The Linux Operating System

a freely available alternative to Windows developed mostly by brilliant young computer programmers in their spare time

Table of Contents

A bit about Linux

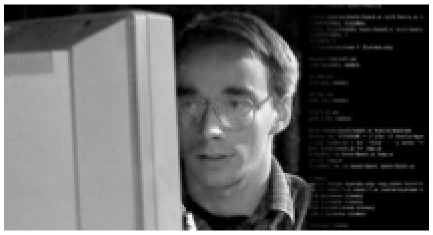
- Linux is a freely available multi-user operating system based on Unix.

  Technically, Linux is only the Kernel or heart of the OS, and is usually combined with Open Source Software to make it useful. The combination is formally known as GNU/Linux.

- Growing from a University of Helsinki thesis, Linus Torvalds first released the Linux Kernel in 1991.
- There are many different distributions of Linux. The primary difference is what is bundled and how it is initially set up.
- Linux and the GNU Utilities included with it are Open Source, meaning the source code is always freely available. This “always free” aspect allows you to freely alter the code as you may desire. Although uncommon, there can be a charge for some software, usually for professional or business applications.
- Linux and most of the software is licensed under the GNU Public License (GPL) or one of several related GNU licenses. The GPL, ensures that all modifications made to GPL software (including Linux) are offered back to the community. This is often referred to as “Copy Left”.

The Famous Newsgroup Post

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)

Newsgroups: comp.os.minix

Subject: What would you like to see most in minix?

Summary: small poll for my new operating system

Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>

Date: 25 Aug 91 20:57:08 GMT

Organization: University of Helsinki

Hello everybody out there using minix –

I'm doing a (free) operating system (just a hobby, won't be big and professional
like gnu) for 386(486) AT clones. This has been brewing since april, and is
starting to get ready. I'd like any feedback on things people like/dislike in
minix, as my OS resembles it somewhat (same physical layout of the file-system
(due to practical reasons) among other things).


I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This
implies that I'll get something practical within a few months, and I'd like to
know what features most people would want. Any suggestions are welcome, but I
won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes – it's free of any minix code, and it has a multi-threaded fs.

It is NOT protable (uses 386 task switching etc), and it probably neverwill
support anything other than AT-harddisks, as that's all I have :-(.
```



Where is Linux Used?

- Air Traffic Control Systems
- Railway Control Systems

  (including Japan's Bullet Trains)
- Amadeus Altea

  (world's largest airline reservation system)
- Android smart phones
- Sony Playstation
- many Netbooks
- U.S. Nuclear Submarines
- Most Smart TVs
- Googles self-driving cars
- Vmware Virtual Servers
- most web hosting
- most cloud computing
- The One Laptop Per Child Initiative
- 8 out of 10 of the world's financial transactions
- 98% of the worlds super computers
- Google
- Facebook
- Twitter
- Yahoo
- Amazon
- DNA sequencing software
- Cern's Large Hadron Collider
- The International Space Station
- McDonald's
- Pizza Hut
- Tommy Hilfiger
- Travelocity
- Walmart
- The New York Stock Exchange
- The U.S. Postal Service
- Schools, Colleges, Universities
- Governments around the world
- The White House

Why Learn Linux?

- Linux is used in almost every industry.
- Learning Linux is interesting. It is not just memorizing how the factory made it work. If you like opening things up to see how they work, or enjoy tinkering, you'll love Linux.
- You can install it on old hardware, create your own custom computer, or even build your own Linux based distribution and offer it to the world!
- Career wise, there is a constant growing need for Linux support. If you work with Internet technologies, Web hosting or almost anywhere else, there will be distinct advantages to having a solid understanding of Linux, even if it is not your primary focus.
- Linux is tightly linked to networking. It provides the tools needed to aid in both understanding networks (private or public), as well as troubleshooting network issues.
- Linux Systems Administrators earn higher salaries than their Windows or Apple counterparts.

Distributions

- Linux needs applications to be useful. The combination of the Linux Kernel, standard GNU utilities, and Open Source Software make up a “Linux Distribution”.
- There are literally hundreds of distributions with a dozen or so making up the majority of installs.

- Many of the popular distributions provide “live” disks that can be run in memory from a DVD or USB drive.
- All distribution are released under a GNU License, such as the GPL, which ensures the source code is available. This means developers can customize any aspect of Linux and any bundled software.

The GNU Public License

(copyleft)

GNU GENERAL PUBLIC LICENSE

- current version Version 3, 29 June 2007
- Free Software Foundation, Inc.   http://fsf.org/

The Foundations of the GPL:

Nobody should be restricted by the software they use. There are four freedoms that every user should have:

1. the freedom to use the software for any purpose
2. the freedom to change the software to suit your needs
3. the freedom to share the software with your friends and neighbors
4. the freedom to share the changes you make.

When a program offers users all of these freedoms, we call it free software.

Red Hat Derivatives

Red Hat

- Enterprise Level. Commercial offering. This is North America's leading enterprise level distribution. Red Hat offers several levels of support services which are often critical to large businesses.

Centos

- “Community Enterprise Operating System” is an exact Red Hat clone. Free, community supported.

Fedora

- Red Hat's free development/testing branch
- Live DVD/USB. Installation optional but not required

Red Hat Derivatives continued...

Red Hat Derivatives (continued)

- Fedora Spins (purpose built sub-distributions):

Base Fedora (uses GNOME Desktop)
KDE (popular alternative desktop)
LXDE (lightweight desktop)
MATE Compiz (3D Desktop)
Xfce (fast, lightweight desktop)
Design (creativity)
Electronic Lab
Games

Jam (Audio)
Robotics
Scientific
Security Lab
Sugar on a Stick
(fits on a USB drive. Developed for the One Laptop Per Child Project)

- There are over a dozen other independent distributions based on Red Hat as well.

Ubuntu Linux

- Debian is a popular distribution particularly for home users. It was introduced in 1993.
- Although many distributions are based on Debian, the most note worthy is Ubuntu which has quickly risen in rank for both home systems as well as use in devices. It is starting to see it's way into enterprise in the last few years as well.
- The good and bad with Ubuntu is that is protects the user from doing any damage, much like OS X. Seasoned users may find it a bit restrictive in it's out-of-the-box state, but like all distributions, it can be adjusted to suit your needs.
- Ubuntu is found in the majority of preinstalled Linux based home computers, netbooks and laptops.

The Good and The Bad

Things to consider...

- Linux is fast, stable and runs on minimal hardware.
- Linux is free and can even run from a CD without installation on the computer so you don't even need a spare computer, but lots of RAM helps for this method.
- Virtually every application you want or need will be included with the installation or easily added afterwords.
- Some proprietary application vendors will not support Linux or release the code. An example is that you can make a topographical map (this is a long complex procedure in any environment) but you cannot convert it to use as a Garmin GPS Map without Garmins Windows-only application. These situations are few and far between but they do exist.
- There is a huge community of Linux enthusiasts and many Internet forums available when you need help.
- Aside from a hardware problem, rebooting is virtually never a requirement. Most misbehaving applications are easily killed.
- The initial learning curve is very steep, especially if you are used to Windows. With time it does become logical, predictable, and the simplest system to use and understand. This requires hands-on time and conscious effort.

Good and Bad continued...

The Good and The Bad (Continued)

- The GUI (graphical user interface) or desktop, like Windows or Apple use, is just another application. You can try several different ones, combine their features or not use graphics at all. It's up to you. The popular desktops are significantly more advanced and flexible than any Microsoft environment. The cost is that when customizing or mixing the different desktop environments can be complex and sometimes frustrating.
- The only differences between a Linux desktop and a Linux server are the applications installed, and possible kernel tweaks. The GUI or desktop is not normally run (or even installed) on a server as a graphical environment uses significant resources and because servers are typically accessed remotely using the command line.
- Linux is addictive. Ask a windows user why they use Microsoft and the answer is most often “I don't know, it came with my PC”. Ask a Linux user why they use Linux and you are in for a long conversation filled with passion and excitement, and most likely including some cool home grown project they did.
- As far as I know there are very few popular games that have been ported to Linux.

Tips for Getting Started

- Like learning anything new, Linux may seem confusing at first
  - don't get discouraged when things don't work as expected
  - it's not Windows or Mac, some things will work differently
- Don't be afraid to experiment with commands
  - as a normal user you cannot do much harm
- Linux gives you access to the entire system
  - everything is a file
  - many files are plain text
- Linux is <u>Case Sensitive</u>
  - “file.txt”, “File.txt” and “FILE.txt” are all different files
  - neither Windows nor Mac are case sensitive

Many Small Tools

- Linux uses <u>many small utilities</u> and not very many mega tools. At the command line many commands can be run together providing very flexible solutions.

- As an example, if you needed to make a 50% reduced size copy of each of 100 images spread out in several sub-directories. Instead of using a graphical application requiring you to open each image, you might choose to do it like this:

```
for each ORIG_IMAGE in `find . -name "*.jpg"`
  do
    convert ${ORIG_IMAGE} -scale 50% ${ORIG_IMAGE/.jpg/-new.jpg}
  done
```

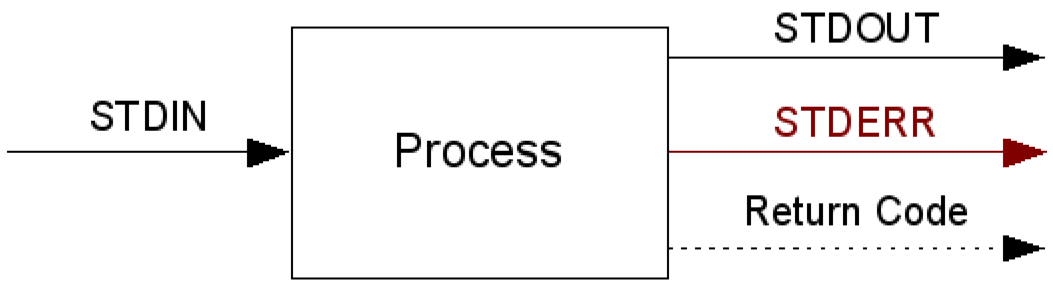
This can also be <u>all on one line</u>:

```
for each ORIG_IMAGE in `find . -name "*.jpg"`;do convert ${ORIG_IMAGE} -scale 50% ${ORIG_IMAGE/.jpg/-new.jpg}; done;
```

Linux Concepts

- The Super User is named “root” and has complete control over all aspects of the system. Generally, servers used in business environments do not allow direct login by user “root”. There are a number of security schemes for doing work as the root user.
- Most services, like a web, ftp or email servers, can only be started by root but then automatically change to run under a less privileged account for security reasons.
- For all intents and purposes there is no “undelete” so be careful when logged in as root!
- File separator is a forward slash “/”. This is the same for Apple but opposite of Windows “\”
- A users PATH (command search path), where the system checks for a command you type, does not include the current directory. This is a security aspect. To use a command in the local directory it must be preceded with a “./” as in './runme'
- Each file has three primary access rights for any particular user or user-group. “read” “write” “execute”. These can be combined in any mix as needed. There are secondary controls as well.
- Help pages are referred to as “man pages” (help manuals). Some are very shy on the details while others are overwhelming. For help with “tr” use “man tr”. Usually “--help” after the command will give a brief usage overview.
- Commands are generally all lower case and command arguments usually start with a “-“ (hyphen) 'ls -a'

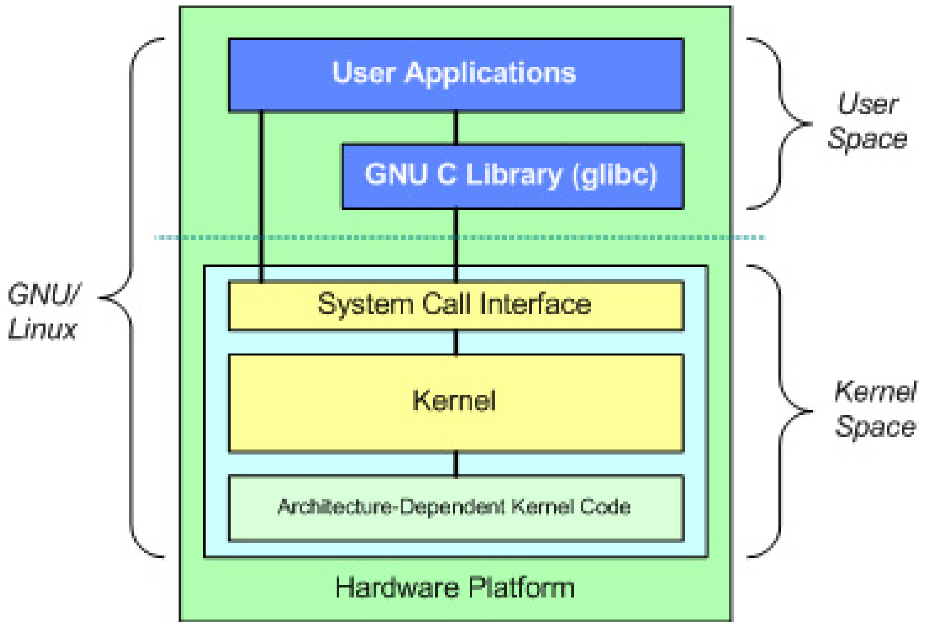
Input, Output, Errors and Exits



- Data entered at the command line is referred to as "*standard input*" or "*standard in*". It is written as "*stdin*"
- Data printed to the screen is referred to as "*standard output*" or "*standard out*" and is written as "*stdout*"
- Any error messages or related output is sent to "*standard error*" (*stderr*) which usually goes to "*standard out*" (the screen) as well as any associated logs.
- The "*return code*", "*exit status*" or "*exit code*" (was it successful or not) of a command is stored in a special variable and often also goes to "*standard out*". An exit status of "0" (zero) means success.
- All of these things can be redirected to files or passed to other commands

How it Fits Together

- GNU/Linux is comprised of two areas, User Space and Kernel Space. The Kernel Space, basically the Linux kernel and related code, sits between the user and the hardware. All hardware is accessed through the kernel.
- The GNU C Library (glibc) is a library of all the system calls used for opening files, printing etc. The user does not normally call these directly but triggers them by running commands. Developers and advanced users use glibc to compile C programming code into a binary executable, or to custom compile an application from source code in order to add or remove supported features.



Linux Directory Structure

Linux uses the **Filesystem Hierarchy Standard** (FHS)

- The FHS describes the common directory structure and location of critical files within Linux (and other Unix-like systems).
- The FHS is only a starting point. Additional directories are added as needed.
- The file system starts at “root” which is denoted as “/”
- Everything in Linux is accessed starting at /
- This is different from Windows which uses separate, unrelated structures for each device such a C:\, D:\, E:\ etc.
- Even remotely mounted file systems (network shares) are accesses as part of the root file system.

Primary Linux Directories

/ - root of all directories in Linux file system structure

/home - keeps Linux user account's home directory

/etc - “environmental text configuration” configuration files and directories

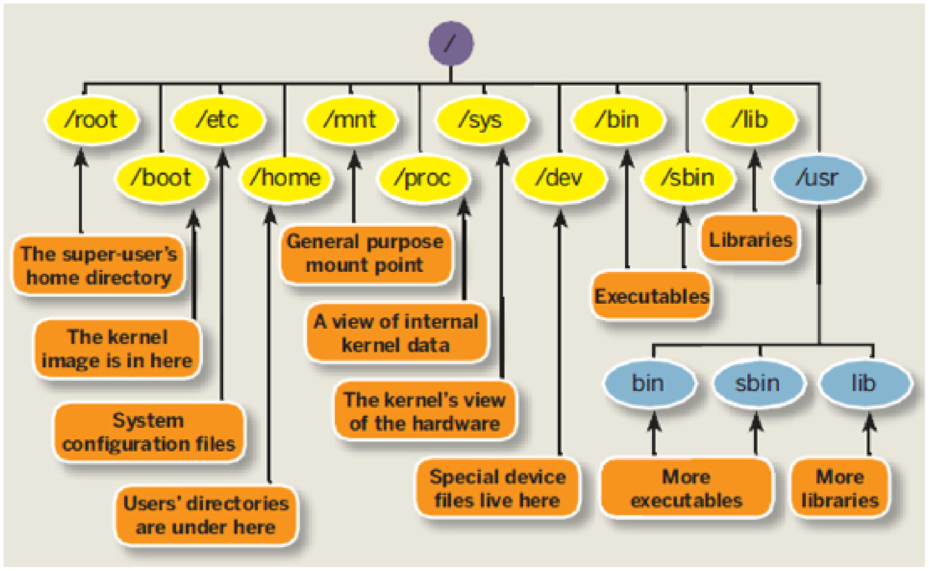
/usr - “Unix system resources” keeps Linux system files

/var - “variable” keeps system log files & changing system files

/bin - “binaries” keeps binary files for user applications

/sbin - “system binaries” keeps binary files for system programs

- This is only a starting point, most systems have additional root-level directories


/
/root
/etc
/mnt
/sys
/bin
/lib
/boot
/home
/proc
/dev
/sbin
/usr
The super-user's home directory
The kernel image is in here
System configuration files
Users' directories are under here
General purpose mount point
A view of internal kernel data
The kernel's view of the hardware
Special device files live here
Libraries
Executables
bin
sbin
lib
More executables
More libraries

File Types

Everything stored in Linux will be one of these types of files:

- Ordinary files

  - readable files - binary files
  - image files - compressed files

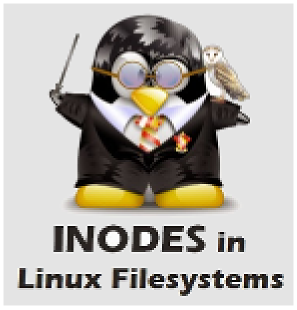
- Directories
  - files that hold a list of what files are in the directory
- Devices
  - Typically block (hard disk) or character devices (terminal display)
- Links
  - *Hard Links* are like a second name for the same file
  - *Symbolic Links* (or soft link), similar to Windows Shortcuts, point to a file name only
- Sockets and Pipes (not commonly used by regular users)
  - A *Socket* file is used to pass information between applications for communication purposes
  - A *Pipe* file (aka "named pipe") is a file that one process or utility writes to, while a second acts on anything written to the pipe.

Example: An application may send data to a named pipe periodically over a long period of time, while a file compression utility (like gzip) watches the pipe and continuously adds any new data to an archive file.

Files and Inodes

Linux files are stored in two distinct and separate pieces.

- inode (Index Node)
  - holds the location on the disk where the file contents are located
  - holds file attributes (access & change time, permissions etc)
  - the file name and inode number are in a separate index used to locate the inode for any given file. This allows associating several file names with one inode (hardlink)
- file (data object)
  - this is the actual data or contents of the file

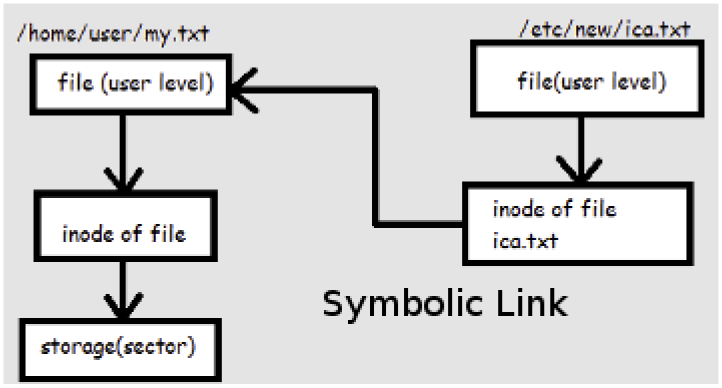


File name (sometimes written as filename)

- the file name is in some perspectives unrelated to the actual file itself
- file name and the associated inode number are stored in a File System Index. This allows several (65k) file names to be associated with one inode and data object.

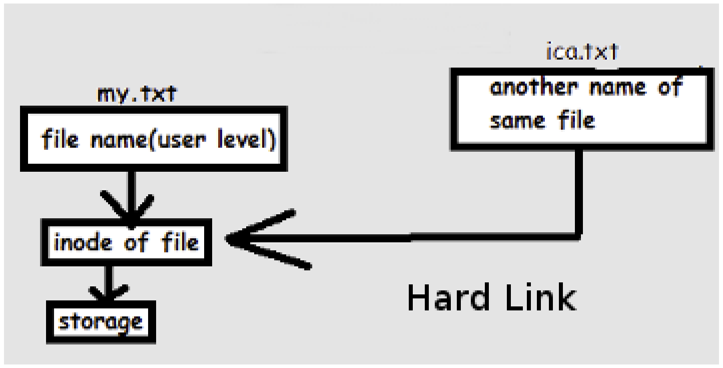
Soft Links and Hard Links

Symlink (Symbolic or Soft Link)



- symlinks are pointers representing the path to another file name only, they are very similar to Windows Shortcuts
- renaming, moving or deleting a file will break the symlink
- deleting a symlink will have no impact on the actual file
- symlinks can point to both remote or local files

Hard Link



- hardlinks are pointers to an inode.
- renaming a file will not impact other hard links
- deleting a file does not break hardlinks because deleting a file removes the file name reference, not the inode or data

Common Commands & Utilities

- Most commands are stored in /bin or /sbin
- “show”, “print”, and “display” all mean ”prints to the screen”

ls	-list the contents of a directory
echo	-print something to the screen. 'echo hello world'
passwd	-change your password
who	-shows who is logged in
w	-like 'who' but more information
id	-prints user and group info
last	-shows log-in history
lastb	-shows bad login attempts (must be root user)
hostname	-display (or changes) computer name
date	-display (or changes) computer date
cal	-calendar utility. 'cal 2003' 'cal 3 1992'
more	-pages through a text file
less	-similar to 'more' but much more
file	-displays some file info like file type
stat	-displays file access and change information
df	-displays file system information

free	-shows memory information
uname	-prints system information
uptime	-display how long the system has been running
ps	-display running process information 'ps -ef'
top	-display top processes currently running (press q to quit)
man	-help manual for a command 'man who' (press q to quit)
apropos	-searches *man pages* for a keyword
clear	-clears the screen
bc	-a calculator
yes	-repeatedly prints a string 'yes hello' (press Ctrl-c to quit)
time	-prints time taken to execute a command
sleep	-pause *n* seconds 'sleep 5'
history	-print a list of recently used commands
which	-displays the full path to a command 'which ls'
type	-similar to 'which' but includes built-in commands
ip	-network information/configuration tool. Many aspects.
	-ip replaces several (still available) commands
arp	-displays or changes ARP information (use 'ip')
ifconfig	-print/change network interface information (use 'ip')
dig	-DNS queries
ethtool	-query or control network driver and hardware settings
tcpdump	-view traffic passing through a network interface

Command Aliases

Command Aliases provide customized command strings which reduce the characters you need to type/remember, and ensure consistency.

Aliases are defined in two locations:

- system wide aliases:
  - apply to all users
  - stored in /etc/bashrc
- user defined aliases:
  - apply to only one user
  - stored in ~/.bashrc

(~ represents the current users home directory)

The format is simple, the keyword *alias* on one line and the definition *key*=*value* on the next line

example:

alias

lsl='ls -aFl|less'

this created an alias of *lsl* to represent *ls -aFl | less*

Joins, Pipes and Redirects

Most tasks require more than one command to complete, this is where Linux shines.

- semicolon ';' runs two unrelated commands in sequence

  command one ; command two

  ls ; date (prints a directory listing, then the date & time)

- pipe '|' sends results of one command as input to another

  command one | command two

  history | grep myfile

  (get previous commands and extract any with the string “myfile” in them)

- “greater than” '>' sends command results to a new file

  command > filename

  ls > mylist.txt (saves a directory listing to a new file in the current directory)

- double “greater than” '>>' appends command results to the end of an existing file

  command >> filename

  ls >> oldfile.txt (adds the directory listing to the end of an existing file)

Note: spaces are not required between commands and the separator but can be added for readability

Process Management

- Every activity on the system is linked to a Process ID which distinguishes it from other tasks running at the same time.
- There is a dynamic directory named "*/proc*" which holds process information. Utilities like top, free, ps and others read into the "*/proc*" directory to get the output for the command.

Example using *ps* command:

```
#ps -efww

UID        PID    PPID  C STIME  TTY   TIME      CMD

pete      1492     1       0 Feb13    ?     00:10:30  /usr/bin/konsole

pete      1500   1487   0 Feb13 pts/0   00:00:00 /bin/bash
```

- *kill* is used to terminate non-responsive commands.

  kill <process id>

Example of *kill*:

kill 30511

  - kill only displays output if there was a problem, if the process is successfully stopped, it just returns control

Backgrounding Processes

When you have a command that does not require any input, but will run for some time, you can push it off screen and run it in the background. This allows you to continue with other tasks at the command prompt.

- To background a command when you run it, add a "*&*" to the end of the command.

  ./my_script.sh &

- Sending a running job to the background is a three step process:

  1) enter "*control z*" -this freezes the process

  2) enter "*jobs*" to list your running and stopped jobs by job number

  3) enter "*bg n*" where *n* is the job number

- To bring a job to the foreground, run "jobs" to get the job number, then enter "*fg n*" where *n* is the job number

Exit Codes

- Every time a command completes it gets an “Exit Code” or “Exit Status” to indicate success or errors.
- An exit code of *0* (zero) means success, everything else means there was some level of error. Normally it is simply *0* or *1* (one for errors). The actual error number will depend on the utility or command. You can usually get exit codes from the man page.
- When a command completes, the exit code is stored in a variable represented by *$?* but it only has the value of the last command run. To see the code run “*echo $?*” after the command completes but before typing any other commands.
- Exit Codes can be used in scripts to make programming decisions based on a commands exit code.

Variables

Variables are placeholders that represent a numeric/character string.

- Variables are referenced as *$name* where *name* is the variable name.
- There are many built-in or system variables, and you can define them yourself. Variables save a lot of time when repeating long strings or just using the same information in numerous places in a command or script. (A script is just a bunch of commands put into a file.)

- To define a variable do not use the *$*, just the name=value

  my_var=”1234”

  (assigns the value '1234' to variable name $my_var)

- To read a variable use the *echo* command

  echo $my_var

  (prints 1234 to the screen)

- In some cases you do not use echo

  grep $myvar somefile.txt

  (checks if the string “1234” is in the file “somefile.txt”)

Command Line Shortcuts

- *ctrl-r*, searches command history for letter typed, suggesting most recent.
- *history* command will list all your commands on the screen
- rerun commands using “*!*” followed by the history reference number.

  output of *history*:

  ```
  167  top
  168  cat /etc/system-release
  169  su -
  170  ls
  ```

  To re-run “*cat /etc/system-release*” simply type “*!168*” & hit enter
- Tab key will attempt to complete a command or list possibilities
- *ctrl-u* erase cursur to start of line
- *ctrl-a* go to start of line
- *ctrl-e* go to end of line
- *ctrl-b* go left one char
- *ctrl-f* go right one char
- Highlighting with the mouse auto copies, and a middle mouse button click pastes. Fast and simple. May be Desktop dependent.

The Vim Text Editor

In order to do anything significant on a Linux system you need to edit files. Linux distributions ship with any number of text editors (both console and GUI based), but one constant is that any Unix or Linux system you work on will have the standard vi or vim console based text editor.

- vi is the original Unix editor and is included with some Linux distributions although most just use the command as an alias to the newer vim ("Vi Improved") which is much more versatile and simpler to use.

- vim can take some time to get used to and may seem frustrating or cumbersome initially. The time invested will definitely be worthwhile as it is one of the most powerful text editors available and you do need to know how to use it in order to work on any given Linux system.

- Don't get overwhelmed once you start to see all the things vim can do. A coworker once told me of a previous administrator who wrote her entire thesis using vi, which was also the subject of her thesis!

Using Vim

There are three primary modes of use: (the mode names are unofficial)

- **Format Mode**: this is the initial state. It allows single or multiple keys to make various changes such as delete a line, join two lines, etc
- **Insert Mode**: this is for normal content input and file editing.
- **Execute Mode**: allows you to save, quit, open a second file, run external commands, etc.

To start editing a new file which you want to name “newfile.txt” enter:

'*vi newfile.txt*'

- To start adding text: go to *insert* mode by pressing “*i*” (do not press enter), and start typing
- To save and exit vi:
  - go to *execute* mode and run commands:
    1. press “*Esc*” (to exit insert mode), then “*:*” to go to *execute* mode (do not press enter)
    2. followed by *wq* to execute the write & quit commands. Now press enter.
  - So to go from typing in the file to save and quit, type: <escape key> *:wq (press enter)*
- A somewhat complex help menu can be access via *:h* (enter) and *:q!* returns to your working file

Vim Format Mode Commands

Some of the more commonly used **format mode commands**:

dd	-delete an entire line
5dd or d5d	-delete a number of lines
D or d$	-delete from the cursor to end of line
d^	-delete to beginning of line
x or 5x	-delete one or more characters
yy	-copy (yank) an entire line
5y or y5y	-copy (yank) a number of lines
p	-paste the yanked or deleted line(s) below the current line, except when buffer created with a D, d$ or x, then paste is always inserted at the cursor position (on the same line)
P	-paste the yanked or deleted line(s) above the current line, except when buffer created with a D, d$ or x, then paste is always inserted at the cursor position (on the same line)
u	-undo the previous action. Can be used multiple times to roll back several actions
Ctrl-R	-(control key plus R) will redo the previous undo
w	-write/save file (under name it was oped as)
w newname	-write/save file under new file name as indicated
q	-quit
q!	-quit without saving current file
J	-join the current and next lines
.	-(dot) repeat the last command at the current cursor position
G	-go to the last line in the file
G5	-go to the indicated line number

Vim Tips

Although vim has it's own multiple item buffer or clipboard, when used in a terminal window within the desktop, it also allows you to paste from the desktops clipboard (KDE's holds 1024 entries). In some (most?) desktops the middle mouse button will paste the clipboard entry where the mouse is pointed (or a right clip option may be used), but vim does not care about the mouse pointer (or it's location) so it will paste the entry where the text cursor is positioned.

- Vim is “code aware” meaning it can recognize some program or computer languages and will highlight text blocks etc. In some cases or for some users this can be annoying. It can be turned off in two ways:
  - Per session: escape from insert mode and enter

    “*:setlocal formatoptions-=ro*”
  - Permanently: add the following to your *.vimrc* file in your home directory:
    - vi ~/.vimrc   (we edit/create the vim resource configuration file in the current users home directory)
    - Go to insert mode and add these two line to the end of the file: (line 1 is a comment)

      # added to stop auto formatting <your name and date>

      *autocmd FileType * setlocal formatoptions-=ro*

Vim Tips continued...

Vim Tips (continued)

- Custom settings can be saved in your ~/.vimrc file (as in the above example).

  Note that in Linux “~/” simply means “my home directory”. So for the account “*pete*”, *~/.vimrc* is the same as */home/pete/.vimrc*

- You can run external commands or even go to a new shell (command line) from within vim without leaving the application or opening a new window. This is very useful when working remotely as you usually only have one terminal window available. It can also be a security concern on poorly configured/administered systems.

- If you open a text document that was saved in Windows via notepad or a similar application, it will contain DOS style line return character at the end of the lines. You can manually remove them or just run the file through the dos2unix utility.