Linux Introduction

**Linux** is an Open-Source Operating System Kernel under GNU GPL built to handle networking & provide system tools & utilities.

A **Distribution** is a packaged version of Linux developed by a group/company including the Kernel & complementary tools & applications (i.e. – *Amazon Linux 2, Red Hat Enterprise Linux, Debian & Ubuntu*).

|  |  |
| --- | --- |
| **Linux Key Components** | |
| **Kernel** | Core Functionality of the Operating System, used to manage Processor Time Scheduling, Memory Application & Peripheral Decide Access. |
| **Daemons** | User & System Support. Deals with background processes to aid the user in their tasks (*usually system services*). |
| **Applications** | Programs (*Text Editors, Web Browsers, Databases*). |
| **Data Files** | User Content (*Text/Music/Image Files*). Usually [directoryName]filename[.extension]. |
| **Config Files** | Linux Settings, Services & Applications stored in text files. Usual extensions include .cnf/.conf/.cfg/.cf/.ini |

|  |  |
| --- | --- |
| **Command Line Interface (CLI)** | **Graphical User Interface (GUI)** |
| * Consumes fewer Hardware Resources. * Can be Automated with Scripts. * Provides more Options. * Most Linux Servers only use this. | * Visual & Intuitive to Navigate. * Like Windows & MacOS. * Used by most User Workstations. |

|  |  |
| --- | --- |
| **Linux Shell Types** | |
| **SH** | Original Bourne Unix Shell |
| **BASH** | Bourne-Again Default Linux Shell |
| **KSH** | KornShell (Common Unix Shell) |

|  |  |
| --- | --- |
| **Major Linux Distributions** | |
| **Fedora** | Mainly sponsored by Red Hat (IBM), used to develop, test & mature the Linux OS. |
| **Debian** | Adheres to Free Software Principles of Open Source, delivers the Ubuntu Distro, maintained by Canonical Ltd. |
| **OpenSUSE** | Sponsored by SUSE, works as the basis for their commercially-supported SUSE Enterprise Linux. |
| **Amazon Linux 2** | Derived from Fedora, Latest OS the AWS has, provides a stable, secure & high-performing runtime environment for apps on EC2’s latest instance. Enhances security by auto-applying critical/important security updates on booting. |
| **CentOS** | Derived from Fedora, *Community Enterprise Operating System,* enterprise-class Linux distro derived from Red Hat. Replaced by **CentOS Stream** December 2020. |

Linux Command Line

PowerShell Commands: <https://www.pdq.com/powershell/>

Linux Commands: <https://phoenixnap.com/kb/linux-commands-cheat-sheet>

**The Usual Linux Workflow**

1. Open the Default Linux Shell (Bash).
2. The Username is checked against the */etc/passwd* file to ensure its correct.
3. The Password is checked against the */etc/shadow* file to ensure its correct.
4. Press Tab Twice to see all available options!
5. Press CTRL+R to Reverse History Search!

The **Command** states what you want Linux to do, **Option** modifies the command, and the **Argument** defines what the command acts on. *There’s an example of these below*:

* Man (Command) shows the man page,
* -i (Option) performs a case-insensitive search
* whoami (Argument) is what’s being searched for.

|  |  |
| --- | --- |
| **Linux Frequently-Used Commands** | |
| **id** | Print Real & Effective User/Group IDs. |
| **hostname** | Displays the TCP/IP Hostname. |
| **uptime** | How long systems been up since last reboot. |
| **date** | Displays Current Time/Set System Date. |
| **cal** | Displays a Calendar on the Current Month. |
| **clear** | Clears the Terminal Screen. |
| **echo ‘Hi!’** | Outputs the string: Hi! To the Terminal Screen. |
| **history** | Views the History of each User’s Commands in a file at the Home Directory (/home/username/.bash\_history). |
| **touch file\_1)** | Create/Change/Modify Timestamps on Existing Files OR Create a New Empty file\_1 in Directory, can create multiple at a time. |
| **cat** | Shows Contents of Files. |
| **head -n 5 /etc/passwd** | Displays the first 5 lines of the passwd file. |
| **tail -n 5 /etc/passwd** | Displays the last 5 Lines of the passwd file. |
| **grep [options] pattern [files]** | Searches for a string, returning all lines with that pattern. |
| **ls** | List Names & Features of Files & Directories. |
| **| (pipe)** | Used to combine Commands. |
| **| more** | View File Contents that don’t fit on screen, can scroll only down. |
| **| less** | View File Contents that don’t fit, but can scroll up & down. |
| **pwd** | Show your Absolute Path to your Current Location. |

|  |  |
| --- | --- |
| **Linux Standards & Values** | |
| **0 - Standard Input (-stdin)** | The Devices which normally receive Inputs. |
| **1 – Standard Output (-stdout)** | The Devices which normally receive Outputs. |
| **2 – Standard Error (-stderr)** | Where a Process writes an Error Message. |

Linux Users & Groups

Practice at <https://bellard.org/jslinux/vm.html?cpu=riscv64&url=fedora33-riscv.cfg&mem=256>

**User Accounts** represent Users on a System. Their information can be locally stored (*in the /etc/passwd file shown below*) on a server accessible through a network. *They shouldn’t share accounts and it’s best only one user is assigned per account.*

*A diagram of a program

Description automatically generated with medium confidence*

**User Groups** are sets of accounts; they provide a convenient way to associate User Accounts with Similar Security Needs as it saves time granting them all at once. *The groups are stored in the /etc/group file shown below.*

*A group of numbers and arrows

Description automatically generated with medium confidence*

It’s possible to delegate specific commands to specific users through adding them to /etc/sudoers (*the syntax is users hosts=(user:group*)). Use the visudo command to edit sudoers.

A diagram of a number of words

Description automatically generated with medium confidence

|  |  |
| --- | --- |
| **Root User** | **Standard User** |
| * Access/Change any file. * Control Services. * Manage any Account/Hardware/Kernel/Software. | * Access files once given permission. * Control files the user owns. * Limited System Management Access. |

*For the Best Security Practice,* ***Administrators*** *should log in as* ***standard users*** *& elevate their permissions when needed to avoid deleting important files.*

**Su Commands** activate fill administrative permissions used when needed after prompted for the root password, while **Sudo Commands** activate only delegated permissions for a specific standard user after prompted for their own password.

* Sudo Permissions are logged at /var/log/messages (*tail /var/log/messages*).

|  |  |
| --- | --- |
| **Linux User & Group Commands** | |
| **useradd [user]** | Creates a new user called “user”. |
| **useradd -c** | Adds a Comment, *useradd -c “new employee” jdoe.* |
| **useradd -e** | Adds an Expiry Date, *useradd -e 2025-01-01 jdoe.* |
| **useradd -d** | Adds a Home Directory Path, *useradd -d /users/jdoe jdoe.* |
| **usermod -c** | Add a Comment to an Existing File, *usermod -c “Mary Major” mmajor.* |
| **usermod -e** | Add an Expiry Date to an Existing File, *usermod -e 2025-01-01 mmajor.* |
| **userdel -r** | Delete a User Account, *userdel -r jdoe.* |
| **passwd [user]** | Sets/Changes a Password for a User. |
| **groupadd** | Creates a new group called “group”, *groupadd group.* |
| **groupmod** | Modifies an Existing Group, *groupmod -n new\_group group.* |
| **groupdel** | Deletes an Existing Group, *groupdel group.* |
| **gpasswd -a** | Add a User to a Group, *gpasswd – a jdoe, idoe marketing.* |
| **gpasswd -d** | Remove a User from a Group, *gpasswd -d jdoe marketing.* |
| **gpasswd -M** | Set the List of Members of a Group, *gpasswd -m –members USER1, USER2.* |
| **gpasswd -A** | Set the list of Administrators for a Group, *gpasswd -A –administrators ADMIN1, ADMIN2.* |
| **su root** | Switches to root with Curent User’s Environment. Can only be run as an administrator, used to login as any user (i.e. – *su student02*) |
| **su - root** | Switches to root with Root’s Environment. |
| **sudo** | Gives temporary elevated positions for current command without needing to know all the passwords, i.e. – *sudo useradd user20 (student01 can now do useradd).* |
| **sudo -lu** | Used to see delegated sudo permissions (*sudo -lu student01).* |
|  |  |

Linux Vim Text Editor

**Vim** is the Default Text Editor for nearly all Linux Distributions and can be customized through defining personalized key mappings & automated sequences. It has no menu buttons, only two different keystroke modes:

* Command Mode: Keystrokes Issue Commands to Vim.
* Insert Mode: Keystrokes Enter Content into the Text File. *(Pess ESC to return to Command Mode)*

**GNU Nano Text Editor** is common in Linux, but not installed on every distribution. Can be installed using “*sudo apt-get install nano*”.

* Use CTRL+G to view the list of shortcuts within Nano.

**GEDIT** is a GUI Text Editor, with Menu Buttons Available. It’s only available if GUI is installed.

|  |  |
| --- | --- |
| **GNU Nano Commands** | |
| **^G** | **Display Help Text.** |
| **^X** | **Close Current File Buffer & Exit.** |
| **^O** | **Write Current File to Disk.** |
| **^W** | **Search for String/Expression.** |
| **^Y** | **Move to Previous Screen.** |
| **^V** | **Move to Next Screen.** |
| **^K** | **Cut Current Line & Store in Cutbuffer (*CUT*).** |
| **^U** | **Uncut from Cutbuffer into Current Line (*PASTE*).** |
| **^C** | **Display Cursor Position.** |
| **^-** | **Go to Line & Column Number.** |
| **^\** | **Replace String/Expression.** |
| **M-W** | **Repeat Last Search.** |
| **M-^/M-6** | **Copy Current Line & Store in Cutbuffer (COPY).** |
| **^E** | **Move to Current Line’s End.** |
| **M-]** | **Move to Matching Bracket.** |
| **M-</M-,** | **Switch to Previous File Buffer.** |
| **M->/M-.** | **Switch to Next File Buffer** |

|  |  |
| --- | --- |
| **Vim Commands** | |
| **x** | Delete Character at the Cursor. |
| **g** | Move Cursor to Bottom of File. |
| **gg** | Move Cursor to Top of File. |
| **xx** | Delete Line at the Cursor |
| **42G** | Move the Cursor to Line 42. |
| **/keyword** | Search the File for the Keyword. |
| **y** | Yank Text (Cut) |
| **p** | Put Text (Paste) |
| **i** | Insert Mode. |
| **zz** | Save Changes & Exit Vim. |
| **U** | Undo the Last Command. |
| **/G** | Global. |
| **O** | Start Insert Mode & Create a Line below the Cursor. |
| **A** | Start Insert Mod & Create a line after the Cursor. |
| **h,j,k,l** | Left, Right, Up & Down. |
| **:w** | Writes File (Save). |
| **:q** | Quit Vim. |
| **:wq** | Writes File & Quits Vim. |
| **:wq!** | Writes File & Force Quits. |
| **:q!** | Quits without Saving Changes. |
| **dd** | Delete the Entire Line. |
| **R** | Insert a New Character to Replace the Existing one. |
| **c$** | Delete everything after the current character and insert new words. |

Linux Filing Systems

The **File System (Most Commonly Ext4)** organizes how files are stored on the Hard Drive*; files are located inside directories*.

|  |  |
| --- | --- |
| **File Directories** | |
| **/** | File System Root. |
| **/boot** | Boot Files & Kernel. |
| **/dev** | Devices. |
| **/etc** | Config Files. |
| **/home** | Standard Users’ Home Directory. |
| **/media** | Removable Media. |
| **/mnt** | Network Drives. |
| **/root** | Root Users’ Home Directory. |
| **/var** | Log Files, Print Spool & Network Services. |

An **Absolute Path** is the complete path from the root of a file, whereas the **Relative Path** is the path from the current Directory.

|  |  |
| --- | --- |
| **Ls Options** | |
| **-l** | Long Format (Shows Permissions). |
| **-h** | File Sizes (Human-Friendly Format). |
| **-a** | Shows all Files (Incl. Hidden). |
| **-R** | List Subdirectories. |
| **-X** | Sort Alphabetically (by File Extn). |
| **-S** | Sort by Modification Time. |
| **-v** | Sort by Version Number. |
| **Less Options & Meanings** | |
| **less** | Allows scrolling up & down and loads faster than more because it doesn’t load every page, only the page you’re viewing. “*less [OPTIONS] filename*” |
| **-N** | Show Line Numbers. |
| **-X** | Display Content after Last Command & Doesn’t Clear when Exit. |
| **+F** | Watches for File Content Changes. |
| **More Options & Meanings** | |
| **more** | Used to load the files entire contents before viewing and can be used in conjunction with other commands. “*more [-options] [-num] [+/pattern] [+linenum] [file\_name]*” |
| **-d** | Displays Navigation Info at Screen Bottom. |
| **-f** | Prevent Line Wrap. |
| **-p** | Clears Screen before Displaying Content |
| **-s** | Squeeze Multiple Blank Lines into One. |
| **Num** | Number of Lines to Display. |
| **/pattern** | String to Find in File. |
| **linenum** | Line Number where Content Starts Displaying. |
| **file\_name** | Name of File to Display Content of. |
| **CP Options & Meanings** | |
| **cp** | Copies files & directories, overwriting existing files with the same name by default “*cp <file-name> <destination>”* |
| **cp -a** | Archive Files. |
| **cp -f** | Force Copy Overwriting Dest File if needed. |
| **cp -i** | Interactive (Ask before Overwrite). |
| **cp -l** | Link Files instead of Copy. |
| **cp -L** | Follow Symbolic Links. |
| **cp -n** | No File Overwrite. |
| **cp -R** | Recursive Copy (Incl. Hidden Files). |
| **cp -u** | Update, Copy when Source newer than Dest. |
| **cp -v** | Verbose (Print Info Messages). |
| **RM Options & Meanings** | |
| **rm** | deletes files, asking for confirmation if the file is protected. Many can be deleted at once, and a directory can be removed using *rm -rf dir*. “*rm [OPTIONS] filename(s)*”. |
| **-d** | Removes Directory (must be empty). |
| **-r** | Removes Non-empty Directory. |
| **-f** | Never Prompt user (useful when many files). |
| **-i** | Prompts User per File Deleted (when multiple). |
| **-v** | Display Names of Deleted Files. |
| **MKDIR Options & Meanings** | |
| **mkdir** | Creates new directories. “*mkdir [OPTIONS] filename(s)”* |
| **-m <mask>** | Set Permissions to Directory. |
| **-p** | Create Parent Directory. |
| **Hash Command Options** | |
| **hash** | used to view recently-run programs, their locations & number of times run. This information is stored in the hash table to display, recent or delete the tables contents. “*hash [-lr] [-p pathname] [-dt] [commandName …]*” |
| **-d** | Deletes the location of commandName from Hash Table. |
| **-l** | Displays Output in a Format that can be used as an Input for another command. |
| **-p** | Sets pathname as Full Path Location for commandName. |
| **-r** | Empty Hash Table. |
| **-t** | Display commandName Location |
| **Cksum Meanings** | |
| **cksum** | generates a checksum value for a file/datastream to see whether a file was corrupted during transfer, usually in the format “*cksum [FileName]*” |
| **Find Command Options** | |
| **find** | command searches a directory for an owner/filename/fileSize/file modification date. “*find [directory to start from][options][what to find]*” |
| **-name** | Search by File Name. -name <file name> |
| **-iname** | Search by File Name ignoring cases. -*iname <file name>* |
| **-user** | Search by File Owner. *-user <username>* |
| **-type** | Search by File Type. -*type <file type>* |
| **-fprint** | Write Output to File. |
| **-exec** | Ruen a Command on the Returned File. *-exec commandName* |
| **-delete** | Delete the Returned File. |
| **MV Command Options & Meanings** | |
| **mv** | Moves a file from one directory to another. The format is usually “*mv [OPTIONS] destination*” |
|  |  |
| **Grep Command Options & Meanings** | |
| **grep** | Searches File Contents for a Certain Text Pattern/String & Displays each Occurrence. “*grep <text patter/string> <where to search>*” |
| **-i** | Ignore Case. |
| **-r** | Recursive Searches. |
| **-l** | List Only File Names. |
| **-n** | Display Line Number. |
| **-c** | Count Matching Lines. |
| **--files-with-matches** | Name of Files that have Selected Lines are written to Standard Output. |
| **Diff Command Options & Meanings** | |
| **Diff** | Compare File Line for Line & Shows Differences. “*diff [options] File1 File2*” |
| **Tar Command Options & Meanings** | |
| **tar** | Bundles a Collection of Files into a Single Archive File for Easier Copying/Downloading, called a Tarball. “*tar -cvf [customname].tar file1 file2 file3*” , to unbundle “*tar -xf [customname].tar*” |
| **-x** | Extracts Tarball Contents. |
| **-z** | Compresses Tarball Contents using GZIP Utility. |
| **-f** | Specifies Tarball Name. |
| **-v** | Produces Verbose Output by showing File Names once Tarball Processed. |
| **gzip** | Used to Compress Files including Tarballs. “*gzip [name].tar*” |
| **gzip -d** | Used to Decompress Tarballs. |
| **Zip Commands** | |
| **zip -r** | Used as a Compression Tool. “*zip -r [foldername]*” |
| **unzip** | Used as an Extraction Tool. “*unzip [foldername].zip*” |

**Links** are an alternate way to refer to files, and each file has an **Inode** Object to uniquely identify its data location & attributes. There at two link types defined below:

* Hard Link: Points to a File’s Inode, can’t reference a directory and the data will exist once deleted until this is also deleted. “*ln [options] [originalFileName] [linkName]*”
* Symbolic Link: Points to the Original File Name/Hard Link, deleted along with the file. “*ln -s [options] [originalFileName] [linkName]*”

Linux Managing File Permissions

1. **Read (r)** gives users the permission to Open & Read a File.
2. **Write (w)** gives the user Permission to Add, Remove and Rename Files and their Contents.
3. **Execute (x)** gives the user permission to run a program. If users don’t have this permission, they can still view the file, but not run it.

There are Two **Modes of Configuring Permissions**:

1. **Absolute Mode**: Uses Numbers to Represent File Permissions (Most Commonly Used). i.e. – *chmod 764 file\_1.*
2. **Symbolic Mode**: Uses a Combination of Letters & Symbols to Add/Remove Permissions. i.e. *– chmod g+w file\_1.*

There are also different Forms of Identification for those who have **Ownership**:

* A black and white text with arrows

  Description automatically generatedUser: Can Create Files/Directories with their ownership set to the User’s ID. *The owner would be c\_salazar.*
* Group: Contains Multiple Users with the Same Group Permissions, Files/Directories created are associated with one group name. *The Group would be reStart\_stu.*
* Other: The User didn’t create the file & doesn’t belong to the group that might have. *The Other permissions are read, shown at r--.*

.

The **Chown** command can be used to change file ownership. The format is usually “*chown new-owner:new-group file1*”, if there’s only either the owner or a group then only use one.

The **Chmod** command is used to change set permissions on files/directories. The format is usually “*chmod [permissions] file1.txt*”, it has its own syntax which is found below.

* The absolute values are added in the format “chmod (users,group,others)” where you add numbers based on commands, i.e. – 666 would be read and write permissions for all groups (4+2).

|  |  |  |
| --- | --- | --- |
| U (User/Owner) | R (Read) | + Grants a Permission |
| G (Group) | W (Write) | “-“ Removes a Permission |
| O (Other) | X (Execute) | = Removes & Sets New One |

A diagram of a network

Description automatically generated with medium confidence

|  |  |
| --- | --- |
| **Permission Values** | |
| Read | 4 |
| Write | 2 |
| Execute | 1 |
| All | 7 |

*So 654 would be -rw-r-xr—which is (rw-) then (r-x) then (r--)*

Linux Managing Commands

**Delimiters (Separators)** such as quotation marks are generally used within Linux Bash.

* For them to be recognised in a comment command, you need speech marks or else they’ll be taken as parameters. i.e. – *usermod -c “This is a dev user” jdoe .*

|  |  |
| --- | --- |
| **Bash Metacharacters** | |
| **\* (star)** | Any Number of Any Character (Wildcard). i.e. – *cp \* .txt documents/ copies all text files from the current directory to documents.* |
| **?(hook)** | Any One Character (Wildcard). i.e. – *ls be???.py will show you bench.py, you use it as a substitute when you don’t remember.* |
| **[]** | Matches Any Character Between the Brackets . i.e. – *ls file[a-z].txt will list all files labelled filea to filez.* |
| **‘cmd’/$cmd** | Command Substitution (Backticks not Apostrophes). i.e. – *cd ~/Dpcuments is equivalent to cd /home/ec2-user/Documents* |
| **;** | Chains Commands Together. |
| **~** | Represents User Home Directory. |
| **-** | Represents Previous Working Directory. |
| **>** | Sends Command Output to a File. |
| **<** | Receives Input for a Command to a file. |
| **|** | Runs Command redirecting its Output as Input for Next Command. |
| **>>** | Appends Command Output to Existing Contents of a File. |
| **2>** | Redirects Errors Generated by a Command to a File. |
| **2>>** | Appends Errors Generated by a Command to Existing Contents of a File. |

The **Noclober** variable prevents redirected output from overwriting an existing file without warning. i.e.- *set -o noclobber | echo “test1” > textfile.txt | echo “test2” > textfile.txt* will say textfile can’’t be overwritten.

|  |  |
| --- | --- |
| **Text Manipulation** | |
| **sed** | A Non-Interactive Text Editor used to Insert/Delete/Search/Replace. i.e. – *sed ‘s/page/website/5’* replaces the fifth occurrence with website in the file (g replaces all). |
| **tee** | Reads Standard Input, Outputting as Text and writing to a file. i.e. – *hostname | tee file1.txt* writes the hostname to the file & prints it. |
| **sort** | Sorts File Contents into Specific Order. |
| **-r** | Sort into Reverse Alphabetical Order. |
| **-u** | Removes Duplicate Entries. |
| **-m** | Outputs Lines in Month Order. |
| **-n** | Sorts into Numerical Order. |
| **awk** | Write Small Programs to Transform Data, in the format “*awk option ‘program’ input-file*”. |
| **-F fs** | Specifies a Field Separator. |
| **-f source-file** | Specifies File containing Awk Script. |
| **-v var=value** | Declares a Variable. |

Linux Managing Processes

**System/Application Programs** are a series of instructions that tell which actions the computer should take.

* System Programs are the Core-Functionality of the Computer & are Always Present.
* Application Programs are used by End-Users to tell the Computer to do something.

**Processes** are Running Programs identified with a Process ID Number (PID) viewed with multiple commands such as ps (shown in a list), pstree (shown in a tree). *There are running, sleeping and zombie states.* The stages of processing are shown below:

1. Start: The Process is Created
2. Ready: The Process is awaiting Assigned Processor Time.
3. Running: The Process is in Progress.
4. Waiting: The Process is awaiting an Event.
5. Stopped: The Process has Finished Running.

There are also Child Processes/Sub Processes that are spawned when services/applications are complex and require more than one process for more functionality.

|  |  |
| --- | --- |
| **Managing Process Commands** | |
| **ps** | Displays Current Operating System Processes, such as PID/Terminal Type (TTY)/Time/CMD. |
| **ps -e** | Display All Processes. |
| **ps -b** | Use Batch Mode. |
| **ps -fp <number>** | List Processes by PID. |
| **ps -a** | Display All Processes not associated with Terminal. |
| **ps -r** | Show All Running Process. |
| **pidof** | Displays PID of Running Program. *pidof [options] programName* |
| **pidof -s** | Returns One PID. |
| **pidof -c** | Returns PIDs in the Same Root Directory. |
| **pidof -w** | Returns Processes that don’t show a Command Line. |
| **pidof -s** | Works as Separator used between PIDs. |
| **pstree** | Displays the Current Running Processes in Tree Format. |
| **pstee -a** | Shows Arguments in Command Line within Output. |
| **pstee -p** | Shows PID in Decimal Numbers outside Parentheses. *pstree [options] [pid, user]* |
| **pstee -c** | Expands Identical Subtrees. |
| **pstee -n** | Sort by Same Parent PID instead of Process Name. |
| **top** | Displays Real-Time Summary of System Performance & Utilization, Lists Processes/Threads Active in System. Can be Running/Sleeping/Stopped/Zombie. *top [options]* |
| **top -h/top -v** | Display Usage & Version Information. |
| **top -b** | Starts top in Batch Mode. |
| **us value** | Time Spent Running Processes in User Space. |
| **sy value** | Time Spent Running Kernel Space Processes. |
| **id value** | Idle Time. |
| **wa value** | Time waiting for I/O to Complete |
| **hi value** | Time Handling Hardware Problems. |
| **si value** | Time Handling Software Problems. |
| **st value** | Time Lost waiting for other CPU Processes to End. |
| **kill** | Explicitly Ends Processes (usually when it can’t quit on its own). *kill [options] processID* |
| **kill -9 SIGKILL** | Immediately Stops without Graceful Exit. |
| **kill -15 SIGTERM** | Exits without immediately terminating. |
| **kill -19 SIGSTOP** | Pauses Process & Can Use Command Line. |
| **nice** | Launches New Process at Given Priority (-20 highest/19 lowest). *nice [options] [cmd]* |
| **renice** | Adjusts Priority of Already Running Process. *renise [options] [cmd]* |
| **jobs** | Processes that Users Start/Manage, Identified by Job Number. |
| **jobs -bg [process number/name]** | Run a Job in the Background. |
| **jobs -fg [process number/name]** | Run a Job in the Foreground. |
| **at** | Runs a Task Once at Specified Time. *at [options]* |
| **at -l** | Display a Scheduled Job. |
| **atrm** | To Delete a Scheduled Job. |
| **cron** | Runs Task on Regular Basis at Specified Times, maintained in a crontab file. *cron [options]* |
| **cron -a fileName** | Create Crontab File to Hold Commands for Cron Daemon to Run. |
| **cron -e** | Edits the Crontab File as Root User. |
| **cron -l** | View a list. |

**Linux Managing Services**

The **systemctl** command uses the format “*systemctl <subcommand> <service name>*” and has many subcommands; *status, start, stop, restart, enable & disable.*

|  |  |
| --- | --- |
| **System Performance Information** | |
| **lscpu** | List CPU Information. |
| **lshw** | List Hardware. |
| **du** | Check File & Directory Sizes. |
| **df** | Display Disk Size & Free Space. |
| **fdisk** | List/Modify Hard Drive Partitions. |
| **vmstat** | Indicate Virtual Memory Use. |
| **free** | Indicate Physical Memory Use. |
| **top** | Display System Processes & Resource Usage. |
| **uptime** | Indicate System Uptime, Amount of Users & CPU Wait Time. |

**Linux Bash Shell**

A **Shell** accepts & interprets commands within an environment which runs commands, programs & shell scripts.

A **Bourne Again Shell (BASH)** is the default Linux shell which offers an efficient environment for interacting with operating systems & scripting.

There is a **Bash RC** file in each user’s home directory, used to Store User-Specific Configurations.

**Shell Variables** are used to store values within a *symbolic labe*l of a letters/number/special character(\_) and are usually *CAPITALIZED, the value stored is usually treated as string by default*. The format is *$* *name=value*.

* Display Variables using *$ echo $(VARIABLE\_NAME)*
* Make the variable name specific so you know what it is!

**Environment Variables** are structurally identical to shell variables, but are system wide, capable of passing information about the current OS to a running program, applications & daemons. The format is *$ KEY=VALUE*

|  |  |
| --- | --- |
| **Environmental Variables** | |
| **$HOME** | Defines User’s Home Directory. |
| **$PATH** | Indicates Command’s Location. |
| **$SHELL** | Defines Login Shell Type. |
| **$USER** | Contains User’s System Name. |

The **Alias** command can substitute a long command for a short one, usually in the format “*$ alias alias\_name=’command’*”. You can also **Unalias** with the same format.

Add an Alias to .bashrc using *$ nano ~/.bashrc*

**Bash Shell Scripting**

**Scripts** are Text Files of Commands/Related Data which are run after being processed acting on **Arguments**. They are consistent due to the automation removing human/manual error. There is a process shown below:

1. Create the script with a Text Editor (The First Line of a Script must always be *#!bin/bash with a shebang at the star.*
2. Set Script Permissions to run. *(# within the editor to label its purpose*)
3. Use *./[name].sh* to Run the Script.

|  |  |
| --- | --- |
| **Frequently-Used Commands** | |
| **echo** | Displays Information on the Console. |
| **read** | Read User’s Input. |
| **subStr** | Gets Substring of a String. |
| **+** | Add Two Numbers/Strings. |
| **-** | Subtracts Two Numbers. |
| **file** | Opens a File. |
| **mkdir** | Creates a Directory. |
| **cp** | Copies Files. |
| **mv** | Moves Files. |
| **chmod** | Set File Permissions. |
| **rm** | Deletes Files/Folders. |
| **ls** | Lists Directories. |
|  |  |

|  |  |
| --- | --- |
| **Conditional Arithmetic** | |
| **-eq** | Is Equal to. |
| **-ne** | Isn’t Equal to. |
| **-gt/>** | Greater Than. |
| **-ge/>=** | Greater Than/Equal to. |
| **-lt /<** | Less Than |
| **-le/<=** | Less Than/Equal to. |

* The **TEST** command is used to check File Types & Compare Values, exiting 0 for true & 1 for false. The format is *test <expression*>.
* The **BREAK** command stops the entire loop.
* The **EXIT** command stops the script & exits to the shell.
* The **CONTINUE** command begins the entire loop anew.

There are **IF Statements** which Run Subsequent Commands once an Initial Command Succeeds and must end in fi. The format is usually *if <condition>;then <command>;fi*

* **ELSE Statements** are in the format *if <condition>;then <command>;else <other command>; fi*
* **ELIF statements** add an alternate condition with the format *if <condition>; then <command>; elif <condition>; then <command>; else <command>; fi*

There are also **LOOP Statements**.

* **FOR Statements** loop a command a specific number of times, the format is *for <number>; do <command>; done*
* **WHILE Statements** continue as long as a condition is true, the format is *while <condition>; do <command; done*
* **UNTIL Statements** run code until a condition is true, the format is *until <condition>; do <command>; done*

**Linux Software Management**

Linux Distributions have Different **Managing Approaches**, *Red-Hat uses Red Hat Package Manager (RPM), Debian uses DPKG & From Source Code uses the GNU Compiler Collection.*

**Repositories** are Servers containing Software Packages online at a vendors site, on an internal administrator-managed server or on the systems local hard-disk.

The **YUM package manager** can be used on Linux to install packages and has the format $ yum [options] [command] [list of package names].

* To Update: *yum update <package>*
* To Check Installed: *yum list installed*
* To Uninstall: *yum remove <package>*

|  |  |
| --- | --- |
| **WGET** | **CURL** |
| Recursive Download. | Single Resource Download Only. |
| Supports HTTP, HTTPS & FTP Protocol. | Support Those & More Protocols. |
| Performs Retries over Unreliable Connection. | Runs on more Platforms than WGET. |

**To Download & Install a Program:**

1. Download the Program: *curl/wget <link>*
2. Unzip Installation File: *unzip <file>.zip*
3. Run Installation Program: *sudo ./<link>*

**Managing Log Files**

**Logging** keeps records of events within a system to help with auditing within the /var/log directory. *There are many types of logs including the following:*

* System Logs (Startup & Shut Down Information).
* Event Logs (User Login & Logout Events).
* Application Logs (Startup Time, Actions & Errors).
* Service Logs.

|  |  |  |
| --- | --- | --- |
| **Logging Levels** | | |
| **0** | **EMERGENCY** | Logs Messages when the System becomes Unstable. |
| **1** | **ALERT** | Logs when Immediate Action Needed. |
| **2** | **CRITICAL** | Logs Messages for Critical Errors (System may be Unstable). |
| **3** | **ERROR** | Logs Non-Critical Errors. |
| **4** | **WARN** | Logs Warnings & Serious Messages (Default Level). |
| **5** | **NOTICE** | Logs Normal Event Messages. |
| **6** | **INFO** | Logs Informational Messages. |
| **7** | **DEBUG** | Logs Debug-Level & Info Messages. |

*A close-up of a computer code

Description automatically generated*

*Grep is quite useful for searching log files.*

|  |  |
| --- | --- |
| **Important Log Files** | |
| **/var/log/syslog** | System Information. |
| **/var/log/secure** | Authentication Information for Red Hat Distros. |
| **/var/log/kern** | Linux Kernel Information. |
| **/var/log/boot.log** | Startup Messages. |
| **/var/log/maillog** | Mail Messages. |
| **/var/log/daemon.log** | Stores Background Service Information. |
| **/var/log/auth.log** | Authentication Information for Debian Distros. |
| **/var/log/cron.log** | Cron Messages for Scheduled Tasks. |
| **/var/log/httpd** | Apache Information for Red Hat Distros/ |

The **LASTLOG** Command retrieves User Info from */var/log/lastlog* & outputs it to console.

* To Display ec2-user Information: *lastlog -u ec2-user*
* To Display Login Info more recent than a day ago: *lastlog -t 1*

The **LOGROTATE** Command can be used to compress, rename & clean up log files, it can be activated on a regular basis and maximum amount can be set.