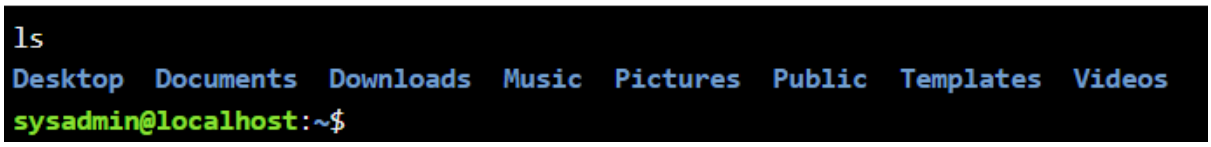
LINUX UNATACHED

Family of open source operating systems built around the Linux kernel.

Linux kernel is the core, open source component in the Linux operating system. Runs on the hardware computing system. An operating system is software that allows other programs like word processors and web browsers to be installed and run on a computer.

Linux command line is a simple interface, (CLI) but there is also a GUI which is what you use on an everyday computer. But behind the scenes is code.

Works for extremely well for low power devices to administrate. It could also work for cloud computers or more advanced systems.



Command: "ls" displays a listing of information about files.

Most commands follow a simple pattern:

Command | [Options] | [Arguments]

As we can see, ls doesn't follow any arguments in this case. However, it can be!

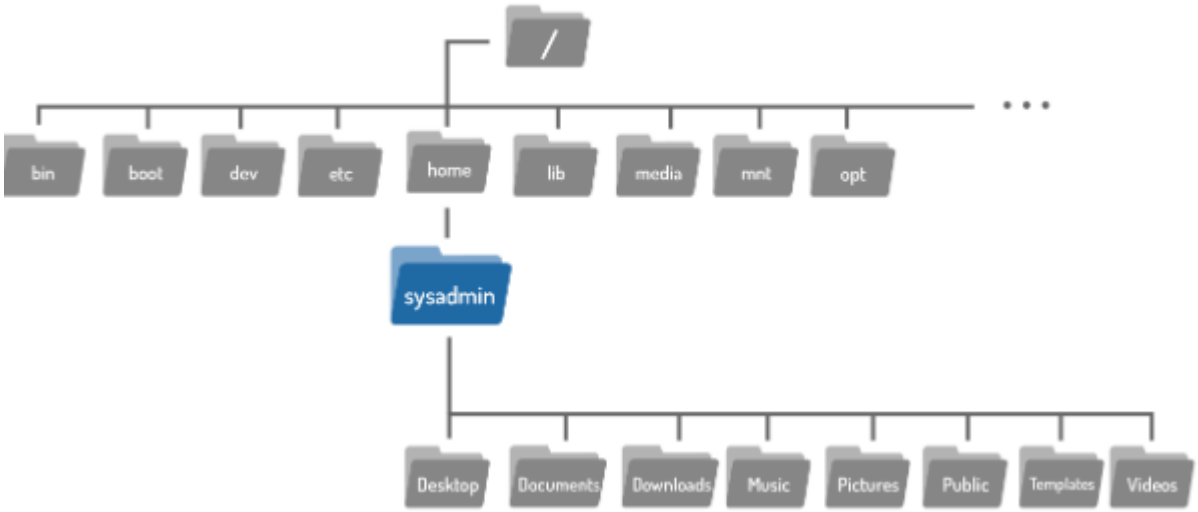
Command: "ls Documents" Which will list the documents contained inside the Documents folder.

But this command follows the criteria **[Arguments]**

Command: "ls -l" Alters the behaviour because we are changing the **[Options]** which **results** in the output being "long display".

Command: "ls -r" **Reverses** the output of the display, or inverts it, so the last item becomes the first.

Command: "pwd" prints the **working directory!**



This would be the output: **/home/sysadmin**

Command: "cd" meaning change directory follows the criteria **cd [Options] | [Path]**

Command: "cd Documents" means the working directory will be **~/Documents**

/home/sysadmin = ABSOLUTE PATH – Specifies exact location

~/Documents = RELITIVE PATH – Specifies relative location without specifying file path

Command: "cd .." No matter the file path, will go up 1 level.

```
sysadmin@localhost:~/Documents/School/Art$ cd ..
sysadmin@localhost:~/Documents/School$
```

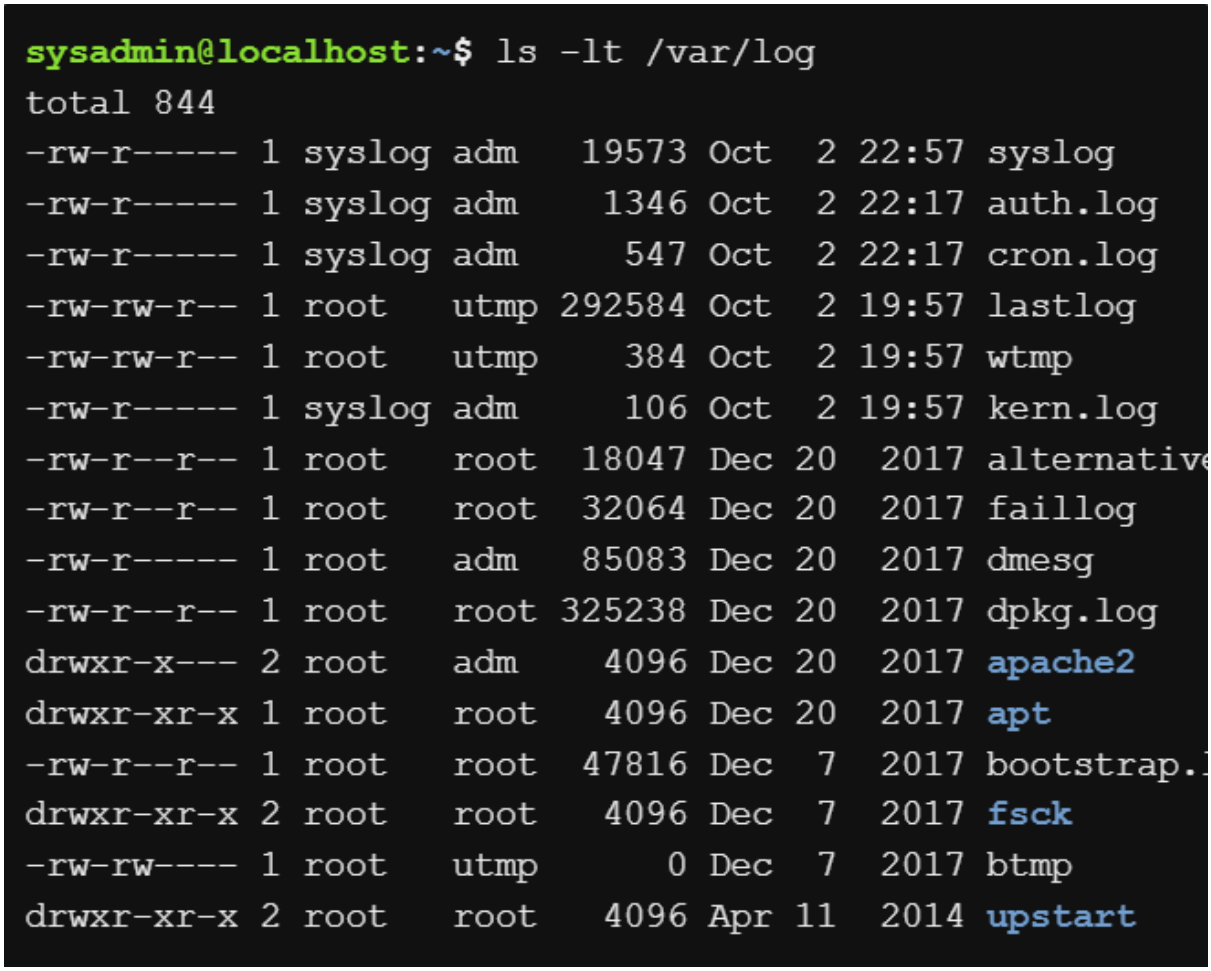
Command: The "." Character always represents your current directory.

Command: The "~" Character will return you to the home directory.

Listening files

Command: "ls" -> Use any "-" command to filter the follow files:

Symbol	File Type	Description
`d`	directory	A file used to store other files.
`-`	regular file	Includes readable files, images files, binary files, and compressed files.
`l`	symbolic link	Points to another file.
`s`	socket	Allows for communication between processes.
`p`	pipe	Allows for communication between processes.
`b`	block file	Used to communicate with hardware.
`c`	character file	Used to communicate with hardware.



|FILE TYPE|PERMISSIONS|HARD LINK COUNT|USER OWNER|FILE SIZE|TIMESTAMP|FILENAME|

How to filter?

Command: “ls -lt” Will filter by time stamp

Command: “ls -l -S” Will filter by file size

Command: “ls -lSr” Will filter the reverse order of file sizes

Administration Access:

Command: “su” Will allow you to temporarily act as a different user. Creating a new shell. A simple console.

Command: “su –“, “su -l”. “su –login” use temporary shells configuring a login with a new user.

Command: “exit” To exit the interface.

Command: “logout” To return to the existing user / default user.

Command: “sudo” is used to switch to another account.

Command: “sudo -sl” to specify a specific user or

Command: “sudo sl” to specify ROOT user

Permission Types

There are three different permissions that can be placed on a file or directory: read, write, and execute. The manner in which these permissions apply differs for files and directories, as shown in the chart below:

Permission	Effects on File	Effects on Directory
read (`r`)	Allows for file contents to be read or copied.	Without execute permission on the directory, allows for a non-detailed listing of files. With execute permission, `ls -l` can provide a detailed listing.
write (`w`)	Allows for contents to be modified or overwritten. Allows for files to be added or removed from a directory.	For this permission to work, the directory must also have execute permission.
execute (`x`)	Allows for a file to be run as a process, although script files	Allows a user to change to the directory if parent directories have execute

Changing file permissions:

Command: “chmod” which translates to change modes of access.

Command: “chmod” **[<SET><ACTION><PERMISSIONS>] FILE**

```
chmod [ <SET> <ACTION><PERMISSIONS>]... FILE
```

Symbol	Meaning
u	User: The user who owns the file.
g	Group: The group who owns the file.
o	Others: Anyone other than the user owner or member of the group owner.
a	All: Refers to the user, group and others.

```
chmod [<SET> <ACTION> <PERMISSIONS>]... FILE
```

Symbol	Meaning
+	Add the permission, if necessary
=	Specify the exact permission
-	Remove the permission, if necessary

After an action symbol, specify one or more permissions to be acted upon.

```
chmod [<SET><ACTION><PERMISSIONS>]... FILE
```

Symbol	Meaning
r	read
w	write
x	execute

Finally specify the file name

Command: “chmod u+x hello.sh” What this does is:

chmod = Change mode access

u+x = User owner permission set + executable permission

Command: “ls -l hello.sh” Will list permissions / check to see if the command worked.

Changing file ownership ->

Command: “chown” Used to change the ownership of files and directories.

chown [OPTIONS] [OWNER] FILE

Command: “cat” **[OPTIONS] [FILE]** to open / view file.

Command: “head” views upper portion of text

Command: “tail” views lower portion of text

Command: “head -n” or “tail -n” number_of_lines filename.

Command: “head -n 3 alpha.txt” removes the first 3 lines from the text.

Command: “cp” **[OPTIONS] SOURCE DESTINATION** to copy a file to a destination.

Command: “dd” Creates a tempory file /tmp/swapex with 50 blocks of 0, 1MB in size.

dd uses a specialised command, eg:

Command: “dd if=/dev/zero of=/tmp/swapex bs=1M count=50”

Of= specialised output file is to be written

Bs= Block size to be used.

Count= Amount of blocks to be written to output.

If the whole file was to be copied, a Block size doesn't have to be specified.

Command: "mv" moves the file. **Mv | SOURCE | DESTINATION**

Command: "rm" removes the file. **Rm | OPTION | FILE**

RM COMMAND REMOVES PERMANANTLY

Command: "grep" is a text filter. **Grep | [OPTIONS] | Pattern [FILE]**

Could be used in this way: **"grep sysadmin passwd"**

Grep can filter out information that isn't needed.

Basic Regex Character(s)	Meaning
.	Any one single character
[ ]	Any one specified character
[^]	Not the one specified character
*	Zero or more of the previous character
^	If first character in the pattern, then pattern must be at beginning of the line to match, otherwise just a literal ^
$	If last character in the pattern, then pattern must be at the end of the line to match, otherwise just a literal $

Command: "egrep" is extended filter->

Extended Regex Character(s)	Meaning
+	One or more of the previous pattern
?	The preceding pattern is optional
{ }	Specify minimum, maximum or exact matches of the previous pattern
|	Alternation - a logical "or"
( )	Used to create groups

Command: “grep ‘root’ passwd” specify user.

Command: “grep ‘^root’ /etc/passwd”

Command: “grep ‘r..f’” **FILENAME** will filter words with 4 letters, beginning with r and ending with f. Useful for finding specific words in a huge text document.

Command: “shutdown” **[OPTIONS] TIME [MESSAGE]**

To do the shutdown command we must have root access.

Command: “shutdown now” -> Shuts down the system.

Command: “shutdown 01:10” will specify the time of which shutdown will occur.

Command: “ipconfig” lists all interface IP configurations

Command: “ipconfig /all” Lists all IP connections

Command: “ping -c” How many iterations, such as -c 4 would be 4 iterations.

Command: “ps” lists processes.

- `PID` : The process identifier, which is unique to the process. This information is useful for controlling the process by its ID number.
- `TTY` : The name of the terminal where the process is running. This information is useful for distinguishing between different processes that have the same name.
- `TIME` : The total amount of processor time used by the process. Typically, this information isn't used by regular users.
- `CMD` : The command that started the process.

Command: “ps -e” Lists all running processes.

Command: “ps -eg” Adds more columns – More detail.

Packet management can be installed through linux, two of the most common ones are Debien and Red Hat.

Command: “dpkg” is the lowest level of Debien packet management

Command: “apt-get” front end program.

Command: “sudo apt-get update” To do an update.

Command: “apt-cache search” **KEYWORD** Multiple keywords could be used to further clarify the search.

Command: “sudo apt-get install [package]” Will install new package specified

Command: “sudo apt-get upgrade” Will look for better versions of specified package.

Command: “sudo apt-get remove [package]” Will remove package from the system.

Or replace remove with purge will completely remove it from the system entirely.

Updating user passwords can be done via the command line interface:

Command: “passwd [OPTIONS] [USER]”

Field	Example	Meaning
User Name	`sysadmin`	The name of the user.
Password Status	`P`	`P` indicates a usable password. `L` indicates a locked password. `NP` indicates no password.
Change Date	`03/01/2015`	The date when the password was last changed.
Minimum	`0`	The minimum number of days that must pass before the current password can be changed by the user.
Maximum	`99999`	The maximum number of days remaining for the password to expire.
Warn	`7`	The number of days prior to password expiry that the user is warned.
Inactive	`-1`	The number of days after password expiry that the user account remains active.

Command: “vi” Vee-Eye or “vim” Have two aspects, 1. A motion and 2. An operational number prefix (how many times it is done)

Motion	Result
`h`	Left one character
`j`	Down one line
`k`	Up one line
`l`	Right one character
`w`	One word forward
`b`	One word back
`^`	Beginning of line
`$`	End of the line

Command Mode Actions

The standard convention for editing content with word processors is to use copy, cut, and paste. The `vi` program has none of these. Instead, `vi` uses the following three commands:

Standard	Vi	Meaning
cut	`d`	delete
copy	`y`	yank
paste	`P` | `p`	put

Delete

Delete removes the indicated text from the page and saves it into the buffer, the buffer being the equivalent of the "clipboard" used in Windows or Mac OSX. The following table provides some common usage examples:

Action	Result
`dd`	Delete current line
`3dd`	Delete the next three lines
`dw`	Delete the current word
`d3w`	Delete the next three words
`d4h`	Delete four characters to the left

Change

Change is very similar to delete; the text is removed and saved into the buffer, however, the program is switched to insert mode to allow immediate changes to the text. The following table provides some common usage examples:

Action	Result
`cc`	Change current line
`cw`	Change current word
`c3w`	Change the next three words
`c5h`	Change five characters to the left

Yank

Yank places content into the buffer without deleting it. The following table provides some common usage examples:

Action	Result
`yy`	Yank current line
`3yy`	Yank the next three lines
`yw`	Yank the current word
`y$`	Yank to the end of the line

Put

Put places the text saved in the buffer either before or after the cursor position. Notice that these are the only two options, put does not use the motions like the previous action commands.

Action	Result
`p`	Put (paste) after cursor
`P`	Put before cursor

Searching in vi

Another standard function that word processors offer is find. Often, people use **CTRL+F** or look under the edit menu. The `vi` program uses search. Search is more powerful than *find* because it supports both literal text patterns and regular expressions.

To search forward from the current position of the cursor, use the `/` to start the search, type a search term, and then press the **Enter** key to begin the search. The cursor will move to the first match that is found.

To proceed to the next match using the same pattern, press the `n` key. To go back to a previous match, press the `N` key. If the end or the beginning of the document is reached, the search will automatically wrap around to the other side of the document.

To start searching backwards from the cursor position, start by typing `?`, then type the pattern to search for matches and press the **Enter** key.

Insert Mode

Insert mode is used to add text to the document. There a few ways to enter insert mode from command mode, each differentiated by where the text insertion will begin. The following table covers the most common:

Input	Purpose
`a`	Enter insert mode right after the cursor
`A`	Enter insert mode at the end of the line
`i`	Enter insert mode right before the cursor
`I`	Enter insert mode at the beginning of the line
`o`	Enter insert mode on a blank line after the cursor
`O`	Enter insert mode on a blank line before the cursor

Input	Purpose
`:w`	Write the current file to the filesystem
`:w filename`	Save a copy of the current file as *filename*
`:w!`	Force writing to the current file
`:1`	Go to line number 1 or whatever number is given
`:e filename`	Open *filename*
`:q`	Quit if no changes made to file
`:q!`	Quit without saving changes to file