**Linux Runtime Levels**

A run level is a state of **init** and the whole system that defines what system services are operating. Run levels are identified by numbers. Some system administrators use run levels to define which subsystems are working, e.g., whether X is running, whether the network is operational, and so on. Others have all subsystems always running or start and stop them individually, without changing run levels, since run levels are too coarse for controlling their systems. You need to decide for yourself, but it might be easiest to follow the way your Linux distribution does things.

The following table defines how most Linux Distributions define the different run levels. However, run-levels 2 through 5 can be modified to suit your own tastes.

**Table 9-1. Run level numbers**

|  |  |
| --- | --- |
| 0 | Halt the system. |
| 1 | Single-user mode (for special administration). |
| 2 | Local Multiuser with Networking but without network service (like NFS) |
| 3 | Full Multiuser with Networking |
| 4 | Not Used |
| 5 | Full Multiuser with Networking and X Windows(GUI) |
| 6 | Reboot. |

Services that get started at a certain runtime are determined by the contents of the various rcN.d directories. Most distributions locate these directories either at /etc/init.d/rcN.d or /etc/rcN.d. (Replace the N with the run-level number.)

In each run-level you will find a series of if links pointing to start-up scripts located in /etc/init.d. The names of these links all start as either K or S, followed by a number. If the name of the link starts with an S, then that indicates the service will be started when you go into that run level. If the name of the link starts with a K, the service will be killed (if running).

The number following the K or S indicates the order the scripts will be run. Here is a sample of what an /etc/init.d/rc3.d may look like.

|  |
| --- |
| # **ls -l /etc/init.d/rc3.d**  lrwxrwxrwx 1 root root 10 2004-11-29 22:09 K12nfsboot -> ../nfsboot  lrwxrwxrwx 1 root root 6 2005-03-29 13:42 K15xdm -> ../xdm  lrwxrwxrwx 1 root root 9 2004-11-29 22:08 S01pcmcia -> ../pcmcia  lrwxrwxrwx 1 root root 9 2004-11-29 22:06 S01random -> ../random  lrwxrwxrwx 1 root root 11 2005-03-01 11:56 S02firewall -> ../firewall  lrwxrwxrwx 1 root root 10 2004-11-29 22:34 S05network -> ../network  lrwxrwxrwx 1 root root 9 2004-11-29 22:07 S06syslog -> ../syslog  lrwxrwxrwx 1 root root 10 2004-11-29 22:09 S08portmap -> ../portmap  lrwxrwxrwx 1 root root 9 2004-11-29 22:07 S08resmgr -> ../resmgr  lrwxrwxrwx 1 root root 6 2004-11-29 22:09 S10nfs -> ../nfs  lrwxrwxrwx 1 root root 12 2004-11-29 22:40 S12alsasound -> ../alsasound  lrwxrwxrwx 1 root root 8 2004-11-29 22:09 S12fbset -> ../fbset  lrwxrwxrwx 1 root root 7 2004-11-29 22:10 S12sshd -> ../sshd  lrwxrwxrwx 1 root root 8 2005-02-01 09:24 S12xntpd -> ../xntpd  lrwxrwxrwx 1 root root 7 2004-12-02 20:34 S13cups -> ../cups  lrwxrwxrwx 1 root root 6 2004-11-29 22:09 S13kbd -> ../kbd  lrwxrwxrwx 1 root root 13 2004-11-29 22:10 S13powersaved -> ../powersaved  lrwxrwxrwx 1 root root 9 2004-11-29 22:09 S14hwscan -> ../hwscan  lrwxrwxrwx 1 root root 7 2004-11-29 22:10 S14nscd -> ../nscd  lrwxrwxrwx 1 root root 10 2004-11-29 22:10 S14postfix -> ../postfix  lrwxrwxrwx 1 root root 6 2005-02-04 13:27 S14smb -> ../smb  lrwxrwxrwx 1 root root 7 2004-11-29 22:10 S15cron -> ../cron  lrwxrwxrwx 1 root root 8 2004-12-22 20:35 S15smbfs -> ../smbfs |

How run levels start are configured in /etc/inittab by lines like the following:

|  |
| --- |
| l2:2:wait:/etc/init.d/rc 2 |

The first field is an arbitrary label, the second one means that this applies for run level 2. The third field means that **init** should run the command in the fourth field once, when the run level is entered, and that **init** should wait for it to complete. The /etc/init.d/rc command runs whatever commands are necessary to start and stop services to enter run level 2.

The command in the fourth field does all the hard work of setting up a run level. It starts services that aren't already running, and stops services that shouldn't be running in the new run level any more. Exactly what the command is, and how run levels are configured, depends on the Linux distribution.

When **init** starts, it looks for a line in /etc/inittab that specifies the default run level:

|  |
| --- |
| id:2:initdefault: |

You can ask **init** to go to a non-default run level at startup by giving the kernel a command line argument of single or emergency. Kernel command line arguments can be given via LILO, for example. This allows you to choose the single user mode (run level 1).

While the system is running, the **telinit** command can change the run level. When the run level is changed, **init** runs the relevant command from /etc/inittab.

The following are the 6 high level stages of a typical Linux boot process.  
  


**1. BIOS**

* BIOS stands for Basic Input/Output System
* Performs some system integrity checks
* Searches, loads, and executes the boot loader program.
* It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 of F2, but it depends on your system) during the BIOS startup to change the boot sequence.
* Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
* So, in simple terms BIOS loads and executes the MBR boot loader.

**2. MBR**

* MBR stands for Master Boot Record.
* It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
* MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
* It contains information about GRUB (or LILO in old systems).
* So, in simple terms MBR loads and executes the GRUB boot loader.

**3. GRUB**

* GRUB stands for Grand Unified Bootloader.
* If you have multiple kernel images installed on your system, you can choose which one to be executed.
* GRUB displays a splash screen, waits for few seconds, if you don’t enter anything, it loads the default kernel image as specified in the grub configuration file.
* GRUB has the knowledge of the filesystem (the older Linux loader LILO didn’t understand filesystem).
* Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this). The following is sample grub.conf of CentOS.

#boot=/dev/sda

default=0

timeout=5

splashimage=(hd0,0)/boot/grub/splash.xpm.gz

hiddenmenu

title CentOS (2.6.18-194.el5PAE)

root (hd0,0)

kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/

initrd /boot/initrd-2.6.18-194.el5PAE.img

* As you notice from the above info, it contains kernel and initrd image.
* So, in simple terms GRUB just loads and executes Kernel and initrd images.

**4. Kernel**

* Mounts the root file system as specified in the “root=” in grub.conf
* Kernel executes the /sbin/init program
* Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a ‘ps -ef | grep init’ and check the pid.
* initrd stands for Initial RAM Disk.
* initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

**5. Init**

* Looks at the /etc/inittab file to decide the Linux run level.
* Following are the available run levels
  + 0 – halt
  + 1 – Single user mode
  + 2 – Multiuser, without NFS
  + 3 – Full multiuser mode
  + 4 – unused
  + 5 – X11
  + 6 – reboot
* Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.
* Execute ‘grep initdefault /etc/inittab’ on your system to identify the default run level
* If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
* Typically you would set the default run level to either 3 or 5.

**6. Runlevel programs**

* When the Linux system is booting up, you might see various services getting started. For example, it might say “starting sendmail …. OK”. Those are the runlevel programs, executed from the run level directory as defined by your run level.
* Depending on your default init level setting, the system will execute the programs from one of the following directories.
  + Run level 0 – /etc/rc.d/rc0.d/
  + Run level 1 – /etc/rc.d/rc1.d/
  + Run level 2 – /etc/rc.d/rc2.d/
  + Run level 3 – /etc/rc.d/rc3.d/
  + Run level 4 – /etc/rc.d/rc4.d/
  + Run level 5 – /etc/rc.d/rc5.d/
  + Run level 6 – /etc/rc.d/rc6.d/
* Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.
* Under the /etc/rc.d/rc\*.d/ directories, you would see programs that start with S and K.
* Programs starts with S are used during startup. S for startup.
* Programs starts with K are used during shutdown. K for kill.
* There are numbers right next to S and K in the program names. Those are the sequence number in which the programs should be started or killed.
* For example, S12syslog is to start the syslog deamon, which has the sequence number of 12. S80sendmail is to start the sendmail daemon, which has the sequence number of 80. So, syslog program will be started before sendmail.

**Proc in Linux**

Under **Linux**, /**proc** includes a directory for each running process, including kernel processes, in directories named /**proc**/PID, where PID is the process number. Each directory contains information about one process, including: /**proc**/PID/cmdline, the command that originally started the process.

/proc is very special in that it is also a virtual filesystem. It's sometimes referred to as a process information pseudo-file system. It doesn't contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc). For this reason it can be regarded as a control and information centre for the kernel. In fact, quite a lot of system utilities are simply calls to files in this directory. For example, 'lsmod' is the same as 'cat /proc/modules' while 'lspci' is a synonym for 'cat /proc/pci'. By altering files located in this directory you can even read/change kernel parameters (sysctl) while the system is running.

He purpose and contents of each of these files is explained below:

* /proc/PID/cmdline => Command line arguments.
* /proc/PID/cpu => Current and last cpu in which it was executed.
* /proc/PID/cwd => Link to the current working directory.
* /proc/PID/environ => Values of environment variables.
* /proc/PID/exe => Link to the executable of this process.
* /proc/PID/fd => Directory, which contains all file descriptors.
* /proc/PID/maps => Memory maps to executables and library files.
* /proc/PID/mem => Memory held by this process.
* /proc/PID/root => Link to the root directory of this process.
* /proc/PID/stat => Process status.
* /proc/PID/statm => Process memory status information.
* /proc/PID/status => Process status in human readable form.

For more information, go through the below mentioned links:

[http](http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html)

[://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html](http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html)

<https://en.wikipedia.org/wiki/Procfs>

**Squid Proxy Server**

Squid is a full-featured web proxy cache server application which provides proxy and cache services for Hyper Text Transport Protocol (HTTP), File Transfer Protocol (FTP), and other popular network protocols. Squid can implement caching and proxying of Secure Sockets Layer (SSL) requests and caching of Domain Name Server (DNS) lookups, and perform transparent caching. Squid also supports a wide variety of caching protocols, such as Internet Cache Protocol (ICP), the Hyper Text Caching Protocol (HTCP), the Cache Array Routing Protocol (CARP), and the Web Cache Coordination Protocol (WCCP).

The Squid proxy cache server is an excellent solution to a variety of proxy and caching server needs, and scales from the branch office to enterprise level networks while providing extensive, granular access control mechanisms, and monitoring of critical parameters via the Simple Network Management Protocol (SNMP). When selecting a computer system for use as a dedicated Squid caching proxy server for many users ensure it is configured with a large amount of physical memory as Squid maintains an in-memory cache for increased performance.

**TCP Wrapper**

It is a host-based networking ACL system, used to filter network access to Internet Protocol servers on (Unix-like) operating systems such as Linux or BSD. It allows host or subnetwork IP addresses, names and/or ident query replies, to be used as tokens on which to filter for access control purposes.

**LPT Port:**

**LPT** (line print terminal) is the usual designation for a parallel **port** connection to a printer or other device on a personal computer. Most PCs come with one or two **LPT** connections designated as LPT1 and LPT2.

**init.d directory or file:**

The **init.d** directory contains a number of start/stop scripts for various services on your system.

* + - **/etc/init.d**

**Soft links:** if the original file is deleted, the content in the link file also deleted.

**Hard links:** even the content or total original file is deleted, the content in the link file is remains same (unchanged).

**Unlink:** to remove or delete the link that has been created earlier.

**Absolute Path:** referring to the file / directory along with the root.

**Relative Path:** referring using the current directory

**uname:** to get the details of the os (a- alls, s- kernel name, h- hostname of the system, r- release number, v- version number, m- n/w architecture, p- process type, i- h/w platform, o- os, )

**xargs:** searches for the text or string using grep command.

**Globbing:** The process of expanding non-specific filename containing wildcards.

* ? => single character
* \* => all or set of characters
* [] => for range of characters

**Quoting:**

**Double quotes:**  substitute the value of variables and commands

**Single quotes:**  preserves the input text

**Black slash:** removes the meaning of special characters.

**Default Port Numbers:**

**Port no Protocol**

20, 21 ftp

22 SSH

23 TELNET

25 SMTP => email server

53 DNS

67 BOOTP

80 HTTP

137, 138, 139 NetBIOS => webserver

443 HTTPS

3306 MySQL

8080 Jenkins

110 POP3 => email server

* yum utility to install the rpm based packages or files => linux
* dpkg utility to install the Debian based packages or files => ubuntu -> apt-get

**Archiving files:**

**Tar (tape archive)**

tar -cf filename => creates the file with .tar extension

tar -xf filename => extracts the file with .tar extension to a normal file

**Compression:**

**Gzip:** gzip to compress and gunzip to extract the files.

**Bzip2:** bzip2 for the compression and bunzip2 for the extraction of files.

**Zip:** zip filename.zip filename => to zip the file and unzip filename.zip to unzip the contents.

**Compression using tar:**

tar -zcf (gzip) filename => files will be compressed to .tar.gz or .tgz extension

tar -jcf (bzip2) filename => files will be compressed to .tar.bz2 or .tbz extension

**TCP/IP:** Transfer control protocol/ Internet Protocol

**UDP:** User Datagram Protocol

**HTTP:** Hyper Text Transfer Protocol

**HTTPS:** Hyper Text Transfer Protocol Secured

**ICMP:** Internet Control Message Protocol => for testing the verifying the n/w communication b/w the hosts.

**IANA:** Internet Assign Number Authorities

**NAT:** N/W Address Translation

**DNS:** Domain Name Service => to translate the domain name into IP addresses (/etc/resolv.conf)

**ping:** testing the connectivity of remote n/w device

**dig:** allows us to lookup IP addresses for DNS names

**netstat:** list n/w connections, routing and other n/w information

**route:** current route or net settings

**traceroute:** traces the route that a packet takes

**ifconfig:** current n/w settings => linux

**Ipconfig:** to find out IP address in windows operating system.

**ipaddr:** current IP addresses and n/w settings

**IPV4:** 32bit IP scheme

**IPV6:** 128bit IP scheme

**/var =>** contains the files that often change such as email, logs, etc.

**/var/tmp =>** contains the files that don’t get deleted on reboot

**/tmp =>** contains the files that do get deleted on system reboot

**/etc/services =>** port numbers allocation in Linux system

**Sticky bits (t):**

This bit should have always been called the "restricted deletion bit" given that's what it really connotes. When this mode bit is enabled, it makes a directory such that users can only delete files & directories within it that they are the owners of. (sonly the owners of the files can delete the files)

* **1 =>** to add the sticky bits
* **0 =>** removes the sticky bits

**Private IP Ranges:**

10.0.0.0 – 10.255.255.255 => Class A

172.16.0.0 – 172.31.255.255 => Class B

**Lsof** => list of open files

**Limits.conf** => to change the user limit for a file or directory

**Limits.conf:**

limits.conf - configuration file for the pam\_limits module

The pam\_limits.so module applies ulimit limits, nice priority and number of simultaneous login sessions limit to user login sessions. This description of the configuration file syntax applies to the /etc/security/limits.conf file and \*.conf files in the /etc/security/limits.d directory.

The syntax of the lines is as follows:

<domain> <type> <item> <value>

**cksum:**

The **cksum** command calculates and writes to standard output a [cyclic redundancy check](https://www.computerhope.com/jargon/c/crc.htm) (**CRC**) and [byte](https://www.computerhope.com/jargon/b/byte.htm) count for each input file. The **cksum** tool calculates its sum using a method similar to the FCS (frame check sequence) used to verify that Ethernet packets are transferred correctly

The checksum of a file is a simple way to check if its data has become corrupted when being transferred from one place to another. If the checksum value of the file is the same before and after being transferred

**Syntax:**

cksum myfile.txt

**Swap space**

Swap space in Linux is used when the amount of physical memory (RAM) is full.

* If the system needs more memory resources and the RAM is full, inactive pages in memory are moved to the swap space. While swap space can help machines with a small amount of RAM, it should not be considered a replacement for more RAM.
* Swap space is located on hard drives, which have a slower access time than physical memory.
* Swap space can be a dedicated swap partition (recommended), a swap file, or a combination of swap partitions and swap files.
* Swap should equal 2x physical RAM for up to 2 GB of physical RAM, and then an additional 1x physical RAM for any amount above 2 GB, but never less than 32 MB

**inodes in Linux**

The **inode (index node)** is a fundamental concept in the Linux and UNIX filesystem. Each object in the filesystem is represented by an inode. But what are the objects? Let us try to understand it in simple words. Each and every file under Linux (and UNIX) has following attributes:

=> File type (executable, block special etc.)  
=> Permissions (read, write etc.)  
=> Owner (UID)  
=> Group (GID)  
=> File Size  
=> File access, change and modification time (remember UNIX or Linux never stores file creation time, this is favorite question asked in UNIX/Linux sys admin job interview)  
=> File deletion time  
=> Number of links (soft/hard) and location of the file on hard disk  
=> Extended attribute such as append only or no one can delete file including root user (immutability)  
=> Access Control List (ACLs) and some other meta data about the file

All the above information stored in an inode. In short, the inode identifies the file and its attributes (as above). Each inode is identified by a unique inode number within the file system. Inode is also know as index number.

## **inode definition**

An inode is an entry in inode table, containing information (the metadata ) about a regular file and directory. An inode is a data structure on a traditional Unix-style file system such as ext3 or ext4.

An inode is a data structure on a traditional Unix-style file system such as UFS or ext3. An inode stores basic information about a regular file, directory, or other file system object.

## **How do I see file inode number?**

* $ ls -i /etc/passwd or filename
* $ls -il filename

You can also use stat command to find out inode number and its attribute:

* $ stat /etc/passwd

Searching a file on basis of inode number

* # find /root -inum 1150561

**Delete a file:** Deleting a file in linux decrements the link count and freeing the inode bumber to be reused.

**Move or Rename a file:** if destination is same filesystem as the source, Has no impact on inode number, it only changes the time stamps in inode table.

**Copy file:** cp allocates a free inode number and placing a new entry in inode table.

**To restore the corrupted file Linux (Found 3 answers in google – not sure which one is correct)**

**Process 1:**

* Install foremost

Sudo yum or apt-get install foremost

* Then determine the target device

Sudo fdisk -l

* Issue the foremost command

Sudo foremost -t all -i /dev/sba1 -o output dir

**Process 2:**

Fsck (File System Consistency check) => check and repair a Linux filesystem

* Open the terminal and type:

Fsck /dev/sda1 => checks for sda1 partition

**Note**: fsck cannot be used on a mounted partition. If you do so, there is a high chance that it will damage the filesystem.

* check your Home folder that resides on another partition, say sda2, use the following commands:

umount /home

fsck /dev/sda2

**Note**: you will need root/superuser permission to run the “fsck” command.

* You can also use “fsck” to check external drive, such as your thumb drive or SD card. For example:

umount /dev/sdb1 #thumb drive

sudo fsck /dev/sdb1

* If you are not sure of the partition number, you can use the command

sudo fdisk -l => to list out all the paritions

## **Advanced Usage:** There are a few parameters that you can add to “fsck” to make it more powerful.

### Auto repair filesystem when errors are detected

* During the filesystem check, if errors are detected, you can get “fsck” to auto repair the filesystem with the -a flag. For example:

fsck -a **/**dev**/**sda1

* Similarly, using the -y flag can get the job done as well:

fsck -y **/**dev**/**sda1

### Check all filesystems in one run

* If there are several filesystems in your computer, you can get fsck to check all of them at the same time with the -A flag.

fsck -A

* What it will do is to grab all the filesystem entries from /etc/fstab and scan them for errors. You can use it together with the -R and -y flag to prevent it from scanning the root filesystem and fix all errors, if there is any.

fsck -AR -y

### Exclude check on mounted filesystem

As mentioned earlier, fsck cannot be run on a mounted filesystem. If you are using the -A flag to scan all the filesystems, and some of them are mounted, you might damage those filesystems. A way to overcome this is to use the -M flag to prevent it from checking mounted system.

For example, running the command

fsck -M **/**dev**/**sdc1

returns nothing and a return code 0 (which means “no error”). No scan was done at all since all the filesystems are mounted.

### Specifying the filesystem type

* There are times when you just want to check filesystems of a specific type, say ext2. You can make use of the -t flag to specify the filesystem type to check. For example, the command

fsck -t ext4 **/**dev**/**sdc1

will scan the external drive only if it is in the ext4 format.

* Alternatively, you can combine with the -A flag to scan all filesystems of the specific type:

fsck -A -t ext4 -y

## **Force fsck to run everytime during bootup**

By default, Ubuntu will run fsck after every 30 bootups, but if you want the system to to do a “fsck” check everytime it boots up, all you need to do is to create an empty file call “forcefsck” and place it in the root folder. This will notify the system to do a “fsck” check everytime during boot up.

sudo touch **/**forcefsck

To change the checking frequency, you can make use of the command “tune2fs“.

The following command instructs the system to run “fsck” after every 30 boot up.

tune2fs -c 30 **/**dev**/**sdaX

You can also specify number of days instead of boot up:

sudo tune2fs -i 10d **/**dev**/**sdaX

**Process 3:**

* Open a terminal window and type the command **ews-restore** and hit enter. This must be done on an EWS linux lab machine, or over SSH with X-forwarding

ews-restore

**Network Bonding Linux:**

Bonding is nothing but Linux kernel feature that allows to aggregate multiple like interfaces (such as eth0, eth1) into a single virtual link such as bond0. The idea is pretty simple, get higher data rates and as well as link failover.

Linux allows binding of multiple network interfaces into a single channel/NIC using special kernel module called bonding.

The Linux bonding driver provides a method for aggregating multiple network interfaces into a single logical “bonded” interface. The behavior of the bonded interfaces depends upon the mode; generally speaking, modes provide either hot standby or load balancing services. Additionally, link integrity monitoring may be performed.

To create the bonding (configuration), refer to the following link.

<https://www.cyberciti.biz/tips/linux-bond-or-team-multiple-network-interfaces-nic-into-single-interface.html>

<https://wiki.linuxfoundation.org/networking/bonding>

<https://docs.oracle.com/cd/E37670_01/E41138/html/ch11s05.html>

**Resetting a password (root) in Linux:**

To reset your root password (or any other account’s password for that matter), there are essentially two different situations which require varying approaches.

**When you can use GRUB**

If you have GRUB installed and you have accesses to edit boot parameters of selected entries then the job is as easy as it can get. Follow along and you shall have root access in no time. Once you have root access you can pretty much do anything you want to do!

* Highlight the GRUB entry for the Linux installation that you want to reset the password for.
* Press ‘e’ to edit. Select the Kernel line. Add ‘single’ at the end of the kernel line. Press ‘b’ to boot. If your system still requires you to enter the root password, add **init=/bin/bash** at the end. Press ‘b’ to boot.
* Either you would be taken to the root prompt directly or shown the recovery menu from where you can choose the root prompt. Use **passwd <username>** to change the password for any account.
* Type **reboot** to reboot the system and then log in in with your new password.

Also, note that some distributions will create a recovery mode entry during install. If you have the recovery mode entry listed in GRUB, you need not do any of the above, just choose the recovery mode and then choose root prompt at the screen that follows.

**When you can’t use GRUB:**

Refer the below given link for the whole process to change the root password when you are not using GRUB

<http://www.makeuseof.com/tag/how-to-reset-any-linux-password/>

**$$ in Linux:**

* $$ => is the process ID (PID) of the script itself.
* $BASHPID => is the process ID of the current instance of Bash. This is not the same as the $$ => variable, but it often gives the same result.
* $? => is the return code of the last executed command.
* $# => is the number of arguments in $\*
* $\*=> is the list of arguments passed to the current process
* **$\_**=> The default parameter for a lot of functions.
* **$.** => Holds the current record or line number of the file handle that was last read. It is read-only and will be reset to 0 when the file handle is closed.
* **$/** => Holds the input record separator. The record separator is usually the newline character. However, if $/ is set to an empty string, two or more newlines in the input file will be treated as one.
* **$,**=> The output separator for the print() function. Nor-mally, this variable is an empty string. However, setting $, to a newline might be useful if you need to print each element in the parameter list on a separate line.
* **$\**=> Added as an invisible last element to the parameters passed to the print() function. Normally, an empty string, but if you want to add a newline or some other suffix to everything that is printed, you can assign the suffix to $.
* **$#**=> The default format for printed numbers. Normally, it's set to %.20g, but you can use the format specifiers covered in the section "Example: Printing Revisited" in Chapter 9to specify your own default format.
* **$%**=> Holds the current page number for the default file handle. If you use select() to change the default file handle, $% will change to reflect the page number of the newly selected file handle.
* **$=**=> Holds the current page length for the default file handle. Changing the default file handle will change $= to reflect the page length of the new file handle.
* **$-** => Holds the number of lines left to print for the default file handle. Changing the default file handle will change $- to reflect the number of lines left to print for the new file handle.
* **$~** => Holds the name of the default line format for the default file handle. Normally, it is equal to the file handle's name.
* **$^** => Holds the name of the default heading format for the default file handle. Normally, it is equal to the file handle's name with \_TOP appended to it.
* **$|**=> If nonzero, will flush the output buffer after every write() or print() function. Normally, it is set to 0.
* **$?** => Holds the status of the last pipe close, back-quote string, or system () function.
* **$&** => Holds the string that was matched by the last successful pattern match.
* **$`**=> Holds the string that preceded whatever was matched by the last successful pattern match.
* **$'**=> Holds the string that followed whatever was matched by the last successful pattern match.
* **$+** => Holds the string matched by the last bracket in the last successful pattern match. For example, the statement /Fieldname: (.\*)|Fldname: (.\*)/ && ($fName = $+)=> will find the name of a field even if you don't know which of the two possible spellings will be used.
* **$\*** => Changes the interpretation of the ^ and $ pattern anchors. Setting $\* to 1 is the same as using the /m option with the regular expression matching and substitution operators. Normally, $\* is equal to 0.
* **$0** => Holds the name of the file containing the Perl script being executed.
* **$<number>** => This group of variables ($1, $2, $3, and so on) holds the regular expression pattern memory. Each set of parentheses in a pattern stores the string that match the components surrounded by the parentheses into one of the $<number> variables.
* **$[** => Holds the base array index. Normally, it's set to 0. Most Perl authors recommend against changing it without a very good reason.
* **$]** => Holds a string that identifies which version of Perl you are using. When used in a numeric context, it will be equal to the version number plus the patch level divided by 1000.
* **$"** => This is the separator used between list elements when an array variable is interpolated into a double-quoted string. Normally, its value is a space character.
* **$;** => Holds the subscript separator for multidimensional array emulation. Its use is beyond the scope of this book.
* **$!** => When used in a numeric context, holds the current value of errno. If used in a string context, will hold the error string associated with errno.
* **$@**=> Holds the syntax error message, if any, from the last eval() function call.
* **$<** => This UNIX-based variable holds the read uid of the current process.
* **$>** => This UNIX-based variable holds the effective uid of the current process.
* **$)** => This UNIX-based variable holds the read gid of the current process. If the process belongs to multiple groups, then $) will hold a string consisting of the group names separated by spaces.
* **$:** => Holds a string that consists of the characters that can be used to end a word when word-wrapping is performed by the ^ report formatting character. Normally, the string consists of the space, newline, and dash characters.
* **$^D** => Holds the current value of the debugging flags. For more information.
* **$^F** => Holds the value of the maximum system file description. Normally, it's set to 2. The use of this variable is beyond the scope of this book.
* **$^I** => Holds the file extension used to create a backup file for the in-place editing specified by the -i command line option. For example, it could be equal to ".bak."
* **$^L** => Holds the string used to eject a page for report printing.
* **$^P** => This variable is an internal flag that the debugger clears so it will not debug itself.
* **$^T** => Holds the time, in seconds, at which the script begins running.
* **$^W** => Holds the current value of the -w command line option.
* **$^X** => Holds the full pathname of the Perl interpreter being used to run the current script.

**Netstat**

(network statistics) is a command line tool for monitoring network connections both incoming and outgoing as well as viewing routing tables, interface statistics etc.

tcp - transfer communication protocol

udp- user datagram protocol

netstat -plnt (indicates the port number and PID/Program name of tcp)

netstat -plnu (indicates the port number and PID/Program name of udp)

to check port ~ netstat -anlp | grep 22

to check app ~ netstat -anlp | grep java

**Telnet**

* Is used for interactive communication with another host using telnet protocol
* telnet host :portnumber

**find and grep commands:**

* find . -type d -mtime +1 -exec rm -rf {} \; (deletes the directories created with in a day)
* **du -xh / |grep '^\S\*[0-9\.]\+G'|sort –rn**
* **find / -printf '%s %p\n'| sort -nr | head -10**
* **find / -xdev -type f -size +100M -exec ls -la {} \; | sort -nk 5**

**SAR COMMANDS :**

* sudo yum install package (or) rpm -ivh package.rpm
* download : wget url
* scp –i “pem” <source file> user@dns:location

**SED:**

The primary use of the Linux command SED (short for stream editor) is to modify each line of a file or stream by replacing specified parts of the line. It makes basic text changes to a file or input from a pipeline.

* Used to find and replace any string and also to print particular line in a file or to delte particular line in a file we use sed command.
* sed -i 's/thrinath/3nath/g' text.txt
* sed -n '3p' text.txt (to print the 3rd line)
* sed -n '3,5p' text.txt (to print 3rd to 5th line)
* sed -i '3d' text.txt (deletes the 3rn line in text.txt)
* replace the name Thrinath with 3nath from line 5 to 15 in a file text.txt
* sed –n ‘5,15p’ text.txt | sed –i 's/thrinath/3nath/g'

**AWK in Linux:**

The AWK command is a powerful method for processing or analyzing text files—in particular, data files that are organized by lines (rows) and columns.

Simple AWK commands can be run from the command line. More complex tasks should be written as AWk programs (so-called AWK scripts => Aho, Weinberger, and Kernighan) to a file.

The basic format of an awk command looks like this:

awk 'pattern {action}' input-file > output-file

For more information:

<https://www.computerhope.com/unix/uawk.htm>

**Other commands in Linux**

* Inode, also called as index number, identifies the file and its attributes – use **-I flag**
* Few other important commands in Linux include:
* To search for files and directories – **find. -name “string” -type f/d**
* To search within a specified period - **-mtime -/+n**
* To search within the contents of a file – **grep ‘string’ file\_path**
* To list all processes on the OS – **ps -eaf**
* To find and replace a string – **sed ‘s/find\_term/replace\_term/g’ file\_path**
* To redirect the output into next command or file - **>**
* To pass two or more commands at a time - **| (pipe)**
* To display various network related information – **netstat**

-a – all ports

-t – tcp

-u – udp

-l – listening

-s – statistics

-p – PID and program name

-r – kernel routing

* To communicate with another host using Telnet protocol – **telnet [host or IP [port] ]** (if the telnet server is running, it will be listening on tcp 23 by default)
* To kill a process – kill -3 PID
* To login to a remote machine and execute command – **ssh remote\_Host**
* To securely copy files between remote hosts – **Scp source\_file\_path user@dest\_host:dest\_path**
* To check the file systems disk space usage – **df -h (-T type, -m MB, -k KB)**
* To check the disk usage of the directories and sub directories and files – **du -h** (-s summarize)
* To find top 10 files using most disk space – **du -a | sort -nr | head -n 10**
* To monitor processes based on cpu/memory usage – **top** (M -memory, P – CPU usage, N – by PID, T – running time, i – idle processes).
* To automate the routine tasks – **crontab -e**
* To preserve time stamp while copying – **cp -p**
* To display firewall status – **iptables -L -n**

## **What is LVM and what is its role?**

LVM stands for Logical Volume Manager. It is used to resize the file system online.

## **What is the role of Ivextend and Ivreduce?**

They are commands used to extend or reduce the volume.

## **What is umask?**

umask stands for “user file creation mask”. It is used to determine those settings of a mask that control file permission data and directory data.

## **In the context of Linux, what does POSIX stand for?**

POSIX stands for Portable Operating System Interface for Computer Environments. It is the de facto standard for ensuring compatibility between UNIX versions.

## **How is Linux connected to UNIX?**

The kernel that Linux uses resembles that of UNIX but does not require UNIX code.

## **In the context of Linux, what is GPL?**

GPL stands for General Public License. It was originally created to protect the GNU project.

## **Explain the GNU project.**

GNU is a project that was initiated by Richard Stallman in 1983 at MIT. The project began with the idea of providing control and freedom to the users in using software. As part of the GNU project, users are free to use, run, copy and share software.

## **How do you turn a Linux system into a proxy server?**

Using the Squid service, you can turn a Linux system into a proxy server.

## **What is LILO in the context of Linux?**

LILO is a boot loader for Linux. It is used for loading the Linux operating system into the main system memory before beginning operations.

## **Define BIOS, MBR and GRUB.**

BIOS stands for Basic Input/Output System. MBR is Master Boot Record and GRUB is Graand Unified Bootloader.

## **What is the difference between home directory and working directory?**

Home directory is the default directory when a user logs in whereas working directory is the user’s current working directory.

## **In Linux, how do you track system events?**

To track events, a daemon called syslogd is used.

## **What do you do when you encounter a suspicious IP?**

* We should first block the suspicious IP by integrating tcp\_wrapper.
* Next, we need to enable the “tcp\_wrapper=YES” parameter in the configuration file at ‘/etc/vsftpd.conf’.
* Final step – include the suspicious IP in the ‘host.deny’ file in ‘/etc/host.deny’.

## **Describe Telnet and SSH.**

Both Telnet and SSH are communication protocols that are used to manage systems remotely. While SSH requires exchange of keys, Telnet transmits data in plain text. Hence, SSH is said to be more secure than Telnet.

## **State the difference between the locate and slocate commands.**

Locate searches for a file with the latest entries while slocate searches for files that users have accessed most recently.

## **How many partitions do you need to install Linux?**

You need at least two partitions to install Linux on your system.

## **How do you review boot messages in Linux?**

By using the dmesg command. Dmesg will pull out boot messages stored in the kernel ring buffer.

## **What are symbolic links?**

Symbolic links are “shortcut keys” in Linux. These links point to specific programs, files or directories.

## **What do Hard Links do?**

Hard links point directly to the physical file located on the disk, and not on the path name.

## **What does pwd stand for?**

In Linux, it stands for print working directory.

## **Explain the three different permissions in Linux.**

* Read: Gives users the permission to read files or list directories
* Write: Gives users the permission to write to the file of new files and directories
* Execute: Gives users the permission to run the file or lookup a specific file within a directory

## **What is the # symbol used for?**

# is used for creating new comments.

**What is Linux?**

Linux is an operating system based on UNIX, and was first introduced by Linus Torvalds. It is based on the Linux Kernel, and can run on different hardware platforms manufactured by Intel, MIPS, HP, IBM, SPARC and Motorola. Another popular element in Linux is its mascot, a penguin figure named Tux.

**What is the difference between UNIX and LINUX?**

Unix originally began as a propriety operating system from Bell Laboratories, which later on spawned into different commercial versions. On the other hand, Linux is free, open source and intended as a non-propriety operating system for the masses.

**What is BASH?**

BASH is short for Bourne Again SHell. It was written by Steve Bourne as a replacement to the original Bourne Shell (represented by /bin/sh). It combines all the features from the original version of Bourne Shell, plus additional functions to make it easier and more convenient to use. It has since been adapted as the default shell for most systems running Linux.

**What is Linux Kernel?**

The Linux Kernel is a low-level systems software whose main role is to manage hardware resources for the user. It is also used to provide an interface for user-level interaction.

**What is LILO?**

LILO (Linux Loader) is a boot loader for Linux. It is used mainly to load the Linux operating system into main memory so that it can begin its operations.

**What is a swap space?**

A swap space is a certain amount of space used by Linux to temporarily hold some programs that are running concurrently. This happens when RAM does not have enough memory to hold all programs that are executing.

**What is the advantage of open source?**

Open source allows you to distribute your software, including source codes freely to anyone who is interested. People would then be able to add features and even debug and correct errors that are in the source code. They can even make it run better, and then redistribute these enhanced source code freely again. This eventually benefits everyone in the community.

**What are the basic components of Linux?**

Just like any other typical operating system, Linux has these components:

kernel, shells and GUIs, system utilities, and application program. What makes Linux advantageous over other operating system is that every aspect comes with additional features and all codes for these are downloadable for free.

**Does it help for a Linux system to have multiple desktop environments installed?**

In general, one desktop environment, like KDE or Gnome, is good enough to operate without issues. It’s all a matter of preference for the user, although the system allows switching from one environment to another. Some programs will work on one environment and not work on the other, so it could also be considered a factor in selecting which environment to use.

**What is the basic difference between BASH and DOS?**

The key differences between the BASH and DOS console lies in 3 areas:

* BASH commands are case sensitive while DOS commands are not
* under BASH, / character is a directory separator and \ acts as an escape character. Under DOS, / serves as a command argument delimiter and \ is the directory separator
* DOS follows a convention in naming files, which is 8-character file name followed by a dot and 3 characters for the extension. BASH follows no such convention.

**What is the importance of the GNU project?**

This so-called Free software movement allows several advantages, such as the freedom to run programs for any purpose and freedom to study and modify a program to your needs. It also allows you to redistribute copies of a software to other people, as well as freedom to improve software and have it released to the public.

**Describe the root account.**

The root account is like a systems administrator account, and allows you full control of the system. Here you can create and maintain user accounts, assigning different permissions for each account. It is the default account every time you install Linux.

**What is CLI?**

CLI is short for Command Line Interface. This interface allows user to type declarative commands to instruct the computer to perform operations. CLI offers an advantage in that there is greater flexibility. However, other users who are already accustom with using GUI find it difficult to remember commands including attributes that come with it.

**What is GUI?**

GUI, or Graphical User Interface, makes use of images and icons that users click and manipulate as a way of communicating with the computer. Instead of having to remember and type commands, the use of graphical elements makes it easier to interact with the system, as well as adding more attraction through images, icons and colors.

**How do you open a command prompt when issuing a command?**

To open the default shell (which is where the command prompt can be found), press Ctrl-Alt-F1. This will provide a command line interface (CLI) from which you can run commands as needed.

**How can you find out how much memory Linux is using?**

From a command shell, use the “concatenate” command: cat /proc/meminfo for memory usage information. You should see a line starting something like: Mem: 64655360, etc. This is the total memory Linux thinks it has available to use.

**What is typical size for a swap partition under a Linux system?**

The preferred size for a swap partition is twice the amount of physical memory available on the system. If this is not possible, then the minimum size should be the same as the amount of memory installed.

**What are symbolic links?**

Symbolic links act similarly to shortcuts in Windows. Such links point to programs, files or directories. It also allows you instant access to it without having to go directly to the entire pathname.

**Does the Ctrl+Alt+Del key combination work on Linux?**

Yes, it does. Just like Windows, you can use this key combination to perform a system restart. One difference is that you won’t be getting any confirmation message and therefore, reboot is immediate.

**How do you refer to the parallel port where devices such as printers are connected?**

Whereas under Windows you refer to the parallel port as the LPT port, under Linux you refer to it as /dev/lp . LPT1, LPT2 and LPT3 would therefore be referred to as /dev/lp0, /dev/lp1, or /dev/lp2 under Linux.

**Are drives such as hard drive and floppy drives represented with drive letters?**

No. In Linux, each drive and device has different designations. For example, floppy drives are referred to as /dev/fd0 and /dev/fd1. IDE/EIDE hard drives are referred to as /dev/hda, /dev/hdb, /dev/hdc, and so forth.

**How do you change permissions under Linux?**

Assuming you are the system administrator or the owner of a file or directory, you can grant permission using the chmod command. Use + symbol to add permission or – symbol to deny permission, along with any of the following letters: u (user), g (group), o (others), a (all), r (read), w (write) and x (execute). For example, the command chmod go+rw FILE1.TXT grants read and write access to the file FILE1.TXT, which is assigned to groups and others.

**In Linux, what names are assigned to the different serial ports?**

Serial ports are identified as /dev/ttyS0 to /dev/ttyS7. These are the equivalent names of COM1 to COM8 in Windows.

**How do you access partitions under Linux?**

Linux assigns numbers at the end of the drive identifier. For example, if the first IDE hard drive had three primary partitions, they would be named/numbered, /dev/hda1, /dev/hda2 and /dev/hda3.

**What are hard links?**

Hard links point directly to the physical file on disk, and not on the path name. This means that if you rename or move the original file, the link will not break, since the link is for the file itself, not the path where the file is located.

**What is the maximum length for a filename under Linux?**

Any filename can have a maximum of 255 characters. This limit does not include the path name, so therefore the entire pathname and filename could well exceed 255 characters.

**What are filenames that are preceded by a dot?**

In general, filenames that are preceded by a dot are hidden files. These files can be configuration files that hold important data or setup info. Setting these files as hidden makes it less likely to be accidentally deleted.

**Explain virtual desktop.**

This serves as an alternative to minimizing and maximizing different windows on the current desktop. Using virtual desktops, each desktop is a clean slate where you can open one or more programs. Rather than minimizing/restoring all those programs as needed, you can simply shuffle between virtual desktops with programs intact in each one.

**How do you share a program across different virtual desktops under Linux?**

To share a program across different virtual desktops, in the upper left-hand corner of a program window look for an icon that looks like a pushpin. Pressing this button will “pin” that application in place, making it appear in all virtual desktops, in the same position onscreen.

**What does a nameless (empty) directory represent?**

This empty directory name serves as the nameless base of the Linux file system. This serves as an attachment for all other directories, files, drives and devices.

Daemons are services that provide several functions that may not be available under the base operating system. Its main task is to listen for service request and at the same time to act on these requests. After the service is done, it is then disconnected and waits for further requests.

**How do you switch from one desktop environment to another, such as switching from KDE to Gnome?**

Assuming you have these two environments installed, just log out from the graphical interface. Then at the Log in screen, type your login ID and password and choose which session type you wish to load. This choice will remain your default until you change it to something else.

**How does case sensitivity affect the way you use commands?**

When we talk about case sensitivity, commands are considered identical only if every character is encoded as is, including lowercase and uppercase letters. This means that CD, cd and Cd are three different commands. Entering a command using uppercase letters, where it should be in lowercase, will produce different outputs.

**What are environmental variables?**

Environmental variables are global settings that control the shell’s function as well as that of other Linux programs. Another common term for environmental variables is global shell variables.

**What are the different modes when using vi editor?**

There are 3 modes under vi:

* Command mode – this is the mode where you start in
* Edit mode – this is the mode that allows you to do text editing
* Ex mode – this is the mode wherein you interact with vi with instructions to process a file

**Is it possible to use shortcut for a long pathname?**

Yes, there is. A feature known as filename expansion allows you to do this using the TAB key. For example, if you have a path named /home/iceman/assignments directory, you would type as follows: /ho[tab]/ice[tab]/assi[tab]. This, however, assumes that the path is unique, and that the shell you’re using supports this feature.

**What is redirection?**

Redirection is the process of directing data from one output to another. It can also be used to direct an output as an input to another process.

**What is grep command?**

grep a search command that makes use of pattern-based searching. It makes use of options and parameters that is specified along the command line and applies this pattern into searching the required file output.

**What could possibly be the problem when a command that was issued gave a different result from the last time it was used?**

One highly possible reason for getting different results from what seems to be the same command has something to do with case sensitivity issues. Since Linux is case sensitive, a command that was previously used might have been entered in a different format from the present one. For example, to lists all files in the directory, you should type the command ls, and not LS. Typing LS would either result in an error message if there is no program by that exact name exist, or may produce a different output if there is a program named LS that performs another function.

**What are the contents in /usr/local?**

It contains locally installed files. This directory actually matters in environments where files are stored on the network. Specifically, locally-installed files go to /usr/local/bin, /usr/local/lib, etc.). Another application of this directory is that it is used for software packages installed from source, or software not officially shipped with the distribution.

**How do you terminate an ongoing process?**

Every process in the system is identified by a unique process id or pid. Use the kill command followed by the pid in order to terminate that process. To terminate all process at once, use kill 0.

**How do you insert comments in the command line prompt?**

Comments are created by typing the # symbol before the actual comment text. This tells the shell to completely ignore what follows.

For example: “# This is just a comment that the shell will ignore.”

**What is command grouping and how does it work?**

You can use parentheses to group commands. For example, if you want to send the current date and time along with the contents of a file named OUTPUT to a second file named MYDATES, you can apply command grouping as follows: (date cat OUTPUT) > MYDATES

**How do you execute more than one command or program from a single command line entry?**

You can combine several commands by separating each command or program using a semicolon symbol. For example, you can issue such a series of commands in a single entry:

|  |
| --- |
| ls –l cd .. ls –a MYWORK which is equivalent to 3 commands: ls -l cd.. ls -a MYWORK |

\*\*Note that this will be executed one after the other, in the order specified.

**Write a command that will look for files with an extension “c”, and has the occurrence of the string “apple” in it.**

Find ./ -name “\*.c” | xargs grep –i “apple”

**Write a command that will display all .txt files, including its individual permission.**

|  |
| --- |
| ls -a -l \*.txt |

**Write a command that will do the following:  
-look for all files in the current and subsequent directories with an extension c,v  
-strip the,v from the result (you can use sed command)  
-use the result and use a grep command to search for all occurrences of the word ORANGE in the files.**

|  |
| --- |
| Find ./ -name “\*.c,v” | sed ‘s/,v//g’ | xargs grep “ORANGE” |

**What, if anything, is wrong with each of the following commands?  
a) ls -l-s  
b) cat file1, file2  
c) ls – s Factdir**

Answers:  
a) there should be space between the 2 options: ls -l -s  
b) do not use commas to separate arguments: cat file1 file2  
c) there should be no space between hyphen and option label: ls –s Factdir

**What is the command to calculate the size of a folder?**

To calculate the size of a folder use the command **du –sh folder1.**

**How can you find status of a process?**

Use the command => ps ux

**How can you check the memory status ?**

You can use the command

free -m  to display output in MB

free -g  to display output in GB

**Explain how to color the Git console?**

To color the Git console, you can use the command **git config—global color.ui auto.** In the command, the color.ui variable sets the default value for variable such as **color.diff** and **color.grep.**

**How can you append one file to another in  Linux?**

To append one file to another in Linux you can use command **cat file2 >> file 1.** The operator >> appends the output of the named file or creates the file if it is not created.  While another command **cat file 1 file 2 > file 3** appends two or more files to one.

**Explain how you can find a file using Terminal?**

To find a file you have to use command, **find . –name “process.txt”** .  It will look for the current directory for a file called process.txt.

**Explain how you can create a folder using Terminal?**

To create a folder, you have to use command **mkdir.** It will be something like these :  ~$ mkdir Guru99

**Explain how you can view the text file using Terminal?**

To view the text file, go to the specific folder where the text files are located by using the command **cd** and then type **less filename.txt.**

**Explain how to enable curl on Ubuntu LAMP stack?**

To enable curl on Ubuntu , first install libcurl, once done use following command **sudo/etc/init .d /apache2 restart** or **sudo service apache2 restart.**

**Explain how to enable root loging in Ubuntu?**

The command which enables root loging is

#sudo sh-c ‘echo “greater-show-manual-login=true” >>/etc/lightdm/lightdm.conf’

**How you can run an Linux program in the background simultaneously when you start your Linux Server?**

By using **nohup.**It will stop the process receiving the **NOHUP** signal and thus terminating it you log out of the program which was invoked with.  **&** runs the process in the background.

**Explain how to uninstall the libraries in Linux?**

To uninstall the libraries in Linux, you can use command

sudo apt – get remove library\_name

What happens at the boot up process?

* When you turn on the power, the power is distributed to all parts of the system.
* POST – Power on Self-Test
* BIOS initializes tests to make sure devices connected are working all right like printer, keyboard, mouse, speakers
* The third step is BIOS check the hard drive and Boot disk / OS. Once it is confirmed that there is OS, This OS is loaded into RAM. When this process is done we will see the screen.

**Local keyword In BASH SCRIPT?**

(you cannot access from outside of the function) If I am writing a script and if I say within the function I am declaring some variable some x=1 if I say local that variable valid is only within that function, we can declare it but its accessible within the function.

**What is CoreOS, and what are alternatives?**

CoreOS is stripped down linux distribution meant for running containters, mainly with its own rkt format but others are also supported. It was initially based on ChromeOS and supported Docker. The alternatives to this are canonical's ubuntu snappy or red hat enterprise linux atomic host. Of course, Containers can also be run on regular Linux system.

**What is Kickstart?**

It is a way to install Red Hat based systems by automated way. During manual install process, Anaconda installer creates file anaconda-ks.cfg which then can be used with system-config-kickstart tool to install same configuration automatically on multiple systems.

**Find length of string**: expr length "aeargazalabaa"

Shell script

* Bash environment
* Core bash configuration files
* Bash environment variables
* What makes a file a bash script?
* Exit codes and return status
* Pipes and redirects
* Signals

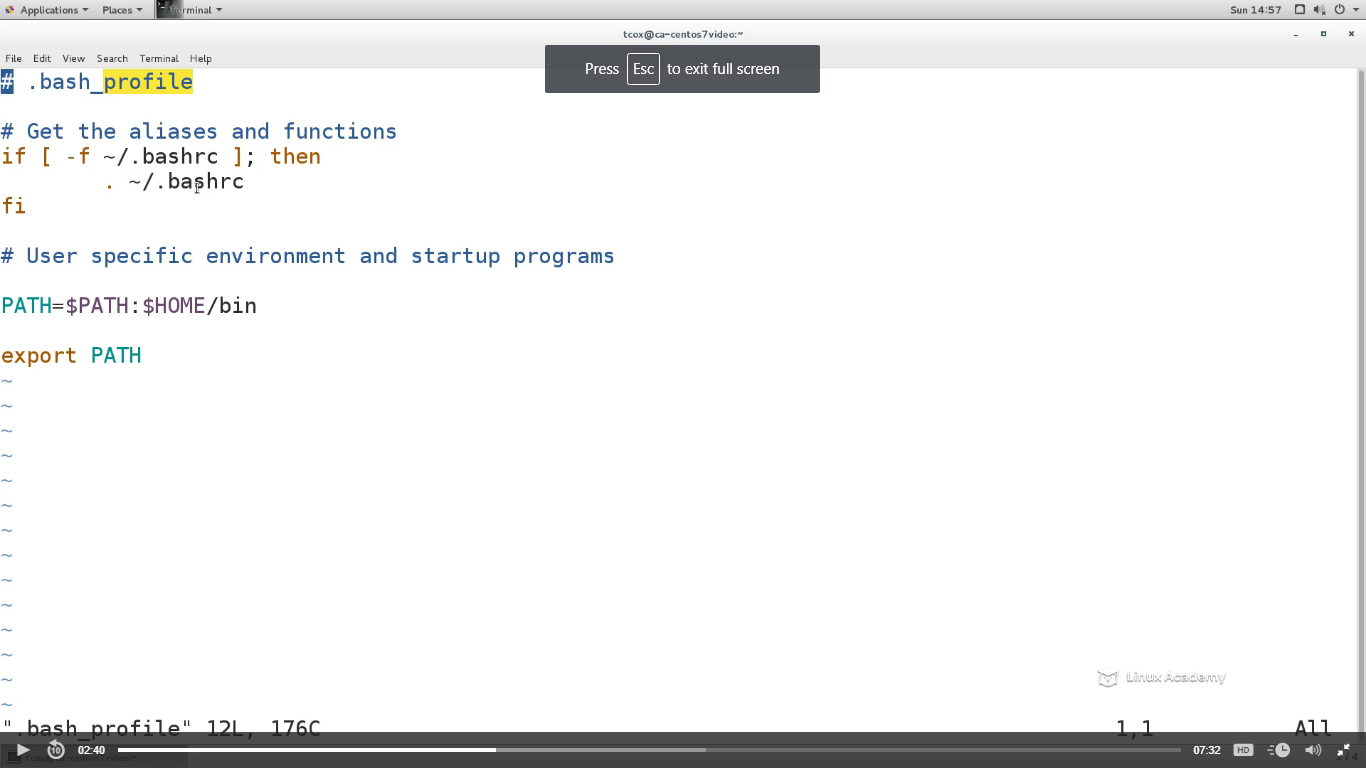
**Scripting:**

* Execute permissions and PATH
* The runtime environment
* Comments
* Static variables
* Dynamic variables
* I/O Handling
* Conditionals
* Flow control
* Declaring functions
* Arrays
* Parsing runtime arguments
* Value testing
* Functions
* Code blocks
* Sub shell
* Special structures (&&, ||, {})
* How to do math
* Handling errors within a script
* Advanced I/O (IPC, signals, interactive user input, redirection, file handling)
* Advanced find
* External dependencies (sourcing external file, libraries)
* Date and time (timezones, formatting, adding samples to files)
* Metadata
* Directory and file attributes
* Automation of common system administration activities
* Using in conjunction with CRON, things to be aware of

**CORE CONCEPTS:**

**Bash files - .bash\_profile:**

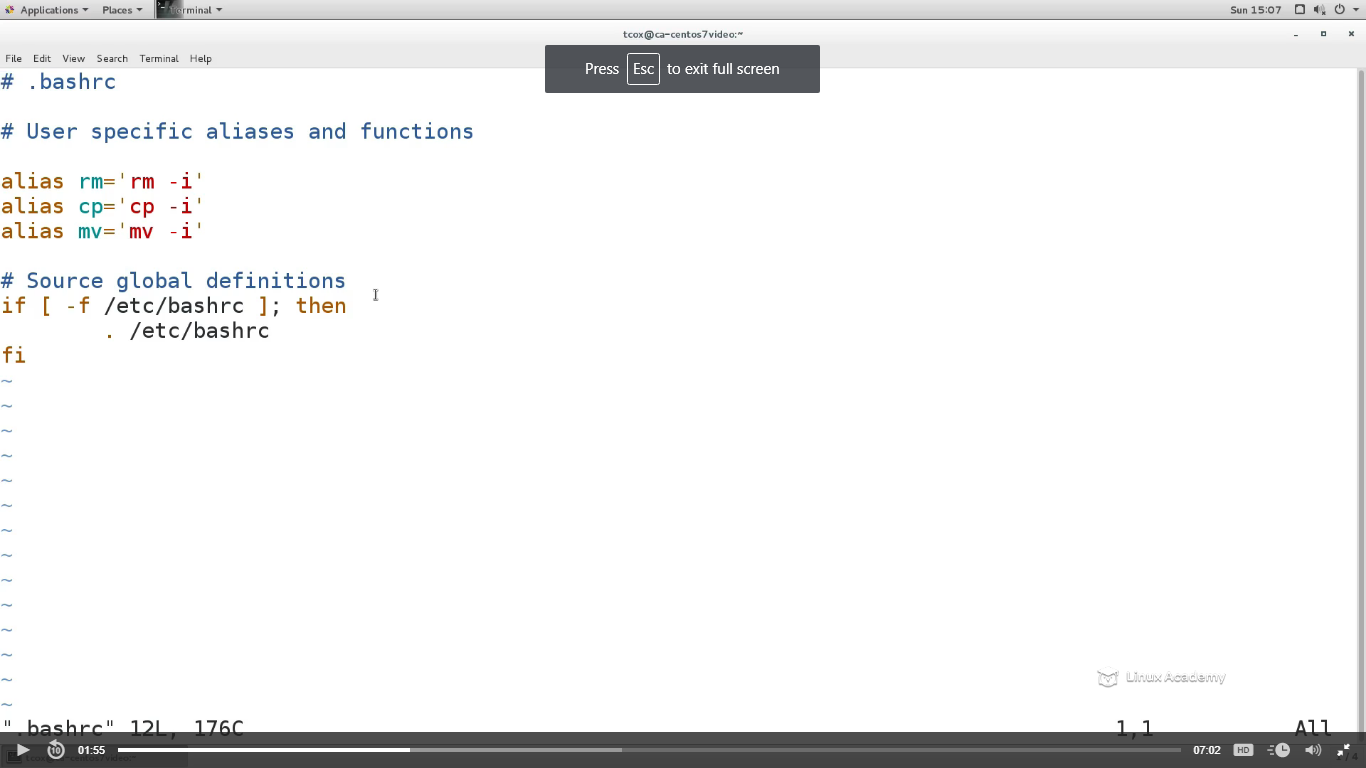
**>**vi .bash\_profile



User specific environment and startup programs

**Bash files - .bashrc:**

>vi .bashrc



User specific aliases and functions

Source global definitions

>env | grep NEWVAL #all variables and paths

>bash

>env | grep NEWVAL

**Bash files - .bash\_history:**

The history of what you typed in the terminal

It will capture the info

>env | grp HISTCONTROL

>export HISTCONTROL-$HISTCONTROL:ignorespace

>env | grep HISTCONTROL

**Bash files - .bash\_logout**

#~/.bash\_logout

The last thing that happen when your logout

>env | grep JAVA

>bash

>env | grep JAVA

**What makes a file a shell script?**

>vi test.sh

Echo “hello world”

>chmod u+x tets.sh

>test.sh

Shell script: executes the s

#!/bin/bash

!:shebang

/bin/bash – shell

Clear

>cp test.sh testm

>testm

It will execute

The .sh is visually to identify , not mandory to put .sh

>which bash

/bin/bash

**Displaying environment variables in a script:**