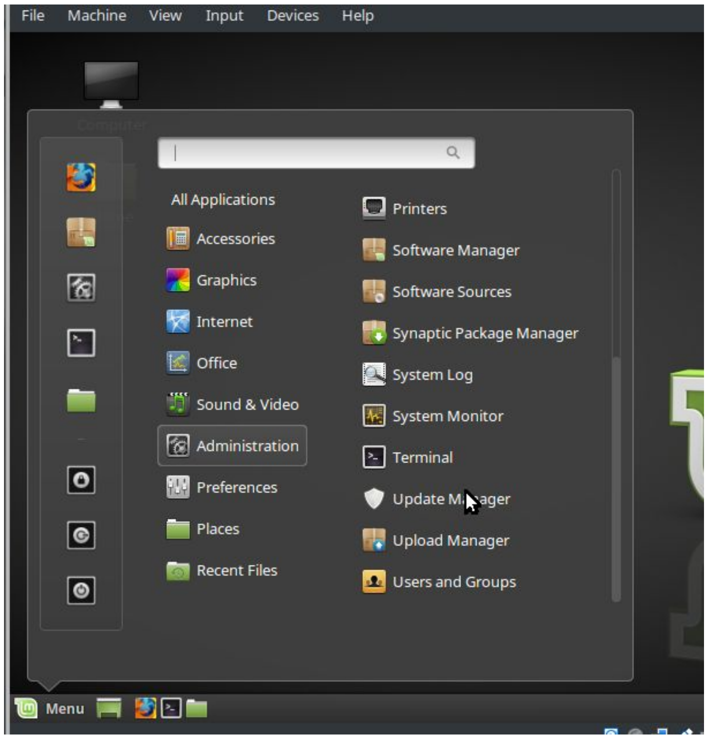
2.1 File Structure

So now that we have a fresh install of Linux Mint, diving into the all mighty terminal is what were are doing next. What is a terminal? Well to put it in simple terms, it's a console that connects the user directly to the operating system via command execution. Essentially, everything you can do with a GUI, you can do it with the terminal. So why learn the Linux Terminal if performing simple tasks like deleting a file can be done with the GUI? because not all systems you encounter will have a graphical user interface(GUI) and it's important to know your way around a system that does not have a UI. This is very crucial if you truly want to understand the Linux OS.

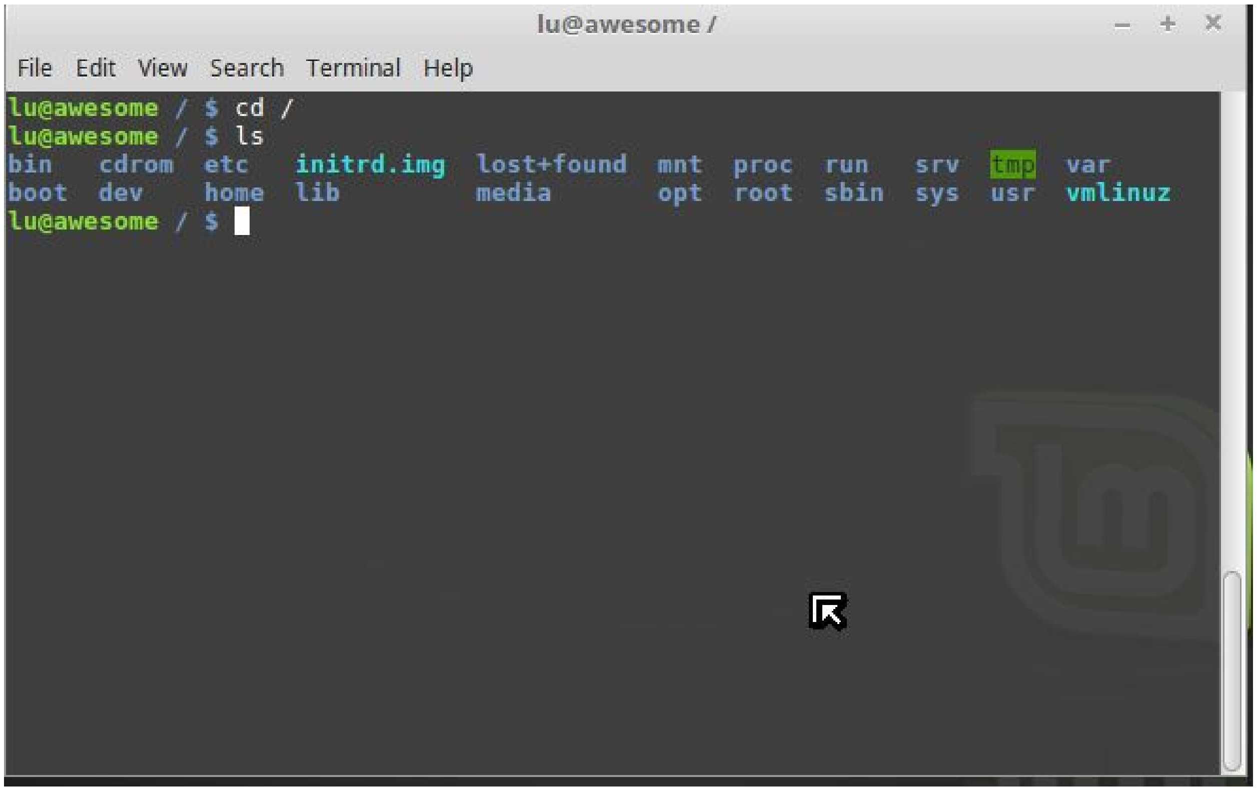
Before we start any kind of terminal work, we really need to know about the directory structure of Linux and the meaning behind these directories. Also we need to know some common shorthand alias' and some very common things you will see in the linux environment.

So we all know that the root directory is represented by '/', its the folders inside of the root directory that we will be focusing on. Go ahead fire up your Linux Mint Virtual machine. On the Desktop.
Click -> Menu -> Administration -> Terminal .
The terminal can also be accessed by the task bar or favorites.



Now in the terminal type
lu@awesome ~/ $ cd /

lu@awesome ~/ $ ls



```
lu@awesome / $ cd /
lu@awesome / $ ls
bin   cdrom  etc   initrd.img  lost+found  mnt  proc  run   srv  tmp  var
boot  dev    home  lib         media       opt  root  sbin  sys  usr  vmlinuz
lu@awesome / $ 
```

What is all this stuff? Well this is the Folder structure!. This is equivalent to microsoft's C:// drive. It is the / root of the operating system. So let's break some of these down. These all have different functions and contain different programs and devices. Such as windows32 or Programs in windows.

- **/** Root directory that is the base of the entire file system. All files and folders will be found in this directory
- **/bin** contains programs that are apart of the core linux operating system, think the very basic commands. ls, cat, tar, cp etc…
- **/boot** contains the compiled linux kernel images that are necessary for the bootloader to boot the system properly. Most common bootloaders are grub2 and lilo.
- **/cdrom** yup you guessed it cd and dvd is mounted here.
- **/dev** contains device files, linux treats everything as a file naturally. In a case for a usb flash drive, the actual location might be /dev/sdb . this translates to the second file system mounted by the linux kernel. If you notice /dev/sda1 is your primary partition /dev/sda2 is your swap etc… but /dev/sda is your hard drive. I will show you how to

manually mount USB flash drives later. Like I said everything you can do with a GUI can be done with the terminal

- **/etc** is primarily used for configuration files. In here you will find the configuration files for mostly all services and programs. EX: apache: /etc/apache2 , bind: /etc/bind9 , ssh: /etc/ssh/sshd_config.
- **/home** this directory is for the user accounts to put their files and folders in. ex **/home/lu** , **/home/AnyOtherUserWithLoginCredentials** .
    - Note : the root home directory is **/root** and not **/home/root**
    - Note: shorthand notation for **/home/user** can be typed as **~/**
- **/lib** standard system libraries and loadable driver modules needed to boot the system. Library files written in the C programming language are used by the kernel to boot properly. (Linux kernel is written in C) These files will have,
    - .a for tradition , static libraries
    - .so and .sa for shared libraries.
- **/lost+found** The lost+found directory is used by file system check tools (fsck). When a system crashes and there is some inconsistency, fsck might be able to (partially) recover lost information (files or directories). It might not know where these files should reside, so it places them in the lost+found directory so that the administrator can move them back to their original location(s). The lost+found directory is not vital. If the administrator decides to delete it, on the next run of fsck will be recreated
- **/media** a folder where file systems are usually mounted. Typically removeable media such as flash drives , CD-ROM drives.. And so on… remember what i said earlier that usually devices are mounted as /dev/sda and /dev/sdb1 ? well this is still true, if you are under a regular user account without root access. The Linux system will mount removable media to /media, so that regular users don't have to go and mount /dev/sdb to a directory themselves, also the users can navigate to the /dev/sdb1 folder but can't really do much, because of permission issues. /media is a symbolic link to /dev/sd*1 devices.
- **/mnt** another folder where devices can be mounted. But this folder is usually for temporary drives. Devices that are not crucial to operations.
- **/opt** installed software that has nothing to do with booting the system. Unessential software that is added the system. “Add-on ‘ software. Usually third party software . Ex: eclipse-studio.
- **/proc** is a very special folder, in a sense it is a virtual file system on its own. This folder is also known as the proc file system. Most of the files in here are Zero Bytes in size. Most of the files in here will display system information. And can be an alias to already system commands found in bin.
    - Ex: the command **lsusb** is actually ‘ **cat /proc/bus/usb** ‘
    - For more info please read https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/Reference_Guide/ch-proc.html

- **/root** root home directory.
- **/run** is meant to have system daemons to start early in the boot process and store runtime files , like PID files. Daemons like *systemd* and *udev.* I can talk all day about /run and all of the crazy stuff that goes on.
    - **Please , just read https://wiki.debian.org/ReleaseGoals/RunDirectory**
- **/sbin** contains all your admin programs, programs that need superuser rights. **mount , umount, shutdown, lsmod, chmod.**
    - Note some of these programs can be ran by regular users, Ex: **host**
- **/srv** contains site-specific data that is served by the system. What the heck does this mean? Well, it has the location for the files to a particular service. This is different that having configuration files located in **/etc**. For example the config files for an apache server might be located in /etc/httpd but the actual server files can be located in **/srv/http**. Same thing for a ftp server.
    - Some distributions might have their apache files in **/var/www/html** . yes i know linux will make you want to bang your beautiful face on your keyboard. So many different locations and config files.
    - **EX: /srv/http**
    - **EX: /srv/ftp**
- **/sys** modern Linux distributions include a /sys directory as a virtual filesystem (sysfs, comparable to /proc, which is a procfs), which stores and allows modification of the devices connected to the system,[18] whereas many traditional UNIX and Unix-like operating systems use /sys as a symbolic link to the kernel source tree. /sys is a ram-based file system. It allows you to obtain information about the components and the system. About hardware. Similar to /proc, its an interface to the kernel that can view information of devices and hardware.
- **/tmp** temporary files are stored in here. Files are usually created by running programs and services.
    - Note if you delete the contents of this folder, your system can crash. Some programs require these files to function properly/
        - Also note these files might be in here.
        - **File locking i**s a mechanism that restricts access to a computer file by allowing only one user or process access at any specific time. For instance, you cannot run two instances of a package manager, it will lock.
        - A **PID** is an acronym for process identification number on a Linux or Unix-like operating system. A PID is automatically assigned to each process when it is created. A process is nothing but running instance of a program and each process has a unique PID on a Unix-like system.
- **/usr** contains a LOT, I mean a LOT. /usr is used for many thins, such as the users binaries, programs, shareable files such as icons, fonts, themes .. /usr also contains libraries and *src* directories that can contain custom built programs and kernels.

    - **/usr/share/icons** contains the global icons for all users, if a user wants to import custom icons they would be put in ~/.icons
    - **/usr/src/linux** common directory for compiling linux kernels.
    - Read http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/usr.html for more info
- **/var** contains 'variable data' such as logs, variables in configuration files such as /var/www/html for apache2 config files.
    - **/var/log/apache2**
    - /**var/www/html** root directory for apache2 via default apache2 config file(this varies for distribution)
    - **/var/cache/genkernel** contains cache data such as dependencies for programs that have been compiled. In this instance a general kernel cache directory can contain the *busybox* utility set.
- **Vmlinuz** vmlinuz is the name of the Linux kernel executable. vmlinuz is a compressed Linux kernel, and it is capable of loading the operating system into memory so that the computer becomes usable and application programs can be run. vmlinuz = Virtual Memory LINUx gZip = Compressed Linux kernel Executable
- **Initrd.img** "initial ramdisk". It is enough files stored in a ramdisk to store needed drivers . You need the drivers so that the kernel can mount / and kick off init.

For more info about the linux file system structure please read
- **https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard**
- **http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/Linux-Filesystem-Hierarchy.html**

2.2 Basic Commands

Now that we have all that crazy stuff burned into the back of our skulls. Let's start off with the very basics, open a terminal and try out the following commands.

Remember Linux is **CASE SENSITIVE:** 'ls' is different from '*LS' or 'Ls' or 'lS'*. These will be recognized by the terminal as different commands. This goes for all files and folders. For instance. If you have a folder named 'thisfolder' and you type in cd 'Thisfolder" the terminal will tell you that 'Thisfolder' does not exist. For windows users it might be a little harder to adjust since windows is not case sensitive and that would be a valid command (replacing cd with dir).

- **ls:** list the current contents of the folder
- **pwd:** list the path of the current folder
- **cd:** change directory
- **cp**: copy command: files & folders
- **mv**: move files and folders to a different location

- **rm**: remove files and folders
- **touch**: create a file
- **mkdir:** create a directory
- **Dmesg**: displays the kernel log
- **Init 6**: LOL XD
- **runlevel :** displays current runlevel of linux system.
- **Telinit (param) telinit 6** : will switch system run level and tell the shutdown
- **Shutdown (param) (param) shutdown** -h 5 “System going down in 5 min
- **Cal** : calendar program
- **Free**: shows how much ram there is
- **Df**: shows disk and file system information

Next , let's try some of these commands with what we call in the linux world ‘options’ and ‘arguments’. Options are usually a dash ‘ - ‘ or a ‘--’ and are followed by a keyword or letter. Note: not all commands follow this pattern. Depending on the options and arguments, commands can do different things.

For example try this next command, try removing a FOLDER with the rm command.

```
lu@awesome ~/ $ rm ThisFolder
rm: cannot remove 'ThisFolder/': Is a directory
```

Linux will now complain to you and tell you how crazy you are for trying to delete a directory with the rm command. But why? Legend says this was built in the command so that users would not accidently delete entire directory structures with this command by accident. But that's just how the legend goes. So we have to use some options to get rid of this folder, because Linux should do everything we tell it to do.

Let's give it the *-r* option. This will remove directories and all of its contents recursively.

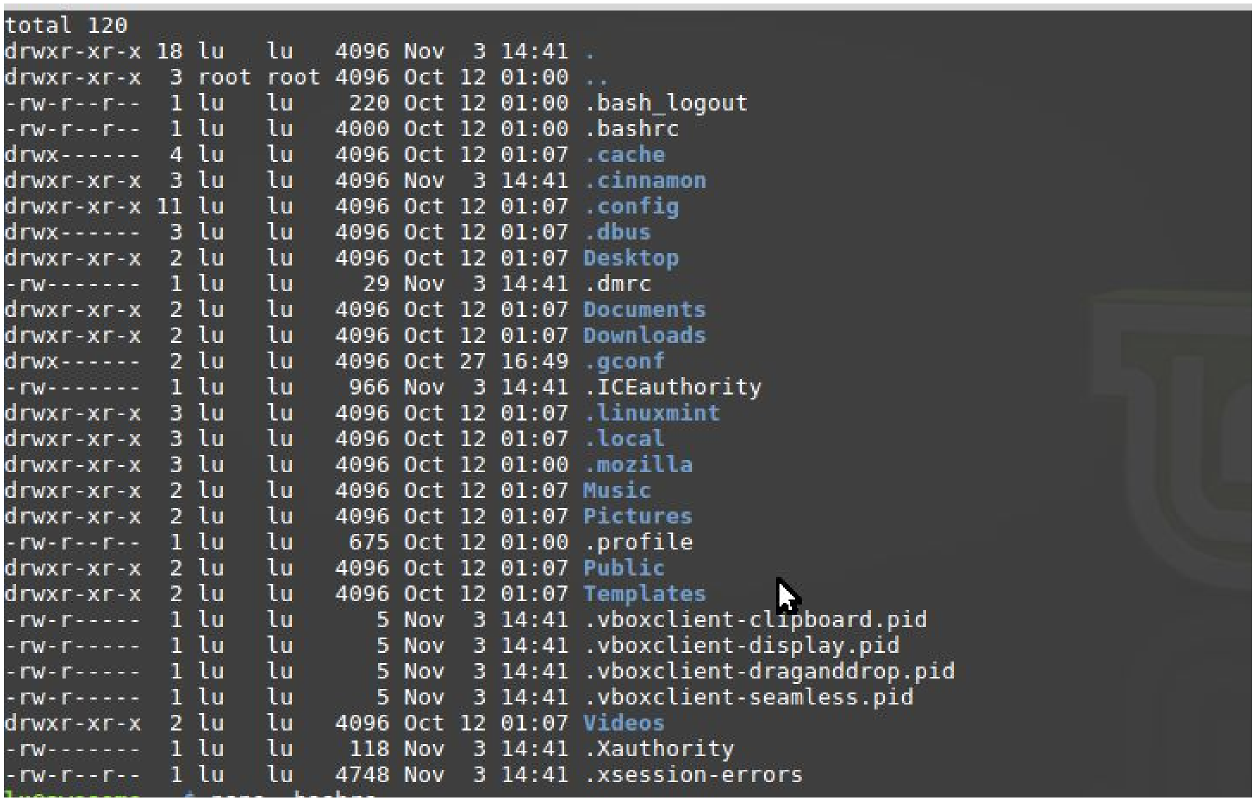
```
lu@awesome ~/ $ rm -r ThisFolder
```

There! That should do it. Lets break this down in the sense of what it actually does.

- rm : command to remove files/folders
- -r : *Recursive* option to the rm command that will remove a directory and its contents
- *ThisFolder* : this is the first argument in the command, we can have many arguments after a command, or even more options & commands after our first initial argument.

So the next concept i'm going to talk about is an **Alias,** In Linux an alias is a custom command. And they can be found inside the **~/.bashrc or ~/.bash_profile** files. So what exactly are these two files and what are the differences? First let's make an alias . navigate to the home directory and execute this command. This command will change the directory to your home and list (**-l**) all(**a**) the contents, including hidden files.

```
lu@awesome/ $ cd ~/ && ls -la
```



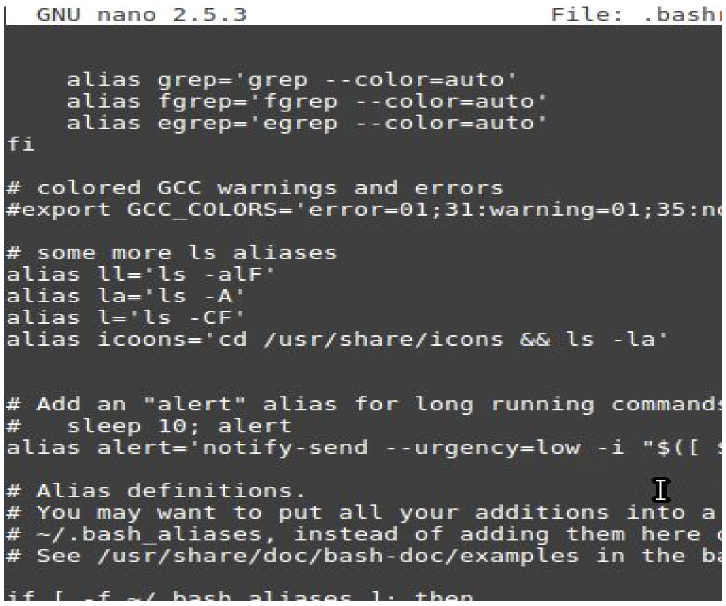
```
total 120
drwxr-xr-x 18 lu   lu   4096 Nov  3 14:41 .
drwxr-xr-x  3 root root 4096 Oct 12 01:00 ..
-rw-r--r--  1 lu   lu    220 Oct 12 01:00 .bash_logout
-rw-r--r--  1 lu   lu   4000 Oct 12 01:00 .bashrc
drwx------  4 lu   lu   4096 Oct 12 01:07 .cache
drwxr-xr-x  3 lu   lu   4096 Nov  3 14:41 .cinnamon
drwxr-xr-x 11 lu   lu   4096 Oct 12 01:07 .config
drwx------  3 lu   lu   4096 Oct 12 01:07 .dbus
drwxr-xr-x  2 lu   lu   4096 Oct 12 01:07 Desktop
-rw-------  1 lu   lu     29 Nov  3 14:41 .dmrc
drwxr-xr-x  2 lu   lu   4096 Oct 12 01:07 Documents
drwxr-xr-x  2 lu   lu   4096 Oct 12 01:07 Downloads
drwx------  2 lu   lu   4096 Oct 27 16:49 .gconf
-rw-------  1 lu   lu    966 Nov  3 14:41 .ICEauthority
drwxr-xr-x  3 lu   lu   4096 Oct 12 01:07 .linuxmint
drwxr-xr-x  3 lu   lu   4096 Oct 12 01:07 .local
drwxr-xr-x  3 lu   lu   4096 Oct 12 01:00 .mozilla
drwxr-xr-x  2 lu   lu   4096 Oct 12 01:07 Music
drwxr-xr-x  2 lu   lu   4096 Oct 12 01:07 Pictures
-rw-r--r--  1 lu   lu    675 Oct 12 01:00 .profile
drwxr-xr-x  2 lu   lu   4096 Oct 12 01:07 Public
drwxr-xr-x  2 lu   lu   4096 Oct 12 01:07 Templates
-rw-r-----  1 lu   lu      5 Nov  3 14:41 .vboxclient-clipboard.pid
-rw-r-----  1 lu   lu      5 Nov  3 14:41 .vboxclient-display.pid
-rw-r-----  1 lu   lu      5 Nov  3 14:41 .vboxclient-draganddrop.pid
-rw-r-----  1 lu   lu      5 Nov  3 14:41 .vboxclient-seamless.pid
drwxr-xr-x  2 lu   lu   4096 Oct 12 01:07 Videos
-rw-------  1 lu   lu    118 Nov  3 14:41 .Xauthority
-rw-r--r--  1 lu   lu   4748 Nov  3 14:41 .xsession-errors
```

For this mint build we have a .bashrc, .profile(.bash_profile) . So the difference between a bashrc file and a profile file is that the , profile file executes scripts on startup of a new login shell. So basically the profile file is used to launch scripts to either limit the user when the user logins or to execute custom scripts every time a new instance of a shell is opened. Configure the shell before it's opened. A .bashrc file is used for custom variables and commands for a particular user, this is where we'll put our alias. So type this command and scroll all the way until you find the **alias** section, or hit ctrl-w to type and search the string **'alias'**

lu@awesome ~/ $ icoons

Naturally linux will tell you there is no such thing as *icoons* command, so let's change that.

lu@awesome ~/ $ nano .bashrc



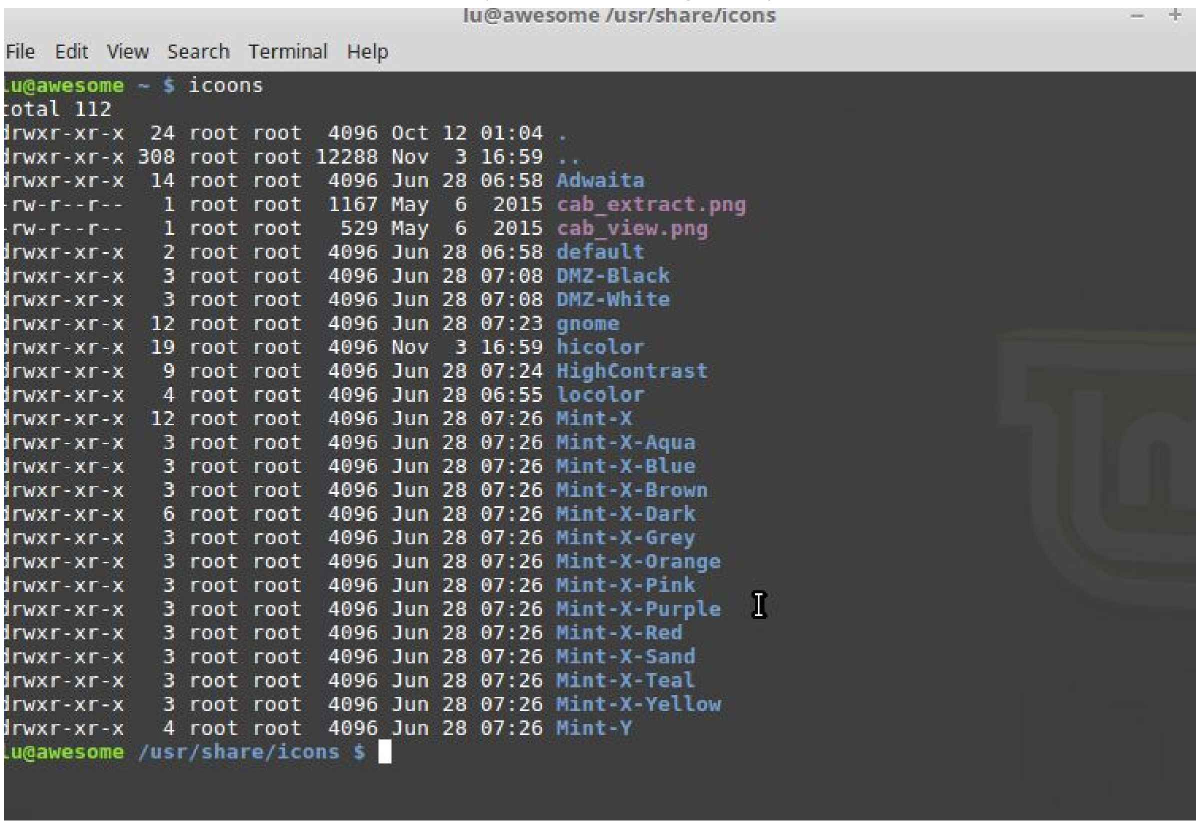
```
 GNU nano 2.5.3                    File:  .bash

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# colored GCC warnings and errors
#export GCC_COLORS='error=01;31:warning=01;35:n

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
alias icoons='cd /usr/share/icons && ls -la'


# Add an "alert" alias for long running command
#   sleep 10; alert
alias alert='notify-send --urgency=low -i "$([ 

# Alias definitions.
# You may want to put all your additions into a
# ~/.bash_aliases, instead of adding them here 
# See /usr/share/doc/bash-doc/examples in the b

if [ -f ~/.bash_aliases ]; then
```

Now after we have made an alias , we need to reload the .bashrc file. To do this we can exit and open another terminal or type in

lu@awesome / $ source ~/.bashrc

lu@awesome / $ icoons

This will run the new command. All my theme icons, globally.



lu@awesome /usr/share/icons
File Edit View Search Terminal Help

```
u@awesome ~ $ icoons
otal 112
rwxr-xr-x  24 root root  4096 Oct 12 01:04 .
rwxr-xr-x 308 root root 12288 Nov  3 16:59 ..
rwxr-xr-x  14 root root  4096 Jun 28 06:58 Adwaita
-rw-r--r--   1 root root  1167 May  6  2015 cab_extract.png
-rw-r--r--   1 root root   529 May  6  2015 cab_view.png
rwxr-xr-x   2 root root  4096 Jun 28 06:58 default
rwxr-xr-x   3 root root  4096 Jun 28 07:08 DMZ-Black
rwxr-xr-x   3 root root  4096 Jun 28 07:08 DMZ-White
rwxr-xr-x  12 root root  4096 Jun 28 07:23 gnome
rwxr-xr-x  19 root root  4096 Nov  3 16:59 hicolor
rwxr-xr-x   9 root root  4096 Jun 28 07:24 HighContrast
rwxr-xr-x   4 root root  4096 Jun 28 06:55 locolor
rwxr-xr-x  12 root root  4096 Jun 28 07:26 Mint-X
rwxr-xr-x   3 root root  4096 Jun 28 07:26 Mint-X-Aqua
rwxr-xr-x   3 root root  4096 Jun 28 07:26 Mint-X-Blue
rwxr-xr-x   3 root root  4096 Jun 28 07:26 Mint-X-Brown
rwxr-xr-x   6 root root  4096 Jun 28 07:26 Mint-X-Dark
rwxr-xr-x   3 root root  4096 Jun 28 07:26 Mint-X-Grey
rwxr-xr-x   3 root root  4096 Jun 28 07:26 Mint-X-Orange
rwxr-xr-x   3 root root  4096 Jun 28 07:26 Mint-X-Pink
rwxr-xr-x   3 root root  4096 Jun 28 07:26 Mint-X-Purple
rwxr-xr-x   3 root root  4096 Jun 28 07:26 Mint-X-Red
rwxr-xr-x   3 root root  4096 Jun 28 07:26 Mint-X-Sand
rwxr-xr-x   3 root root  4096 Jun 28 07:26 Mint-X-Teal
rwxr-xr-x   3 root root  4096 Jun 28 07:26 Mint-X-Yellow
rwxr-xr-x   4 root root  4096 Jun 28 07:26 Mint-Y
u@awesome /usr/share/icons $
```

What takes precedence? ~/.bashrc or /bin/sh ? the ~/.bashrc will.

Try it, make an alias of ‘cd’ and set it to a command, the cd command will be replaced by the alias you created.

Now just like there are hundreds of desktops and thousands of different ways to do things in linux, the shell and bash(ell) are not different, there is tons of different shells out there, such as sh,bash,zsh etc….. So how can i check which shell i'm using?

```
lu@awesome / $  echo $0
```

To some things up

.bash_profile is executed for login shells. When you login (type username and password) via console, either sitting at the machine, or remotely via ssh: **.bash_profile** is executed to configure your shell before the initial command prompt

bashrc is executed for interactive non-login shells. Such as an Alias or custom variables you want loaded into your environment (terminal & desktop)

2.3 Init: SysV, OpenRC, SystemD, Upstart

Init runs scripts one at a time, not simultaneously, init stands for initialization and is managed by either, openRC sysVinit, upstart or SystemD

INIT General:
/etc/inittab : calls the scripts for each run level
/etc/init.d/rc : followed by a number for the runlevel
    S scripts are entering the run level
    K scripts are executing the run level

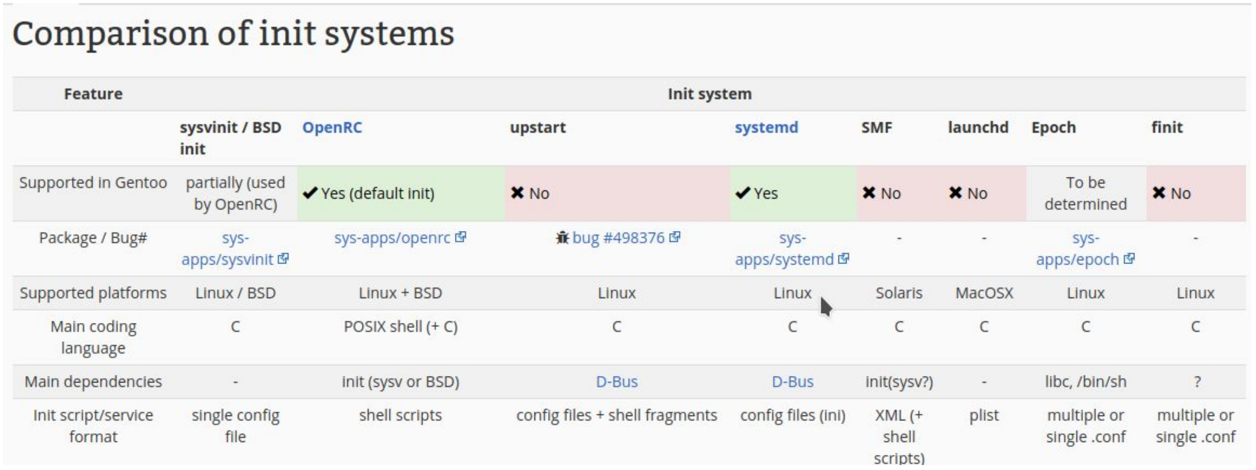
UPSTART:
A replacement for Sysvinit, better support for hotplug devices, cleaner and faster service management, works asynchronously. As of today not a single major distribution supports upstart. Not even ubuntu, the distro that created it. This is because debian adopted systemD instead and so did the rest of the world. However RHEL6 and centOS6 includes upstart (scientific linux, oracle) RHEL 7 has systemd.

/etc/init/ : configuration files located in here
Initctl: controls services, (apachectl)
Initctl list: will list all the process running by upstart

SYSTEMD:
Most distributions have adopted this, it is just as stable as openRC and is faster. Even easier to learn than /etc/init.d/SCRIPT (stop|restart|start) .

/etc/systemd : configuration files in here

Systemctl (param*): is main command that controls services.
Systemctl status network.service: will tell you all about this service.
Systemctl start network.service: will start network service.
Systemctl stop network.service :   will stop network service.
Systemctl restart network.service: will restart network service.



Comparison of init systems

Feature	Init system							
	sysvinit / BSD init	OpenRC	upstart	systemd	SMF	launchd	Epoch	finit
Supported in Gentoo	partially (used by OpenRC)	✔ Yes (default init)	✖ No	✔ Yes	✖ No	✖ No	To be determined	✖ No
Package / Bug#	sys-apps/sysvinit	sys-apps/openrc	bug #498376	sys-apps/systemd	-	-	sys-apps/epoch	-
Supported platforms	Linux / BSD	Linux + BSD	Linux	Linux	Solaris	MacOSX	Linux	Linux
Main coding language	C	POSIX shell (+ C)	C	C	C	C	C	C
Main dependencies	-	init (sysv or BSD)	D-Bus	D-Bus	init(sysv?)	-	libc, /bin/sh	?
Init script/service format	single config file	shell scripts	config files + shell fragments	config files (ini)	XML (+ shell scripts)	plist	multiple or single .conf	multiple or single .conf

Gentoo Boot time Comparison: https://www.youtube.com/watch?v=4NXMmHYNYfA

Run levels is a state of init in which the system boots programs and scripts, there are usually 6, and they vary from distro to distro. For older machines, or distros that have not adapted to systemd have the typical

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.

Redirecting output && Redirecting Input : pe -e > output.txt
Pipes : netstat -tulpn | LISTEN
Apt-get & aptitude :