**Init and run levels in Linux**

It is the first process executed by the kernel during the booting of a system. It is a daemon process which runs till the system is shutdown. That is why, it is the parent of all the processes. First of all, **init** reads the script stored in the file **/etc/inittab.** Command **init** reads the initial configuration script which basically take care of everything that a system do at the time of system initialization like setting the clock, initializing the serial port and so on.

By reading this file, **init** determines how the system should be set up in each runlevel and sets default run level.

After determining default runlevel for the system, **init** starts all background processes required to run the system. First it runs each of the kill script (their file name starts with a K) with a stop parameter. Then it runs all start scripts (their file name starts with an S) to start all services and applications.

Runlevels

A runlevel is a software configuration of Linux system which permits only a selected group of processes to exist. It defines what services are operating on the system.

Runlevels are identified by numbers. **init** can be in one of eight runlevels. It is changed by a privileged user run **telinit,** which sends appropriate signals to **init** to change runlevel.

|  |  |
| --- | --- |
| **Runlevel** | **Function** |
| 0 | Halt the system |
| 1 | Single user mode |
| 2 | Multiuser mode without networking |
| 3 | Multiuser mode with networking |
| 4 | Not used |
| 5 | Multiuser with networking and X windows |
| 6 | Reboot the system |
| S/s | Not used directly |

* 0,1 and 6 are reserved runlevels.
* Runlevel S or s are same.
* 7-9 are also valid runlevels, though they are not documented as traditional Unix variants, do'nt use them. But they are same as runlevels S or s. They are aliased.

Changing runlevels

There are many ways to change runlevels. To make the changes permanently, change the default level in the file **/etc/inittab.**

After all the processes, **init** waits for one of its descendent process to die, for a powerfail signal. Or wait until **telinit** signals to change the runlevel. When above conditions occur, it re-examines the **/etc/inittab** file. New entries can be added to this file any time.

When **init** is not in single user mode and receives a power fail signal **(SIGPWR),** it reads the file **/etc/powerstatus.** Based on the content of this file, **init** starts a command:

* **F** (FAIL) ? When power fails, UPS provides the power. Execute the powerfail and powerwait entries.
* **O** (OK) ? Power has been restored, execute powerokwait entries.
* **L** (LOW) ? Power is degrading and UPS has a low battery, executes powerfailnow entries.

If file **/etc/powerstatus** doesn't exist or contains anything other than **F, O, L,** then **init** behaves as if it has read the letter **F.**

To interact with **init, /dev/initctl** control channel should be used rather than **SIGPWR** and **/etc/powerstatus.**

When a request to change the runlevel is send to **init,** init sends warning signal **SIGTERM** to all the undefined processes in the new runlevel. It then waits for 5 seconds before terminating processes via **SIGKILL** forcibly.

All these processes should remain in the same process group which was created by the init, otherwise they will not be able to receive these signals and they need to be terminated separately.

Telinit

**/sbin/telinit** is linked up with **/sbin/init.** Telinit takes up an argument and signals **init** to perform the respective function.

Arguments are as follows with are one character each:

|  |  |
| --- | --- |
| **Argument** | **Function** |
| 0,1,2,3,4,5,6 | Switch to specified runlevel |
| a,b,c | Processes only file entries from **/etc/inittab** having runlevel a,b,c. |
| Q or q | Re-examine file **/etc/inittab** |
| S or s | Switch to single user mode |
| U or u | init re-execute itself. No re-examine occurs, runlevel should be from S,s,1,2,3,4,5 otherwise request would be ignored silently. |

Default time to wait between sending the signals SIGTERM and SIGKILL is 5 seconds, but **telinit** can also ask **init** to change this time with the option **-t sec.**

Only users having appropriate privileges can execute **telinit.**

The **init** binary checks by the process ID whether it is **init** or **telinit, init's** process ID is always **1.** so one can also use **init** instead of **telinit** as shortcut.

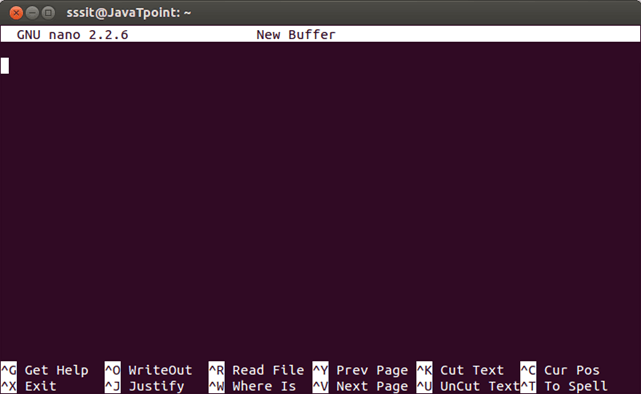
# Nano editor

GNU nano is a friendly and convenient text editor like vi and emac. It offers many other extra features like word searching, replacing, jump to a line or column, filename tab completion, auto-indentation, etc.

Nano is a clone of the pico text editor. Nano is not pre-installed in all distros, but Ubuntu has it.

**Syntax:**

1. nano



Look at the above snapshot, this is the default nano screen on passing command **"nano".**

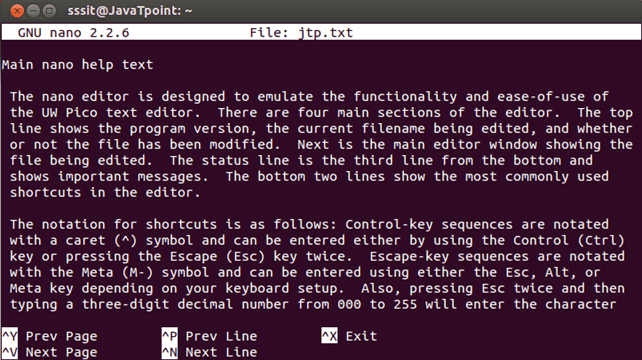
At the top, GNU nano version is shown at the left and in the middle filename is shown being edited (currently no file is being edited hence New Buffer is written).

At the end of the screen, keyboard commands are given. Command written as **^G** means press **ctrl + g** key and command **M-R** means press **alt + r.**

There is no use of uppercase letter in any of the keyboard command in nano editor. You can use lowercase letter with **ctrl** and **alt** keys.

**Note:**For MAC users **escape** key is used instead of **alt** key.

Press **ctrl + g** for the help menu. You'll get all information about nano editor in this menu.



# shred

The shred command allows us to delete our files securely making it very much difficult to recover that file by anyone.

Erasing a file with rm command only erases the file system entry and keeps the content of the file intact. It is quite easy to recover removed files content using some softwares.

To prevent from data recovering, shred **overwrite** the data multiple times by doing maximum destruction of the data.

Shred can be used on files as well as devices such as hard disk partition. By default, shred overwrites file three times with different patterns, but this number can be changed.

On normal deletion of file, that portion is simply marked to be used for another file, but data of that file is still present. Those data, can be retrieved through some advanced softwares.

This command is very much helpful when you want to permanently remove files containing sensitive data. To remove these type of files rm command is not sufficient.

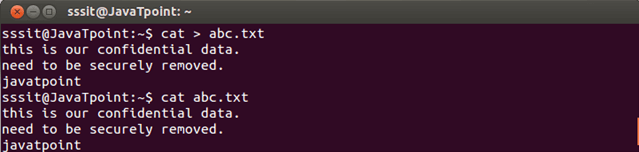
Shred command without argument

**Syntax:**

1. shred **<fileName>**

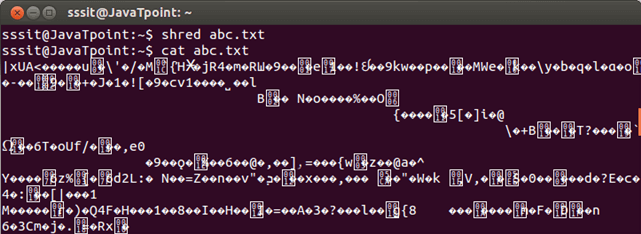
**Example:**

shred abc.txt



Look at the above snapshot, file **abc.txt** is created. It need to be securely deleted as it contains confidential data.

Now, after passing shred command to the above file, it displays following output.



Look at the above snapshot, data of the file is shreded but file still remains there.

To delete file

Use of shred command without any argument only overwrites the file content, but file still remains in the file system.

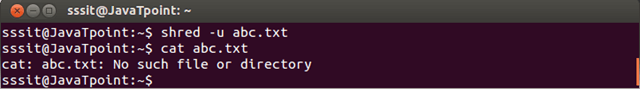
To delete the file from the file system, use option **u.**

**Syntax:**

1. shred -u **<fileName>**

**Example:**

shred -u abc.txt



Look at the above snapshot, file is deleted from the file system.

Verbose information about shred process

If you want to see what shred is doing with your file use option **v.**

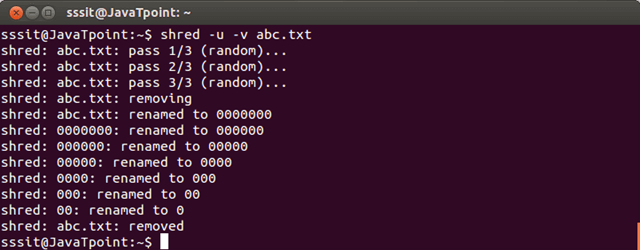
Here, we are using u option to remove the file as well.

**Syntax:**

1. shred -u -v **<fileName>**

**Example:**

shred -u -v abc.txt



Look at the above snapshot, by default three times the file has been overwritten and seven times it is renamed.

Changing overwrite number

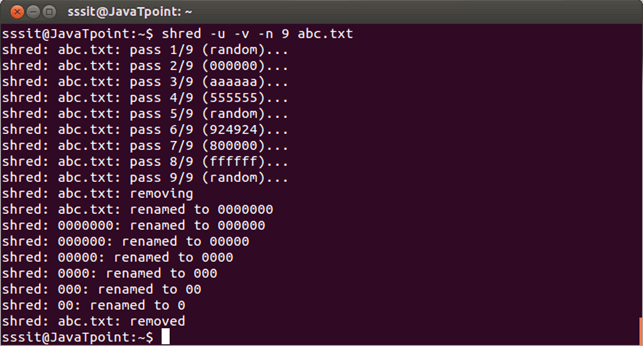
You can also change the overwrite number from default (which is 3) to whatever you want using option **n.**

**Syntax:**

1. shred -u -v -n 9 **<fileName>>**

**Example:**

shred -u -v -n 9 abc.txt



Look at the above snapshot, nine times the file has been overwritten.

Using shred on a drive

The shred command is also used to overwrite the data of a drive. Drive contains a huge amount of data, hence, a lot of time will be required to shred this data.

**Syntax:**

1. shred **<deviceName>**

**Example:**

shred /dev/sda1

# Linux mount

The mount command attaches the filesystem of an external device to the filesystem of a system.

It instructs the operating system that filesystem is ready to use and associate it with a particular point in the system's hierarchy. Mounting will make files, directories and devices available to the users.

It mounts the external storage devices like hard disks, pen drives, USBs etc.

Conversely, **umount** command unmount the mount point and detach the device from the system.

To mount a device generally, following syntax is used

**Syntax:**

1. mount -t type **<device>** **<directory>**

Here, this command instructs kernel to attach filesystem of device at the specified directory.

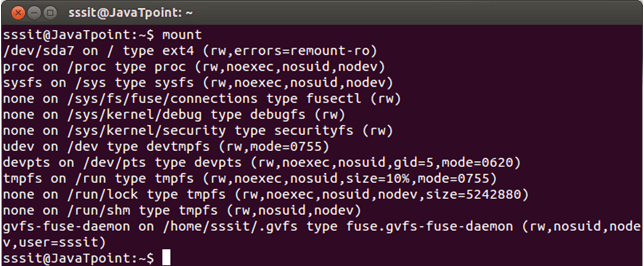
If destination directory is not mentioned, by default, it mounts the device in the /etc/fstab file.

List currently mounted file systems

This command display all currently mounted file system on a system.

**Syntax:**

1. mount



Look at the above snapshot, it displays all currently mounted file systems.

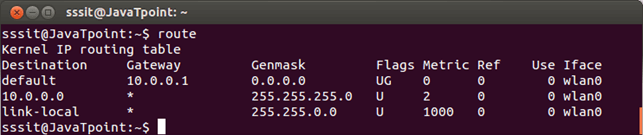
# Linux route

The route command displays and manipulate IP routing table for your system.

A router is a device which is basically used to determine the best way to route packets to a destination.

**Syntax:**

1. route



Look at the above snapshot, it displays all existing routing table entries on our system.

It shows that if the destination is within the network range of 10.0.0.0 to 10.0.0.255, then the gateway is **\*,** which is 0.0.0.0. This is a special address which represents an invalid or non-existent destination.

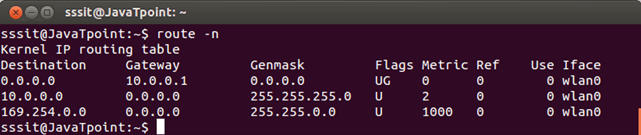
Packets which are not within this IP range, will be forwarded to default gateway, which further routes the packet.

Displaying numerical IP address

This command displays output in full numerical form.

**Syntax:**

1. route -n



Look at the above snapshot, the numerical IP address is displayed.

Adding a default gateway

Packets that are not within the network range are forwarded to the gateway. We can specify this gateway with the following command,

**Syntax:**

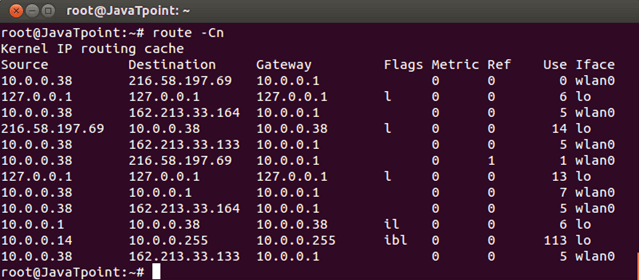
1. route add default gw **<IP** address**>**

Routing cache information

Kernel maintains a routing cache table to route the packets faster. To list this information, use following command,

**Syntax:**

1. route -Cn



# Linux Man Command

The "man" is a short term for manual page. In unix like operating systems such as linux, man is an interface to view the system's reference manual.

A user can request to display a man page by simply typing man followed by a space and then argument. Here its argument can be a command, utility or function. A manual page associated with each of these arguments is displayed.

If you will provide a section number in the command, then man will be directed to look into that section number of the manual and that section page will be displayed. And if not, then by default it will display the first page and you have to go through the entire sections in a pre-defined manner.

We'll read about section number in this tutorial.

**Syntax of man:**

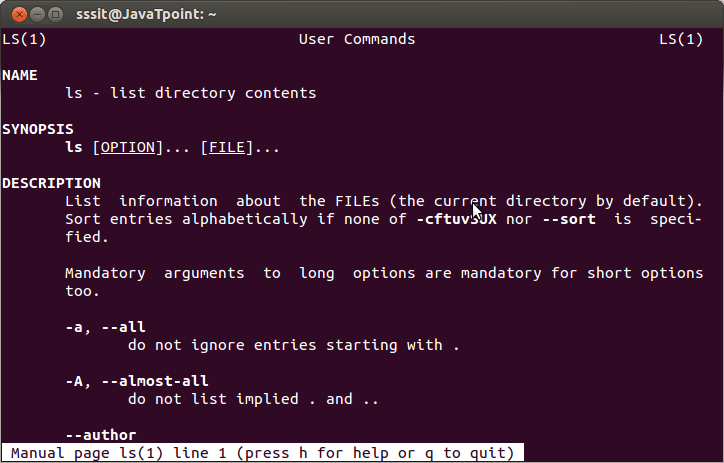
1. man [option(s)] keyword(s)

But generally [option(s)] are not used. Only keyword is written as an argument.

For example,

1. man ls

This command will display all the information about **'ls'** command as shown in the screen shot.



How Output is Displayed in Command Shell

man display its output through pager. A pager is a program that displays its output one screenfull at a time, means whole text doesn?t appear at once and there is no option to scroll down the page.

A colon at the bottom displays end of the on-screen page. To go to the next page you can use ?space bar' or 'f' and to go backward page you can use 'b'.

To exit from the on-screen page use 'q' and you will be directed to the shell program. And for help press 'h'.

Sections in the man Page:

man page is divided into different sections. Each section is divided based up on a particular topic.

The man pages have a number written in the parentheses after the command. These number represent the section number. In the above picture, you can see LS(1) at the top, which shows that it is from section 1.

As I stated earlier, if you will mention a particular section number then your man page will be directed to that section. Or else, you have to go through the whole sections in a pre-defined manner to go to your desired section.

**Sections:**

1. Executable programs and shell commands
2. System calls
3. Library calls
4. Special files
5. File formats and conventions
6. Games
7. Miscellaneous
8. System administration commands
9. Kernel routines

n. Tcl/Tk (a programming language)

**Syntax for a particular section:**

1. man section\_number keyword

For example

1. man 2 passwd

man Options

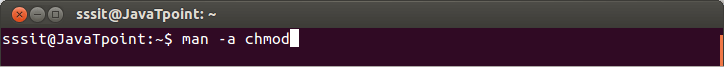
|  |  |
| --- | --- |
| **Commands** | **Function** |
| [man -aw](https://www.javatpoint.com/linux-man-aw) | List all available sections of a command. |
| [man -a](https://www.javatpoint.com/linux-man-a) | To view all man pages of a command. |
| [sman -k (apropos)](https://www.javatpoint.com/linux-man-k) | Shows a list of results in man page containing a keyword match. |
| [-f, whatis](https://www.javatpoint.com/linux-man-f) | It displays description from manual page if available. |
| [whereis](https://www.javatpoint.com/linux-man-whereis) | Used to determine location of a man page |

# Linux man -a

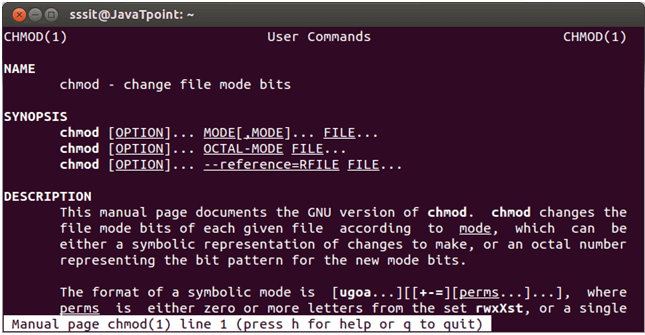
This commnd help us to show all the man page sections of a particular topic. Sections will come in ascending order that is lowest section number will come first.

To go to the next section press 'q' and then enter.

In the below picture, we have given the command **'man -a chmod'**. We know that chmod has two sections (1) and (2).



chmod section 1 man page will appear first as shown below.



To go to next section press q followed by enter key and we'll be directed to the below page of section 2.

