and use encrypted filesystems and partitions.
- know how to configure the system to mount encrypted partitions at boot.

Why use encryption:

Configuration and using block device level encryption provides aone of the strongest protections agianst harm
caused by loss or compromise of data contained in hard drives and other media.

Note: Encryption can not be carried out on an already existing partition in place without a data copying operation.

LUKS:

block level encryption is provided through the use of LUKS(Linux Unified Key Setup).
LUKS is highly recommended for portable systems e.g. laptops, tablets, smartphones.

```
LUKS         plain dm-crypt volumes      loop-AES     TrueCrypt
 |                          |              |              |
 |                          |              |              |
 |--------------------------|--------------|--------------|
                                |
                                |
                                v
                            cryptsetup
```

LUKS stores all necessary information in the partition header itself, it is rather easy to migrate partitions to other disks or systems.
LUKS can also be used to transparently encrypt swap partitions.

Cryptsetup:

command format -

cryptsetup [option...] <action> <action-specific>
listing possibilities ->

```
                cryptset --help
    Using an Encrypted Partition:
        if the LVM partition '/dev/VG/MYSECRETE already exists the following
steps will setup encryption
        --> 1. Make it vailable to LUKS
            2. format it
            3. mount it
            4. use it
            5. unmount it
        -->
                sudo cryptsetup luksFormat /dev/VG/MYSECRET -- (if kernel doesn't
support the default method used by 'cryptsetup' use sudo cat /proc/crypto to find
out which your system supports)
                 sudo cryptsetup luksFormat --cipher aes /dev/VG/MYSECRET
                 sudo cryptsetup --verbose luksOpen /dev/VG/MYSECRET SECRET
                 sudo mkfs.ext4 /dev/mapper/SECRET
            mount it -->
                 sudo mount /dev/mapper/SECRET /mnt
            unmount it -->
                 sudo umount /mnt
            remove the mapper -->
                 sudo cryptsetup --verbose luksClose SECRET


     Mounting at Boot:
            Steps include -->
                  - make appropraite entry in /etc/fstab
                  - add entry to /etc/crypttab
                    --such as 'SECRET /dev/mapper/MYSECRET

     Steps for Using LUKS:
           - create a partition for the encrypted block device
           - format with cryptsetup
           - create an un-encrypted pass through device
           - format with a standard filesystem such as ext4
           - mount the filesystem on the encrypted block device


<--------------------------------------------------------------------------------
----------------->

                  Chp11. Linux Filesystems and the VFS( Virtual File System )

Structure of Linux file system:

          software <-----> VFS <-----> on-disk filesystem

Objectives:
        - explain the basic filesystem organization.
        - understand the role of the VFS.
        - know what filesystema are available in Linux and which ones can be used
```

on your actual system.

- grasp why journaling filesystem represent significant advances.
- discuss the use of special filesystems in Lunux.

Local filesystems generally reside within a disk partition which can be a physical partition on a disk, or a logical partition controlled by a Logical Volume Manager (LVM).
Filesystems can also be of a network nature and their true physical embodiment completely hidden to the local system across the network.

VFS:

```
        user software <---------> VFS <---------> actual filesystem
                                   |
                                   |
                                   |
                                   v
                    /dev, /sys, /temp, /proc, /run
```

Journalling Filesystems:

These recover from system crashes or ungraceful shutdowns with little or no corruption, and they do so very rapidly.

Here operations are grouped into transactions, each must be completeed without error, atomically; otherwise the filesystem is not changed.

Examples include:

- ext3, extension of ext2
- ext4, enhanced ext3
- reiserfs, formally for linux
- JFS, IBM
- XFS, RHEL
- btrfs, latest journalling filesystem under rapid development.

Current Filesystem Types:

to see system currently registered and understood filesystem.

cat /proc/filesystems

Special Filesystems:

Filestsystem	Mount Point	Purpose
rootfs	None	During kernel load, provides an empty root directory.
hugtlbfs	Anywhere	Provides extended page access (2 or 4 MB on x86)
bdev	None	Used for block devices.
proc	/proc	Pseudo filesystem access to many kernel structures and aubsystems.
sockfs	None	Used by BSD Sockets.
tmpfs	Anywhere	RAM disk with swapping, re-sizing.
shm	None	Used by System C IPC

```
Shared Memory.
        pipefs                None                    Used for pipes.
        binfmt_misc           Anywhere                Used by various
executable formats.
        devpts                /dev/pts                Used by Unix98
pseudo-termionals.
        usbfs                 /proc/bus/usb           Used ny USB sub-system
for dynamical devices.
        sysfs                 /sys (or elsewhere)     Used as a device tree.
        debugfs               /sys/kernel/debug (or   Used for simple debugging
filr access.
                              elsewhere)


<--------------------------------------------------------------------------------
-------------------------------->

                        Chpt.12. Filesystem Features: Attributes, Creating,
Checking, Mounting
        Objectives:
                - be familiar with concepts such as inodes, directory files and
extended attributes.
                - create and format filesystems.
                - check and fix errors on filesystems.
                - mount and unmount filesystems.

        Inodes:
                is a data structure on disk that describes and stores file
attributes, including location.
                the ionformation stored includes --
                    > permissions
                    > user and group ownership
                    > size
                    > timestamps (nanoseconds)
                      - last access time
                      - last modification time
                      - change time

        Directory Files:
                  is a particualr type of file that is used to associate file
names and inodes.
                  two ways to associate (or link) a file name with an inode:
                      > HARD links point to an inode.
                      > SOFT (or symbolic) links point to a file name which has
an associated inode.

                  process references pathname ---> kernel [searches directories
to find corresponding inode number]
                                                                                |
                                                                                |
                                                                                |
```

```
                                                                                    v
                                                              converts name to inode
number ---> loads into memory
        Extended Attributes and lsattr/chattr:
                 extended attributes associate not interpreted directly by the
filesystem with files.
                 Namespaces used in fileattributes:
                            > user,
                            > trusted,
                            > system - access control list (ACL),
                            > security- SELinux.

                format:     chattr [-|+|=mode] filename (change attribute)
                            lsattr filename   (list file attribute)
                Namespaces:
                        user --
                             flags --> i: immutable, a: append-only, d: no-dump,
A: No atime update

       Creating anf Formatting Filesystems:
                utility for formatting (making) filesystem on a partition is
'mkfs'
                format --> mkfs [-t fstype] [options] [device name], e.g.: sudo
mkfs -t ext4 /dev/sda10 OR sudo mkfs.ext4 /sda10

       Checking and Fixing Filesystems:
                utility for checking and fixing any errors in a filesystem is
'fsck'
                format --> sudo fsck -t ext4 /dev/sda10 OR sudo fsck.ext4
/dev/sda10
                       --> fsck [-t fstype] [options] [device-file]
                Note: SHOULD ONLY BE RUN ON UNMOUNTED FILESYSTEMS.
                to do so run the following command --> sudo touch /forcefsck,
sudo reboot


      Mount:
                Each file system is mounted under a specific directory as in:
                --> sudo mount -t ext4 /dev/sdb4 /home

                mounting files using label or a UUID
                --> sudo mount /dev/sdb4 /home
                --> sudo mount LABEL=home /home
                --> sudo mount  -L home /home
                --> sudo mount UUID=26d58ee2-94jjjfhv0-vnskjs-44ns48 /home
                --> sudo mount -U UUID=26d58ee2-94jjjfhv0-vnskjs-44ns48 /home
                with mount --help you get a quick summary of mount options

     unmount:
                umount [device-file | mount-point]
                --> sudo umount /home
```

```
                --> sudo umount /dev/sda3

     Mounting Filesystem at Boot:
                this command will show you how to mount all filesystems listed in
the /etc/fstab at boot
                --> cat /etc/fstab

     Listing Currently Mounted Filesystems:
                --> mount

     Notes:
        format filesystems --> mkfs
        checking and fixing errors --> fsck
        list file attributes --> lsattr
        change file attribute --> chattr
        list open files --> lsof



<--------------------------------------------------------------------------------
------------------------------------>

                                Chapter 13. Filesystem Features: Swap, Quotas,
Usage.

  Linux uses robust 'swap space' implementation through which the virtual memory
system permits the apparent use of memory than is physically available.

  Filesystem quotas can be used to administer user account usage of disk space.
  Utilities such as 'df' and 'du' enable easy monitoring of filesystem usage and
capacities.


  Objectives:
    - explain the concepts of swap and quotas.
    - use the utilities that help manage quotas: quotacheck, quotaon, quotaoff,
edquota, and quota.
    - use the utilities df and du.

  Swap:
    virtual memory in Linux uses the COW(Copy On Write) technique ie:
                PARENT memory
                      ^
                      |
                      |
                      |
            |---------|----------|-----------------|
          child A   child B    child C          child D --> [extra memory sector
since a portion of PARENT has changed]

          - when memory pressure is high 'less active memory regions are swapped
```

```
out to disk' and only recalled when nedded.

    - In most cases the recommend swap size is the RAM
    - to see what ure system is currently using for swap areas: $ cat /proc/swaps
                                          and current usage: $ free -o

    - commands invovling swap are:
        - mkswap: format a swap partition or file
        - swapon: activate a swap partition or file
        - swapoff: deactivate a swap partition or file
  Quotas:
    - quotacheck:- generates and updates quota accounting files
    - quotaon:- enable quota accounting
    - quotaoff:- disables quota accounting
    - edquota:- used for editing user of group quotas.
    - quotas:- reports on usage and limits.

    Note:
      Quota operations require the existence of the files 'aquota.user and
aquota.group' in the root directory of the filesystem using quotas.

    Setting up Quotas:
      - steps include:-
            -- mount the filesystem with user and/or group quota options:
                --- add the userquota and/or grpquota options to the filesystem
entry in /etc/fstab
                --- remount the filesystem (or mount it if new)
            -- run 'quotacheck' on the filesystem to set up quotas.
            -- enable quotas on the filesystem
            -- set quotas with the edquota program.

    Setting up Quotas:
      - in /etc/fstab:- /dev/sda5 /home ext4 defaults,usrquota 1 1
      - then test the system:
        -- sudo mount -o remount /home
        -- sudo quotacheck -vu /home
        -- sudo quotaon -vu /home
        -- sudo edquota someuser

      - fstab options include: usrquota and grpquota

    quotacheck:
      creates/updates the quota accounting files aquota.user and aquota.group for
the system

    - to update user files for all filesystems in /etc/fstab with user quota
options:
      $ sudo quotacheck -ua

    - to update group files for all filesystems i /etc/fstab with group quota
options:
```

```
      $ sudo quotacheck -ga

    - to update the user file for a particular filesystem:
      $ sudo quotacheck -u [somefilesystem]

    - to update the group file for a particular filesystem:
      $ sudo quotacheck -g [somefilesystem]

    Note: use -v option to get more verbose output
    Note: quotacheck is generally run:-
          -- when quotas are turned on
          -- fsck reports errors during system start up

    Turning quotas on and off:
      - syntax:
        -- $ sudo quotaon [flags] [filesystem]
        -- $ sudo quotaoff [flags] [filesystem]
          --- where the flags can be:
              -a, --all                turn off for all filesystems
              -f, --full               turn off
              -u, --user               operate on user quotas
              -g, --group                        "
              -p, --print-state        print whether quotas are on or off
              -x, --xfs-command=cmd    perform XFS quota command
              -F, --format=formatname  operate on specific quota format
              -v, --verbose            print more messgaes
              -h, --help               display help text and exit
              -V, --version            display version information

        -- sudo quotaon -av /dev/sda6 / : group quotas turned on
        -- sudo quotaon -av /dev/sda6 /home : user quotas turned on
        -- sudo quotaoff -av /dev/sda6 / : group quotas turned off
        -- sudo quotaoff -av /dev/sda6 /home : user quotas turned off
        -- sudo quotaon -avu /dev/sda6 /home : user quotas turned on
        -- sudo quotaoff -avu /dev/sda6 /home : user quotas turned off
        -- sudo quotaon -avg /dev/sda6 /home : group quotas turned on
        -- sudo quotaoff -avg /dev/sda6 /home : group quotas turned off

        Note: quota operations will fail if aquota.user and aquota.group do not
exist

  Examining Quotas:
    - quota ( or quota -u) for current user
    - quota -g returns ure current group quota
    - with superuser for any user:
            -- sudo quota george
            -- sudo quota gracie

  Setting Quotas:
    only fields that can be modofied are 'soft and hard limits' only
    - edquota -u [username]
```

```
    - edquota -g [groupname]
    - edquota -u -p [userproto] [username] : used in scripts
    - edquota -u -p [groupproto] [groupname] : used in scripts
    - edquota -t : to set grace periods

    Note: soft limits may be exceeded for a grace period, hard limits may never
be exceeded

  Filesystem Usage:
    df (disk free) examines filesystem capacity and usage
    df -hTi

  Diskspace Usage:
    du (disk usage) shows how much space a directory and its subdirectories are
using on a filesystem
    - For current directory: $ du
    - to list all files not directories alone: $ du -a
    - human readable format: $ du -h
    - for specific directory: $ du -h somedir
    - display  only totals: $ du -s



<--------------------------------------------------------------------------------
------------------------>

                    Chapter 14. The Ext2/Ext3/Ext4 Filesystems

  Most used filesystem, with the ext4 being the latest version.


  Objectives:
    - describe the main features of the ext4 filesystem and how it is laid out on
disk.
    - explain the concepts of block groups, superblock, data blocks and inodes.
    - Use the dumpe2fs and tune2fs utilities.
    - list the ext4 filesystem enhancements.

  Ext4 History and Basics:
    Timeline:--> ext2 --> ext3 (with journalling) --> ext4 (extents added for
large filesystems)
    RHEL 7 uses XFS as default

  Ext4 Features:

            ext4
              |
              |
            blocks --->[512, 1024, 2048, 4096]
              |
        Pages of Memory -->[x86(4kb page size), x64(8kb page size)]
```

```
  Ext4 Layout:

             Disk Blocks
                 |
                 |
             Block Groups0 - n
                 |
         |--------------------------------------------------|
         |                                                  |
  block grp 0                                     block grp 1- n (boot sector
is absent)
         |

|-----------------------------|-------------------------|----------------|--------
---------|---------------|-----------|
    1024(boot sector)        Superblock            Grp Descriptors  Data block
bitmap    inode bitmap   inode Table  data blocks
                                  |
                                        |
                                  |
                                        |
                                  |
                                  pre-allocated to files to inprove speed
                                  |
                                  |
                                  |
                                  bit-fields:
                                    used to:
                                      -- ascertain whether or not the FS requires
checking when first mounted
                                      -- determine when FS was last checked, both
in date and number of mounts

  Block Groups:
    Same as above


  dumpefs:
    - scan filesystem information
    $ sudo dumpefs /dev/sda2

  tune2fs:
    -- used to change filesystem parameters
      --- change the maximum number of mounts between filesystem
checks(max-mount-count)
          $ sudo tune2fs -c 25 /dev/sda2
      --- change the time interval between checks (interval-between-checks)
          $ sudo tune2fs -i 10 /dev/sda2
```

      --- list the contents of the superblock including current values of parameters whwich can be changed:
          $ sudo tune2fs -l /dev/sda2

  Superblock Information:
    Contains information about the filesystem including:
        -- mount count and maximum count, set by tune2fs.
        -- block size, set by mkfs
        -- blocks per group
        -- free block count
        -- free inode count.
        -- operating system ID

  Data Blocks and Inodes:
    these are blocks whose bits contain 0 for each free block ot inode and 1 for eachused one.

  Ext4 Filesystem Enhancements:
    - is backwards-compatible with ext3 and ext2
    - increases the maximum fielsystem size to 1 EB (from 16 TB), and the maximum file to 16 TB (from 2 TB).
    - increase without limit the maximum number of subdirectories, which was limited to 32k in ext3
    - splits large files into thr largest possible extents instead of using indirect block mapping.
    - uses multiblock allocation
    - pre-allocate disk space for a file
    - uses allocate-on-flush
    - uses fast fsck
    - uses checksums for the journal which improves reliability
    - uses improved timestamps which is measured in nanoseconds
    - includes snapshot support.

<-->

Chapter 15. The XFS and btrfs Filesystems.

  these are important challengers to the ext4 filesystem.

  Objectives:
    - describe the XFS filesystem
    - maintain the XFS filesystem
    - describe the btrfs filesystem

  XFS Filesystem:
    originally made by SGI for IRIX OS
    - advatages include:

```
      -- handle:
          --- up to 16EB(exabytes) for the total filesystem
          --- up to 8EB(exabytes) for an individual file
      -- high performance:
          --- employing DMA(Direct Memory Access) I/O
          --- guaranteeing an I/O rate
          --- adjusting stripe size to match underlying RAID or LVM devices
      -- can journal quota information, but leads to decrease in recovery time
when a quota-enabled filesystem is uncleanly unmounted
      -- supports extended attributes

  XFS Filesystem Maintenance:
    - advantage:
      -- most maintenance tasks can be done on-line while the filesystem is fully
mounted, these include:
          --- defragmenting
          --- enlarging
          --- dumping/restoring
      -- backup and restore can be done with the native XFS utilities:
        --- xfsdump
        --- xfsrestore

  The btrfs Filesystem:
    B-tree file system:
      - advantages:
          -- addresses the lack of:-
              --- pooling,
              --- snapshots,
              --- checksums, and
              --- integral multi-device spanning


<--------------------------------------------------------------------------------
----------------------------------->

                          Chapter 16. Logical Volume Management(LVM)

  - permits having one logical filesystem span mutiple physical volumes and
partitions while appearing as a simple partition for normal use.
  - makes shrinking and expanding easy

  Objectives:
    - explain the concepts behind LVM
    - create logical volumes
    - display logical volumes
    - resize logocal volumes
    - use LVM snapshots

  LVM:
    - breaks up one virtual partition into multiple chunks, each of which can be
on different partitions and/or disks.
```

- Typical LVM structure or layout
- striping (splitting of data to more than one disk)

```
|--------------------------------------------------------------------------------
---------------------------|
  |                                                         LVM
                          |
  |
-------------------------------------------------------------------------------|
  |
  |         File Systems  |          /home(ext3)              ||
/data(xfs)              |     |
  |
|------------------------------------------------------------------------------|
  |
  |  Logical Volumes(LV)  | /dev/primary_vg/home_|v           ||
/dev/primary_vg/data_|v           |     |
  |
|------------------------------------------------------------------------------|
  |
  |   Volume Groups(VG)   |                         primary_vg
                       |     |
  |
|------------------------------------------------------------------------------|
  |
  |   Physical Volumes    | /dev/sdb1   ||   /dev/sdb2        ||     /dev/sdc1
 ||   /dev/sdc2       |     |
  |
|------------------------------------------------------------------------------|
  |

|--------------------------------------------------------------------------------
---------------------------|

     Partitions           |---------------||-------------|
|----------------||-------------------|
                          | /dev/sdb1     || /dev/sdb2   |              |
/dev/sdc1     ||   /dev/sdc2       |
                          |---------------||-------------|
|----------------||-------------------|
                          |------------------------------|
|-------------------------------------|
    Physical Drives       |             /dev/sdb         |              |
       /dev/sdc             |
                          |------------------------------|
|-------------------------------------|
```

LVM and RAID:

```
    - LVM can be built on RAID

  Volumes and Volume Group:
    - commands:
      -- vgcreate: create volume group
      -- vgextend
      -- vgreduce
      -- pvcreate: convert a partition to a physical volume
      -- pvdisplay: shows the physical volume
      -- pvmove: moves the data from one physical volume group to others
      -- pvremove

  Logical Volumes Utilities:
    - $ ls -lF /sbin/lv* --> doesn't work on ubuntu Trusty

  Creating Logical Volumes:
    - commands:
      -- lvcreate --> allocates logical volumes from within volume groups
      -- lvdisplay --> reports on available logoical volumes
    - filesystems are placed in logical volumes and are formatted with mkfs as
usual
    - steps:
       -- create partitions on disk drives (type 8e in fdsik)
       -- create physical volumes from the partitions                 --> $ sudo
pvcreate /dev/sdc1
       -- create the volumes group                                    --> $ sudo
vgcreate -s 16M vg /dev/sdb1
       -- allocate logical volumes from the volume group              --> $ sudo
lvcreate -L 50G -n mylvm vg
       -- format the logical volumes                                  --> $ sudo
mkfs -t ext4 /dev/vg/mylvm
       -- mount the logical volumes (also update /etc/fstab as needed) --> $
mkdir /mylvm, then --> $ sudo mount /dev/vg/mylvm /mylvm, then --> /dev/vg/mylvm
/mylvm ext4 defaults 0 0 ( to the /etc/fstab)

  Displaying Logical Volumes:
    - sudo pvdisplay or sudo pvdisplay /dev/sda5
    - sudo vgdisplay or sudo vgdisplay /dev/vg0
    - sudo lvdisplay or sudo lvdisplay /dev/vg0/lvm1

  Reszining Logical Volumes:
    -- with fs:
        -- shrink --> 1. shrink fs 2. shrink volume
        -- expand --> 1. expand volume, 2. expand fs
    -- utility: resizefs
  Examples of Resizing:
        sudo lvextend -L +500M /dev/vg/mylvm
        sudo resizefs /dev/vg/mylvm

    - shrink:
        sudo umount /mylvm
```

```
        sudo fsck -f /dev/vg/mylvm
        sudo resizefs /dev/vg/mylvm 200M
        sudo lvreduce -L 200M /dev/vg/mylvm
        sudo mount /dev/vg/mylvm

    - New versions of lvm utility:
      -- sudo
      -- sudo lvextend -r -L +100M /dev/vg/mylvm
      -- sudo lvreduce -r -L -100M /dev/vg/mylvm

    - volume group:
      -- sudo pvmove /dev/sdc1
      -- sudo vgreduce vg /dev/sdc1


  LVM Snapshots:
    - creates an exact copy of an existing logical volume
    - useful for:
      -- backups, application testing, and deploying VMs
    - creating snapshots:
      -- sudo lvcreate -l 128 -s -n mysnap /dev/vg/mylvm
      --- make a mount point and mount the snapshot:
          ---- mkdir /mysnap
          ---- mount -o ro /dev/vg/mysnap /mysnap
      --- to use and remove snapshot:
          ---- sudo umount /mysnap
          ---- sudo lvremove /dev/vg/mysnap
```

<--
-->

Chapter
17. RAID

- The use of RAID spreads I/O activity over multiple physical disks, rather than just one.
- Its purpose is to enhance data integrity and recoverability in case of failure, as well as to boost performance when used with modern
        storage devices.
- RAID has different levels with vary in their relative strengths in safety, performance, complexity and cost.

Objectives:
        - explain the concept of RAID.
        - summarize RAID levels.
        - configure a RAID device using the essential steps provided.
        - monitor RAID devices in multiple ways.
        - use hot spares.

RAID:

```
        - Meaning Reduntant Array of Independent Disk
        - RAID can be implemented in 'software' or in 'hardware'
        - simple implementation:-

                        512GB hard drives ---> RAID software implementation --->
1TB disk

        - Disadvantage of RAID hardware:
                -- if disk controller fails, it must be replaced by a compatible
controller unlike software implementation
                        which can allow the disk to work with any controller.
        - The essential features of RAID are:
                -- mirroring: writing the same data to more than one disk.
                -- striping: splitting of data to more than one disk.
                -- parity: extra data is stored to allow problem detection and
repair, yielding fault tolerance.
                -- Thus use of RAID can improve both performance and reliability.
        - RAID devices are typically created by combining partitions from several
disks together
        - mdadm is used to create and manage RAID devices, '/dev/mdX' is
equivalent to '/dev/sda1'
RAID Levels:
        - many levels exists based on the specification of increasing complexity
and use, these are:
                -- RAID 0: uses only striping.
                        --- advantages:
                                ---- data spread across multiple disks
                                ---- improved performance
                        --- disadvantages:
                                ---- no redundancy
                                ---- no stability
                                ---- no recovery capabilities
                -- RAID 1: uses only mirroring
                        --- advantages:
                                ---- good for recovery as each disk has a
duplicate
                -- RAID 5: uses rotating parity strip;
                        --- advantage:
                                ---- disk failure only causes drop in
performance, no loss of data
                                --Note: they must be at least 3 disks
                -- RAID 6: has striped disks with dual parity,
                        --- can handle loss of two disks
                        --- requires at least 4 disks
                        --- replaces RAID 5 due to its stress on hardware
                -- RAID 10: is a mirrored and striped data set.
                        --- at least 4 drives are needed.
                Note: adding more disks improves performance.
Software RAID Configuration:
        - Steps for setup includes:
                -- create partitions on each disk (type fd in fdisk)
```

```
                -- create RAID device with mdadm
                -- format RAID device
                -- add device to /etc/fstab
                -- mount RAID device
                -- capture RAID details to ensure persistence
        - The command: sudo mdadm -S --> used to stop RAID
        - Example:
                - create two partitions of type fd on disks /dev/sdb, /dev/sdc
                        -- sudo fdisk /dev/sdb, sudo fdisk /dev/sdc
                - set up the array
                        -- sudo mdadm --create /dev/md0 --level=1 --raid-disks=2
/dev/sdbX /dev/sdcX
                - format it
                        -- sudo mkfs.ext4 /dev/md0
                - add to configuration
                        -- sudo bash -c "mdadm --detail --scan >>
/etc/mdadm.conf"
                - mount it
                        -- sudo mkdir /myraid
                        -- sudo mount /dev/md0 /myraid
                        -- add /dev/md0 /myraid ext4 defaults 0 2
                - examine /proc/mdstat to see the RAID status as in:
                        -- cat /proc/mdstat
                - to stop
                        -- sudo mdadm -S /dev/md0

Monitoring RAIDs:
        - ways include:
                -- sudo mdadm --detail /dev/md0
                -- cat /proc/mdstat
                -- you can also use: mdmonitor, requires the configuration of
/etc/mdadm.conf
                -- status of RAID device /dev/mdX:
                        --- sudo mdadm --detail /dev/mdX
                -- show status of all RAID devices on the system
                        -- cat /proc/mdstat
                -- Monitor RAID via email:
                        --- MAILADDR eddie@haskel.com
                        --- starting email service with:
                                ---- sudo service mdmonitor start
                                ---- sudo chkconfig <mdmonitor | mdadm> on

RAID Hot Spares:
        - used to fix redundancies
        - create hot spares for RAID
                -- sudo mdadm --create /dev/md0 -l 5 -n3 -x 1 /dev/sda8 /dev/sda9
/dev/sda10 /dev/sda11
                -- switch '-x 1' tells RAID to use one hot spares
                -- testing the redundancy and hot spare of your array
                        --- sudo mdadm --fail /dev/md0 /dev/sdb2
                -- restoring tested drive:
```

```
--- sudo mdadm --remove /dev/md0 /dev/sdb2
--- sudo mdadm --add /dev/md0 /dev/sde2
```

<-->

Chapter 18. Local System Security

Objectives:
- assessing system security risks
- fashion and implement sound computer security policies and procedures
- efficiently protect BIOS and the boot loader with passwords
- use approprаite mount options, setuid and setgid to enhance security

Local System Security:
- can be defined in terms of:
  -- the systems ability to reguralarly do what it is supposed to do
  -- integrity and correctness of system
  -- ensuring that the ayatem is only available to those authorized to use it

- areas fo security include:
  -- physical
  -- local
  -- remote, and personal

Creating a security policy:
- basic structure of computer security:
  -- be simple and easy to understand
  -- get constantly updated
  -- be in the form of a written documentation
  -- describe both policies and procedures
  -- specify enforcemant actions
  -- specify actions to take in reesponse to a security breach
- basic structure of security policies:
  -- should be generic so its to follow
  -- must safe the data that needs protections
  -- deny access to required services and protect user policy
- policies should be updated on a regular basis

What to Include in the Policy:
- methods of protecting information from being read or copied by unauthorized personel
- protection of information from being altered or deleted without the permission of the ownership
- all services should be protected so they are available and not degraded in any manner without authorization
- aspects include:
  -- confidentiality
  -- data integrity
  -- availability
  -- consistency
  -- control

```
                -- audit
What Risks to Assess:
        - Questions are:
                -- what do I want to protect?
                -- what am I protecting it against?
                -- how much time, personnel, and money is needed to provide
adequate protection?
Choosing a Security Philosophy:
        - basic philosophies:
                -- anything not expressly permitted is denied
                -- anything not expressly forbidden is permitted
Some General Security Guidlines:
        - general guidelines:
                -- human factor is the weakest link:
                        --- educate your users and keep them happy
                -- no computer environment is invulnerable
                -- paranoia is a good thing:
                        --- be suspicious, vigilant, persevere when securing a
computer.
Updates and Security:
        - always aply updates and upgrades
Hardware Accessibility and Vulnerability:
        - physical access to servers and workstations should monitored
Hardware Access Guidelines:
        - steps include:
                -- locking down workstations and servers
                -- protecting your network links against access by people you do
not trusted
                -- protecting your keyboards where passwords are entered to
ensure the keyboards cannot be tampered with
                -- password protecting BIOS to prevent booting from live or
rescue CD/DVD or USB key

Protection of BIOS:
        - should be done with care

Protecting The Boot Loader with Password:
        - should be secured with the BIOS for full protection
        - steps includes:
                -- GRUB version 1:
                        --- run grub-md5-crypt
                        --- copy displayed hashed password
                        --- edit /boot/grub/grub.conf and add the line below the
timeout entry:
                                ---- password --md5
$lkl(8r8lkz.ljgsoorp)i4mmkkklxza
                -- GRUB version 2:
                                --- more complicated to set up
                                --- edit system configuration files in
/etc/grub.d
                                --- run update-grub
```

```
Filesystem Security: mount options:
        - nodev --> do not interpret character or block special devices on the
system
        - nosuid --> the set-user-identifier or set-group-identifier bits do not
take effective
        - noexec --> restric direct execution of any binaries on the mounted
filesystem
        - ro --> mount the filesystem in read-only mode as in:
                -- mount -o ro,noexec,nodev /dev/sda2 /mymountpt
                OR
                -- /dev/sda2 /mymountpt ext4 ro,noexec,nodev 0 0

setuid and setgid:
        - using this commands one can change the default behaviour of any program
to run with the permission of
                the owner rather than with that of the current user of the
program
Setting the setuid/setgid Bits:
        - commands:
                -- chmod u+s somefile
                -- chmod g+s somefile
                -- chmod g+s somedir --> files created in this directory are
group owned by the group owner of the directory.


<--------------------------------------------------------------------------------
----------------------------------------->

                                                Chapter 19. Linux Security
Modules
The responsiblity of protecting a system falls on:
        - application designers
        - Linux kernel developers and maintainers

Objectives:
        - understand how the Linux Security Modules framework works and how it is
deployed
        - list the various LSM implementationa available
        - delineate the main features of SELinux
        - explain the different modes and policies available
        - grasp the importsance of contexts and how to get and set them
        - know how to use the important SELinux utility programs

What Are Linux Security Modules?:
        - using 'mandatory access controls' linux kernel ensures a secure system
        - protocol for Implementation are:
                -- minimize changes to the kernel
                -- minimize overhead on the kernel
                -- permits flexibility and choice between different
Implementations. each of which is presented as a self-contained LSM (Linux
Security Module)
```

```
                -- basic scenerio:
                        --- request to kernel system ---> kernel invokes security
system

                        |

                        |

                        |                                 Yes

                        |                                  |

                        V                                  V
                                                                            does user
have right to make request ---> kernel carries out request

                        |

                        |<----No

                        |

                        V

kernel denies access to user
LSM Choices:
        - current LSMs are:
                -- SELinux
                -- AppArmmor
                -- Smack
                -- TOMOYO
SELinux Overview:
        - used to determine which processes can access which:-
         -- files
         -- directories
         -- ports, and
         -- other items on the system
        - works with these conceptual quantities:
                -- contexts: labels to files, processes, and ports, e.g. user,
role, and type
                -- rules: in terms of contexts, processes, files, ports, users
                -- policies: set of rules that describes what system-wide control
decisions should be made by SELinux

SELinux Modes:
        - ENFORCING: all SELinux code is operative and access is denied according
to policy. All violations are auditedn and logged.
        - PERMISSIVE: Enables SELinux code but only audits and warns about
operations that would be denied in enforcing mode.
        - DISABLED: completely disable SELinux kernel and application code
leaving the system without any of its protections.
```

- modes are selected and explained in a file (usually /etc/selinux/config) whose location varies by distribution (is often either at /etc/sysconfig/selinux or linked from there)
- utilities used here are:
    - -- sestatus: display current mode and policy
    - -- getenforece
    - -- setenforce --> $ sudo setenforce permissive
- to completely disable mode use:
    - -- configuration file, /etc/selinux/config and set SELINUX=disabled
    - -- kernel parameter, adding selinux=0 to the kernel parameter list when booting

SELinux Policies:

- usually /etc/sysconfig/selinux is used to set the SELinux policy.
- multiple policies are allowed but only one can be active at a time
- each policy is installed under /etc/selinux/[SELINUXTYPE]
    - -- COMMON POLICIES are:
        - --- targeted: default policy state in which SELinux is more restricted to targeted processes.
          user processes and init processes are not targeted
        - --- minimum: here only selected processes are targeted
        - --- MLS" Multi-Level Security policy is much more restrictive. Here all processes are are placed in a fined-grained
          security domains with particular policies

Context Utilities:

- SELinux contexts are:
    - -- User
    - -- role
    - -- Type
    - -- Level
- SELinux and Standard Command Line Tools:
    - -- ps, ls, cp, mv, and mkdir

SELinux Context Inheritance and Preservation:

- newly created files inherit the context from their parent directory
- moving or copying files, it is the context ofthe source directory that may be preserved

retorecon:

- resets file contexts, based on parent directory settings.
- $ restorecon -Rv /home/peter

semanage fcontext:

- used to manage the default context for a newly created directory
    - -- mkdir /virtualHosts
    - -- ls -Z
    - -- semanage fcontext -a -t httpd_sys_content_t /virtualHosts
    - -- ls -Z
    - -- effecting the change above:
        - --- restorecon -RFv /virtualHosts
        - --- ls -Z

```
Using SELinux Booleans:
        - SELinux policy behaviour can be configured at runtime without rewriting
the policy.
          using SELinux Booleans which are policy parameters that can be enabled
and disabled.
        - sudo semanage boolen -l (list booleans of current policy, including
current state and short description)
getsebool and setsebool:
        - getsebool -a --> prints only the boolean name and its current status.
        - setsebool: changing boolean status non-persistently with -P it
perssists
        - Examples:
                -- getsebool ssh_chroot_rw_homedir
                -- sudo setsebool ssh_chroot_rw_homedir on
                -- sudo reboot --> removes the above changes
                -- sudo setsebool -P ssh_chroot_rw_homedir on --> perssists
changes
Troubleshooting Tools:
        - example:
                -- echo 'File created at /rrot' > rootFile
                -- mv rootFile /var/www/html
                -- wget -O - localhost/rootFile
                        --- access denied
                -- check error log --> $ tail /var/log/messgaes
                        -- sealert -l dskjk03-kjfklj-kkhh-euykj-3455
                        -- on RHEL 7 --> grep httpd /var/log/audit/audit.log |
audit2allow -M mypol
                                --->> audit2allow (generates SELinux policies)
and audit2why (translates SELinux into a description of why the access was
denied)
                -- correcting access issue --> restorecon -Rv /var/www/html/
                -- wget -q -O - localhost/rootFile
                        --->> success
AppArmor:
        - is an LSM alternative to SELinux
        - used by SUSE, Ubuntu and other destros
        - AppArmor:
                - Provides Mandatory Access Control (MAC)
                - Allows asministrators to associate a security profile to a
program which restricts its capabilities
                - Is considered by some to be easier than SELinux
                - Is considered filesystem-neutral(no Security labels required)
        - AppArmor supplements the traditional UNIX Discreationary Access Control
(DAC) model by providing Mandatory Access Control(MAC)
        - in AppArmor: violations of the profile are logged and can be turned
into a profile based on the programs typical behaviour.

<--------------------------------------------------------------------------------
------------------------------------>

                                                        Chapter 20. processes
```

 - A process is an embodiment of a running application which may or may not consist of mutiple threads
 - processes have both attributes and well-delineated permissions.
 - they must in one of s number of enumerated states, most common being 'running' or 'sleeping'
 - important to know when a process is running in the 'user mode' or 'kernel mode' with elevated privileges for the latter.
 - a number of different ways exist to create 'child' processes and to set and modify their priorities.

Objectives:
- describe a process and the resources associated with it
- distinguish between processes, programs and threads
- understand process attributes, permissions and states
- know the difference between running in user and kernel modes
- describe 'daemon' processes
- understand how new processes are forked (created)
- use nice and renice to set and modify priorities

Processes, Programs and Threads:
- Process:
  - -- this is an executing program and associated resources, including environment, open files, signal handlers
  - -- the same program may be executing more than once simultaneously and thus be responsible for multiple processes
  - -- basic structure

```
                                        Program
                                               |
                                               |----> starts process
                                               |
                                               V
                                         process --> process ID (group ID)
                                               |
                                               V
        |-------------------------------------------------------| --> thread ID
(process ID)
        |               |               |               |               |
|               |           |
thread   thread   thread  thread  thread  thread  thread  thread
```

The init Process:
- this is the first user process on the system with a process ID = 1
- remains till shutdown and is the last process to terminate at shutdown
- ancestral parent of all other user processes

Processes:
- is an instance of a program in execution
- may be in a running state, or sleep state
- each process has:

```
                -- pid (Process ID),
                -- ppid (Parent Process ID),
                -- pgi (Process Group ID)
        - any process from the kernel has a '[]' around their names when '$ ps'
is run
        - when a parent process dies before a child, the ppid is set to 1 (init),
ot to 2 in newer systems using 'systemd'
        - when child process terminates normally or abnormally before its parents
which has not waited for it and examined its exit code is
                known as a 'zombie (or defunct)' process
        - the parent 'init' process checks on its adopted child processes and
lets those who have terminated die gracefully
        - Process are controlled 'scheduling', which is preemptive and is only
done by the kernel
        - largest PID has been limited to a 16-bit number, or 32768
        - pid_max can be altered in the /proc/sys/kernel/pid_max
Process Attributes:
        - these includes:
                -- the program being executed
                -- context (state)
                -- permissions
                -- associated resources
        - context switching: is done by the kernel

Controlling Processes with ulimit:
        - ulimit is a built in bash command that display or resets a number of
resources limits associated
                with processes running under a shell.
        - run $ ulimit -a
Controlling Processes with ulimit(cont.)
        - the 'ulimit -a' values can be changed to:
                -- restrict (limit to a value), or
                -- expand (increase the current limit or value)
        - Kinds of limit include:
                -- Hard: max value can't be exceeded, set only by root user
                -- soft: the current limiting value which a user can modify but
can't exceed hard limit
        - syntax:
                -- $ ulimit [options] [limit]
                        --- as in: ulimit -n 1600
                -- permanently changing this values to affect all currently
logged in shells one must modify /etc/security/limits.conf, then reboot

Process Permission and setuid:
        - process permissions are based on process ownership or user who invoked
it.
        - two types of programs exists:
                -- setuid programs, programs marked with an s execute bit. These
run with a different ownership from the current invoker
                -- non-setuid programs, that run based on the real user id
More on Process States:
```

```
        - Running:
                -- is either running or sitting in run queues
        - Sleeping:
                -- waiting on a request (usually I/O) that was made
        - Stopped:
                -- suspended process, using ctrl-Z or debugger
        - Zombie:
                -- enters the state when it terminates and no other process
(usually parent) has inquired about it state.
                -- here all resources are released except its exit state and its
entry in the process table
                -- it is adopted by the init (PID=1) or kthreadd (PID=2) when its
parent process dies.
Execution Modes:
        - any one process can be in either:
                -- user mode
                -- system mode
        - modes are enforced at hardware level not software level
User Mode:
        - all process are started in the use mode by default
        - all processes are isolaled in their own environment called process
resource isolation
        - system mode processes run in user mode unless they are jumping  into a
system called
Kernel Mode:
        - here CPU has access to all hardware on the system.
Daemons:
        - this is a background process whose sole purpose is to provide some
specific service to users of the system
        - basics of daemons:
                -- daemons can be quite s=efficient because they are only called
when needed
                -- many daemons are started at boot time
                -- daemons names usually end with d
                -- some examples are: httpd, udevd
                -- daemons may respond to external events (udevd), or elapsed
time (crond)
                -- daemons generally have no controlling terminal and no standard
input/output devices
                -- daemons sometimes provide better security control
Kernel-Created Processes:
        - the kernel directly creates two kinds of processes:
                -- internal kernel processes:
                        --- take care of maintanance work
                        --- theses often run as long as the system runs
                -- external user processes:
                        --- run in user space like normal applications but where
started by the kernel
                        --- they are few and short lived
                -- to see these:
                        --- $ ps -elf
```

```
Process Creating and Forking:
        - steps:
 thread 2
                                                    CLIENT REQUEST        /
                                                          Child Process --
                                                           /              \
                                                          /                thread
1
                        WEBSERVER                /
                -- Parent Process --->
                                                         \
                                                          \               thread
1
                                                           \              /
                                                          Child Process --
                                                        CLIENT REQUEST    \

  thread 2

Creating Processes in a Command Shell:
        - steps:
                -- user executes a command at the command shell:
                        ---> a new process is created (forked)
                        ---> a wait call puts the parent to sleep.
                        ---> the command is loaded onto the child's process space
vis the exec system call.
                        ---> the command conpletes executing, and the child
process dies vis the exit system call
                        ---> the parent shell is re-awakended by the death of the
child process and proceeds to issue a new shell command.
                        -->> with '&' the parent shell shell skips the wait
request and is free to issue a new command immediately
                        -->> exceptions are the 'echo' and 'kill' which are built
into the shell and no program files are loaded for them to run.
Using nice to Set Priorities:
        - process priorities are controlled using the nice and renice commands.
        - scale (least nice, highest priority)-20 to +19(most nice, lowest
priority)
        - syntax:
                -- nice -n 5 command [ARGS]
                -- nice -5 command [ARGS]
modofieddifying the Nice Value:
        - only the superuser can run nice or renice
        - editing the /etc/security/limit.conf
                -- renice +3 13848
                -- man renice for more info
Static and Shared Libraries:
        - programs are built using libraries of code.
        - Static:
                -- the code for the libraries functions are inserted into the
program at compile time and
```

```
                        does not change thereafter even if the library is updated
        - Shared:
                -- code is loaded at run time and if updated the program will use
the new library.
                -- advantages:
                        -- shared by applications
                        -- better memory useage
                        -- smaller more manageable executable sizes
                        -- called DLL (Dynamic Link Library)
Shared Library Versions:
        - name convention: sharedlib.so.N --> where N = version number
Finding Shared Libraries:
        - ldd: used to accertain what shared libraries an executable requires.
        - $ ldd /usr/bin/vi
        - ldconfig:
                -- usually run at boot time but ccan be run anytime and uses the
/etc/ld.so.conf
                -- shared libs should only be stored in system directories
```