



Introduction to Linux

Basic Commands & Environment

Alexander B. Pacheco
LTS Research Computing
September 15, 2015

Outline

- 1 What is Linux?
- 2 Variables
- 3 Basic Commands

Installing Linux on VirtualBox

- ❶ Download and install Oracle VirtualBox (and the extension pack) from [here](#)
- ❷ Download the CentOS virtual image from [here](#). (you need to be logged into Lehigh Google to access the image name CentOS.ova. Its about 2.6GB.)
- ❸ Install the image by double clicking on it. If it doesn't work, open virtualbox software that you just installed,
 - ❶ From the menu, click File>Import Appliance
 - ❷ Choose the ova file that you just downloaded and click the next button (this instruction may differ on Windows and Mac systems)
 - ❸ Click the import button
- ❹ Whole process should take a few minutes.
- ❺ Once the process is complete, you should see CentOS listed in the left sidebar.
- ❻ Select CentOS and click the start button or double click CentOS
- ❼ After a minute or two you should see a login prompt
- ❽ Type user and hit enter
- ❾ You should now see a prompt such as `[user@localhost ~]$`
- ❿ Create a password by typing `passwd` and hit enter. You will be prompted to enter a password twice, you will not see any characters on the screen as you type.
- ⓫ Create a password for admin user by first logging in as root: type `su -` at the prompt and hit enter. Follow the previous step

Logging into a remote Linux server

- Mac OSX

- 1 Open the Terminal App
- 2 At the command prompt enter `ssh user@remotehost`
`user` is your username on the remote Linux server
`remotehost` is the hostname or ip address of the remote Linux server
e.g To log into polaris `ssh alp514@polaris.cc.lehigh.edu`

- Windows

- 1 Download and install a ssh client such as putty or MobaXterm
- 2 Open the client
- 3 Putty: Enter the hostname or ip address of the remote Linux server (make sure the SSH radio button is selected) > Click open
- 4 MobaXterm: Click New Session > Select SSH tab > Enter hostname and username in the field provided > Click on OK
- 5 When you are prompted for your password, you may not see any characters on the screen.

What is Linux?

History I

- Unix was conceived and implemented in 1969 at AT&T Bell labs by Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna.
- First released in 1971 and was written in assembler.
- In 1973, Unix was re-written in the programming language C by Dennis Ritchie (with exceptions to the kernel and I/O).
- The availability of an operating system written in a high-level language allowed easier portability to different computer platforms.
- The GNU Project, started in 1983 by Richard Stallman, had the goal of creating a “complete Unix-compatible software system” composed entirely of free software.
- 386BSD released in 1992 and written by Berkeley alumni Lynne Jolitz and William Jolitz. FreeBSD, NetBSD, OpenBSD and NextStep (Mac OSX) descended from this
- Andrew S. Tanenbaum wrote and released MINIX, an inexpensive minimal Unix-like operating system, designed for education in computer science
- Frustrated with licensing issues with MINIX, Linus Torvalds, a student at University of Helsinki began working on his own operating system which eventually became the “Linux Kernel”
- Linus released his kernel for anyone to download and help further development.

Linus's message to comp.os.minix on Aug 26, 1991

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).

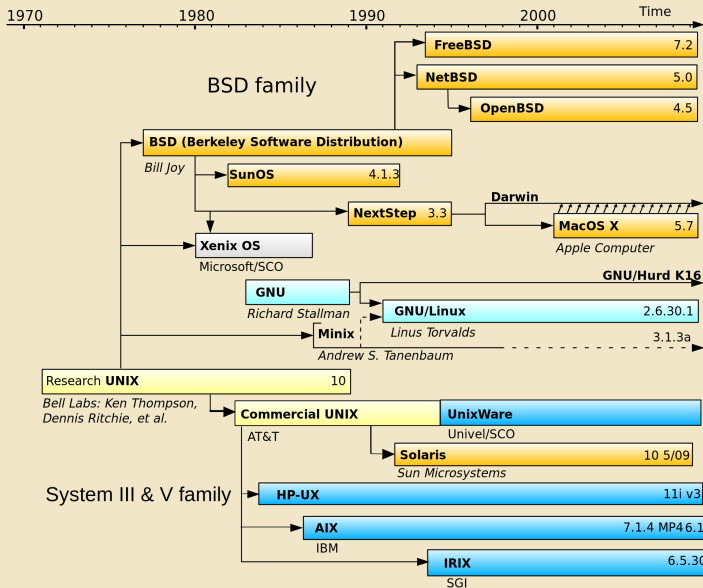
I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (email address)

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 never will support anything other than AT-harddisks, as that's all I have :-).

<https://groups.google.com/forum/?fromgroups=!msg/comp.os.minix/dlNtH7RRrGA/SwRavCzVE7gJ>

- Linux is only the kernel, an Operating System also requires applications that users can use.
- combined with free software available from the GNU project gave birth to a new Operating System known as "GNU/Linux"
- GNU/Linux or simply Linux is released under the GNU Public License: Free to use, modify and distribute provided you distribute under the GNU Public License.



What is Linux?

- Linux is an operating system that evolved from a kernel created by Linus Torvalds when he was a student at the University of Helsinki.
- It's meant to be used as an alternative to other operating systems, Windows, Mac OS, MS-DOS, Solaris and others.
- Linux is the most popular OS used in a Supercomputer

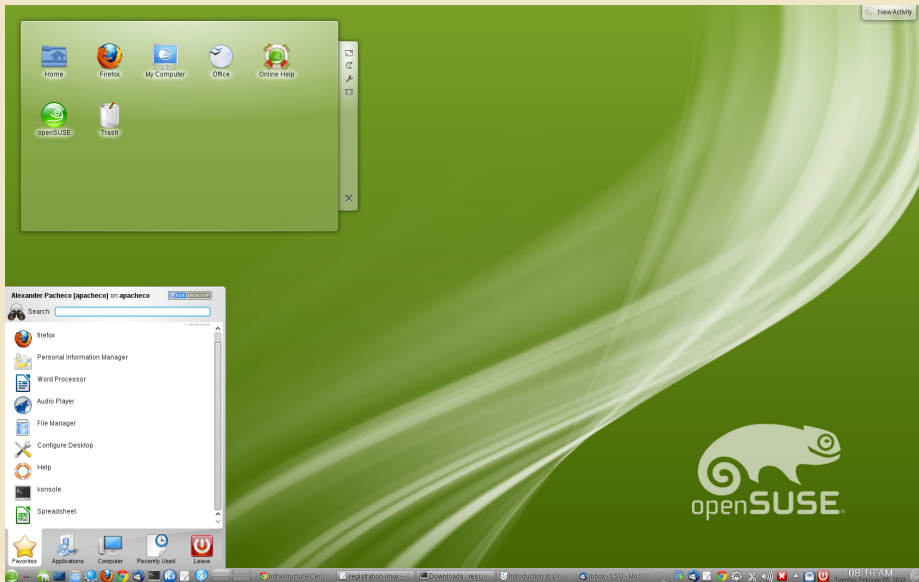
OS Family	Count	Share %
Linux	485	97
Unix	13	2.6
Mixed	1	0.2
Windows	1	0.2

- If you are using a Supercomputer for your research, it will most likely be based on a *nix OS.

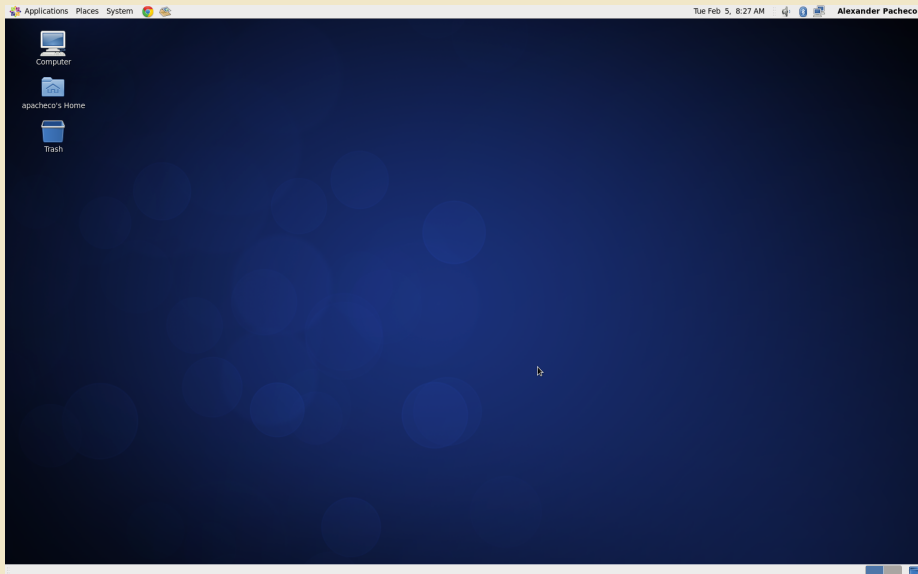
What is Linux?

- Many software vendors release their own packaged Linux OS (kernel, applications) known as distribution
- Linux distribution = Linux kernel + GNU system utilities and libraries + Installation scripts + Management utilities etc.
 - ① Debian, Ubuntu, Mint
 - ② Red Hat, Fedora, CentOS
 - ③ Slackware, openSUSE, SLES, SLED
 - ④ Gentoo
- Application packages on Linux can be installed from source or from customized packages
 - ① deb: Debian based distros e.g. Debian, Ubuntu, Mint
 - ② rpm: Red Hat based distros, Slackware based distros.
- Linux distributions offer a variety of desktop environment.
 - ① K Desktop Environment (KDE)
 - ② GNOME
 - ③ Xfce
 - ④ Lightweight X11 Desktop Environment (LXDE)
 - ⑤ Cinnamon
 - ⑥ MATE

openSUSE KDE Desktop



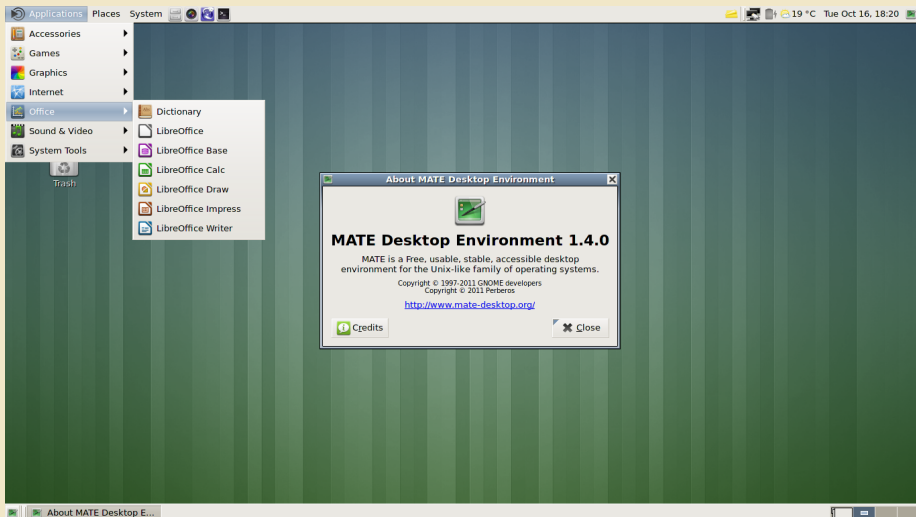
CentOS GNOME Desktop



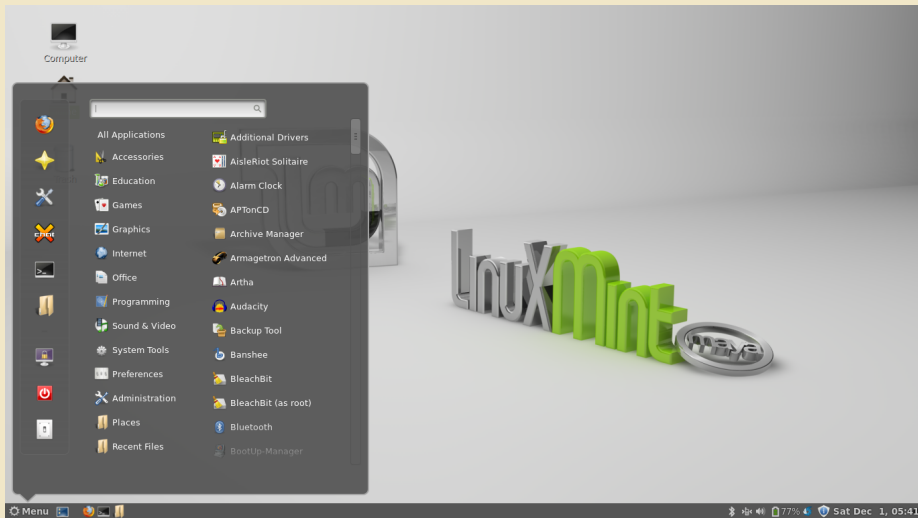
LXDE Desktop



Debian MATE Desktop



Linux Mint Cinnamon Desktop



What is Linux?

- Linux distributions are tailored to different requirements such as
 - ① Server
 - ② Desktop
 - ③ Workstation
 - ④ Routers
 - ⑤ Embedded devices
 - ⑥ Mobile devices (Android is a Linux-based OS)
- Almost any software that you use on windows has a roughly equivalent software on Linux, most often multiple equivalent software

e.g. Microsoft Office equivalents are OpenOffice.org, LibreOffice, KOffice

- For complete list, visit http://wiki.linuxquestions.org/wiki/Linux_software_equivalent_to_Windows_software
- Linux offers you freedom, to choose your desktop environment, software.

Popularity of Linux Distributions

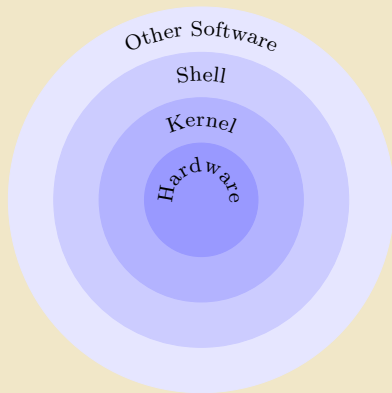
- [DistroWatch](#) provides news, popularity rankings, and other general information about:
 - 1 various Linux distributions,
 - 2 free software/open source Unix-like operating systems such as OpenSolaris, MINIX and BSD.
- DistroWatch is NOT an indication of market-share or quality nor is it an indication of how many users but it is clearly an indication of what users are looking at.

Rank	Distribution	Hits	
1	Mint	2427	▲
2	Ubuntu	1830	▼
3	Debian	1597	▲
4	openSUSE	1420	▲
5	Fedora	1322	▲
6	Mageia	1138	▲
7	CentOS	1116	▲
8	Arch	1035	▼
9	elementary	969	▲
10	Android-x86	832	▲

Linux Components

- Linux is made up of two (three) parts:

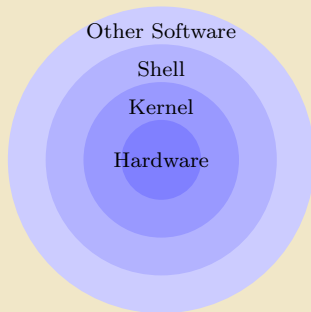
- 1 Kernel
- 2 Shell
- 3 Applications/Programs



Linux Components I

- Linux is made up of two (three) parts:

- 1 Kernel
- 2 Shell
- 3 Applications/Programs



Linux Components II

What is a kernel

- The kernel is the main component of most computer operating systems
- It is a bridge between applications and the actual data processing done at the hardware level.
- The kernel's responsibilities include managing the system's resources (the communication between hardware and software components).
- provides the lowest-level abstraction layer for the resources (especially processors and I/O devices) that application software must control to perform its function.
- It typically makes these facilities available to application processes through inter-process communication mechanisms and system calls.

What is a SHELL

- The command line interface is the primary interface to Linux/Unix operating systems.
- Shells are how command-line interfaces are implemented in Linux/Unix.
- Each shell has varying capabilities and features and the user should choose the shell that best suits their needs.
- The shell is simply an application running on top of the kernel and provides a powerful interface to the system.

Types of Shell

sh : Bourne Shell

- ◆ Developed by Stephen Bourne at AT&T Bell Labs

csh : C Shell

- ◆ Developed by Bill Joy at University of California, Berkeley

ksh : Korn Shell

- ◆ Developed by David Korn at AT&T Bell Labs
- ◆ backward-compatible with the Bourne shell and includes many features of the C shell

bash : Bourne Again Shell

- ◆ Developed by Brian Fox for the GNU Project as a free software replacement for the Bourne shell (sh).
- ◆ Default Shell on Linux and Mac OSX
- ◆ The name is also descriptive of what it did, bashing together the features of sh, csh and ksh

tcsh : TENEX C Shell

- ◆ Developed by Ken Greer at Carnegie Mellon University
- ◆ It is essentially the C shell with programmable command line completion, command-line editing, and a few other features.

Shell Comparison

Software	sh	csH	ksh	bash	tcsh
Programming Language	✓	✓	✓	✓	✓
Shell Variables	✓	✓	✓	✓	✓
Command alias	✗	✓	✓	✓	✓
Command history	✗	✓	✓	✓	✓
Filename completion	✗	★	★	✓	✓
Command line editing	✗	✗	★	✓	✓
Job control	✗	✓	✓	✓	✓

✓ : Yes

✗ : No

★ : Yes, not set by default

<http://www.cis.rit.edu/class/simg211/unixintro/Shell.html>

- When you login to a *NIX computer, shell scripts are automatically loaded depending on your default **shell**
- **sh,ksh**
 - ① `/etc/profile`
 - ② `$HOME/.profile`
- **bash**
 - ① `/etc/profile`, login terminal only
 - ② `/etc/bashrc` or `/etc/bash/bashrc`
 - ③ `$HOME/.bash_profile`, login terminal only
 - ④ `$HOME/.bashrc`
- **csch,tcsh**
 - ① `/etc/csh.cshrc`
 - ② `$HOME/.tcshrc`
 - ③ `$HOME/.cshrc` if `.tcshrc` is not present
- The `.bashrc`, `.tcshrc`, `.cshrc`, `.bash_profile` are script files where users can define their own aliases, environment variables, modify paths etc.

Examples I

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions
alias c="clear"
alias rm="/bin/rm -i"
alias psu="ps -u apacheco"
alias em="emacs -nw"
alias ll="ls -lF"
alias la="ls -al"
export PATH=/home/apacheco/bin:${PATH}
export g09root=/home/apacheco/Software/Gaussian09
export GAUSS_SCRDIR=/home/apacheco/Software/scratch
source $g09root/g09/bsd/g09.profile

export TEXINPUTS=./usr/share/texmf//:/home/apacheco/LaTeX//:${TEXINPUTS}
export BIBINPUTS=./home/apacheco/TeX//:${BIBINPUTS}
```

Examples II

```
# .tcshrc

# User specific aliases and functions
alias c clear
alias rm "/bin/rm -i"
alias psu "ps -u apacheco"
alias em "emacs -nw"
alias ll "ls -lF"
alias la "ls -al"
setenv PATH "/home/apacheco/bin:${PATH}"
setenv g09root "/home/apacheco/Software/Gaussian09"
setenv GAUSS_SCRDIR "/home/apacheco/Software/scratch"
source $g09root/g09/bsd/g09.login

setenv TEXINPUTS " ../usr/share/texmf//:/home/apacheco/LaTeX//:${TEXINPUTS}"
setenv BIBINPUTS " ../home/apacheco/TeX//:${BIBINPUTS}"
```

Files and Processes

- Everything in Linux/UNIX is either a file or a process
- A File is a collection of data, created by users using text editors, running compilers, etc.
- Examples of Files:
 - ❶ document such as collection of ascii text as in report, essay, etc.
 - ❷ program written in some high level programming language
 - ❸ instructions comprehensible to machine but not a casual user such as executable, binary file
 - ❹ directory containing information about its contents such as subdirectories or other files
- A process is an executing program identified by a unique process identifier or PID.

File System Hierarchy



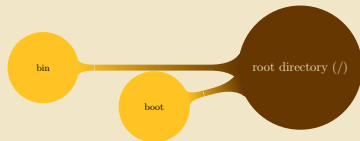
- All files are arranged in a hierarchical structure, like an inverted tree.
- The top of the hierarchy is traditionally called **root** (written as a slash /)

File System Hierarchy



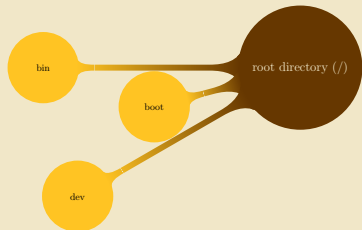
- contains files that are essential for system operation, available for use by all users.

File System Hierarchy



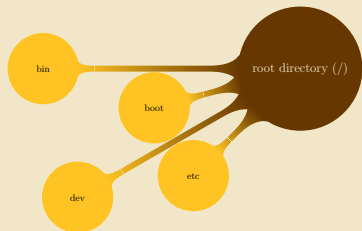
- contains bootable kernel and bootloader

File System Hierarchy



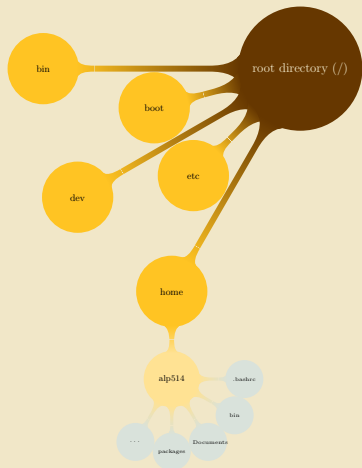
- contains various devices such as hard disk, CD-ROM drive etc

File System Hierarchy



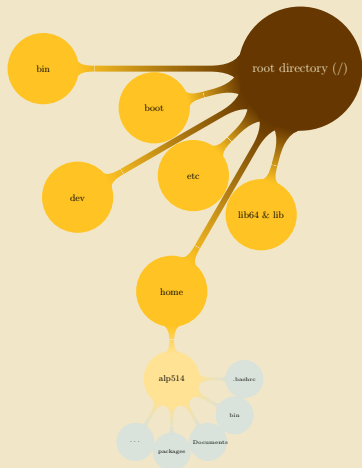
- contains various system configurations

File System Hierarchy



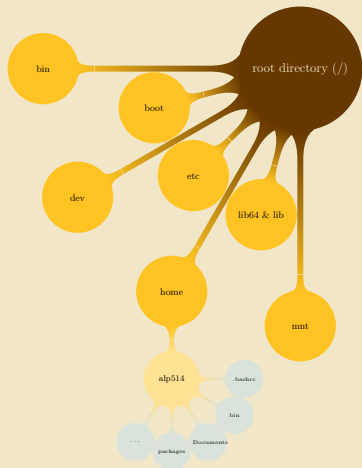
- contains home directories of all users. This is the directory where you are at when you login to a Linux/UNIX system.

File System Hierarchy

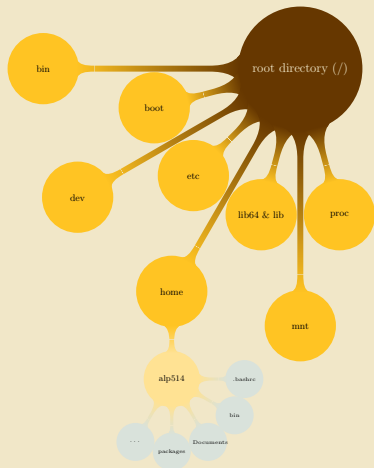


- contains libraries that are essential for system operation, available for use by all users.

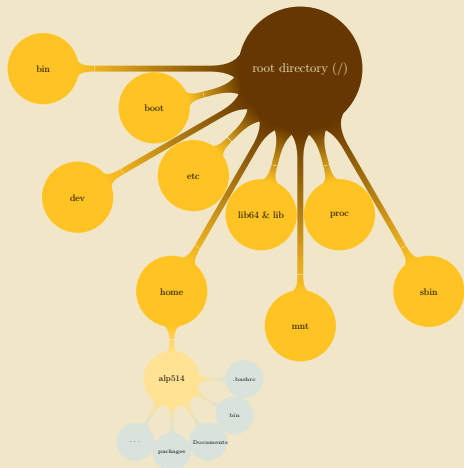
File System Hierarchy



File System Hierarchy

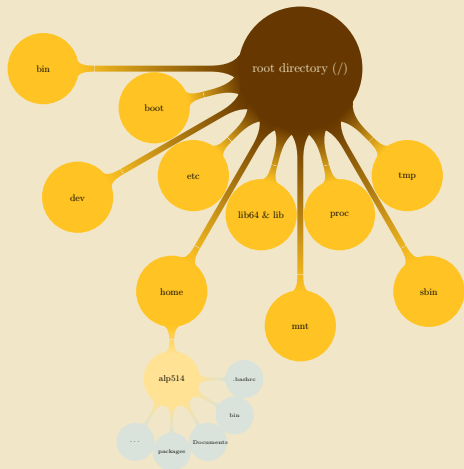


File System Hierarchy



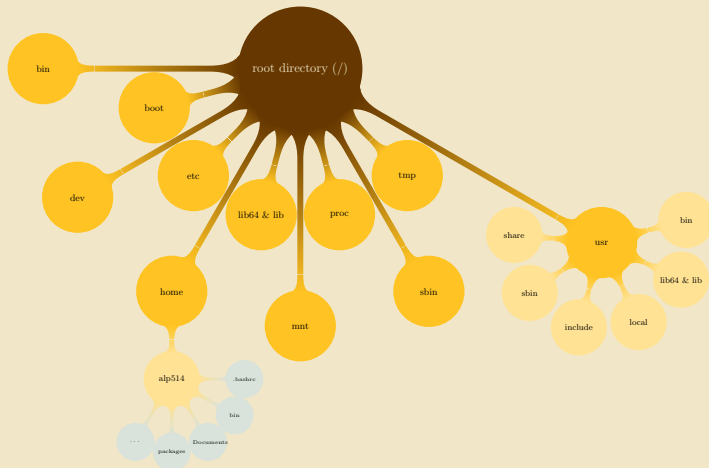
- same as bin but only accessible by **root**

File System Hierarchy



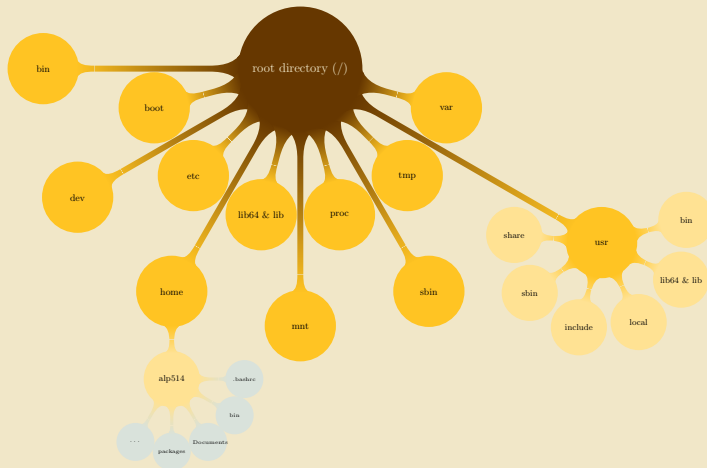
- temporary file storage

File System Hierarchy



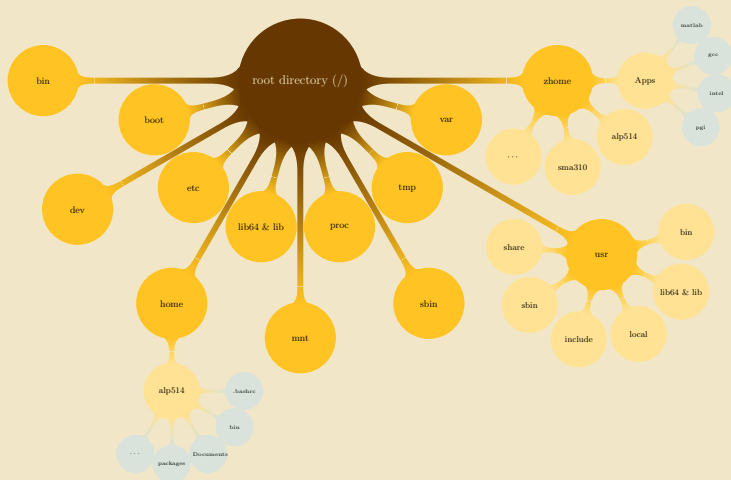
- contains user documentations, binaries, libraries etc

File System Hierarchy



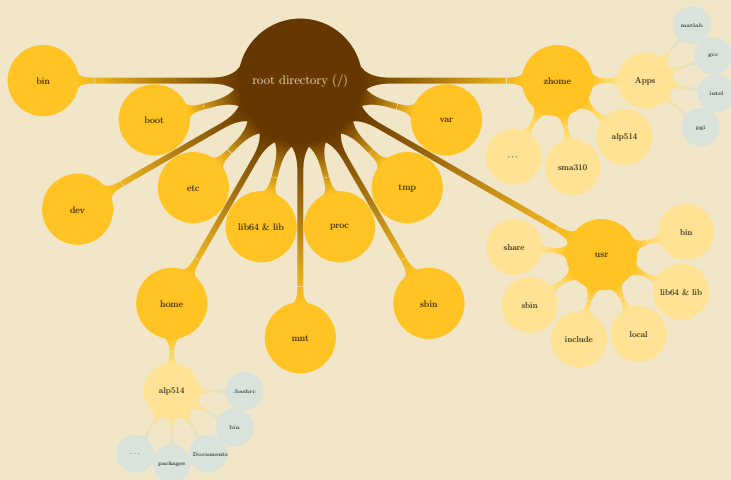
- used to store files which change frequently (system level not user level)

File System Hierarchy



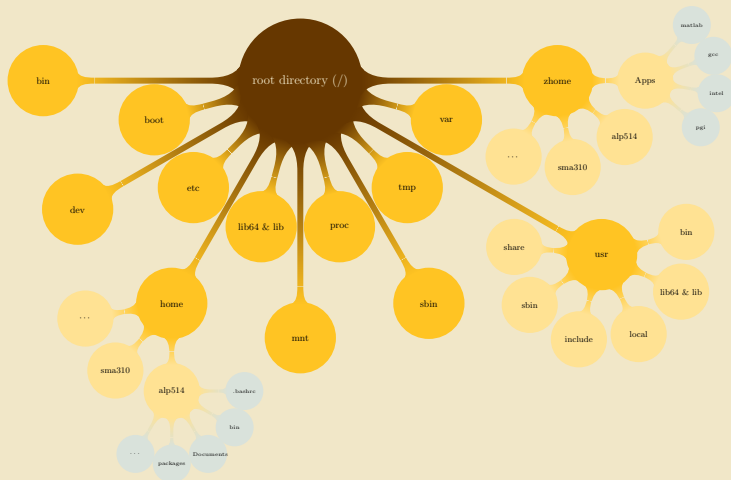
- where we install applications common to all HPC systems

File System Hierarchy

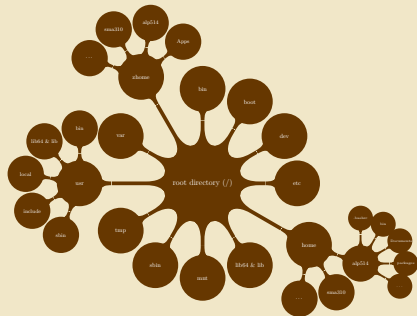


- Installing your own OS: `/bin`, `/lib{64}`, `/etc`, `/dev` and `/sbin` must be on the same partition.

File System Hierarchy

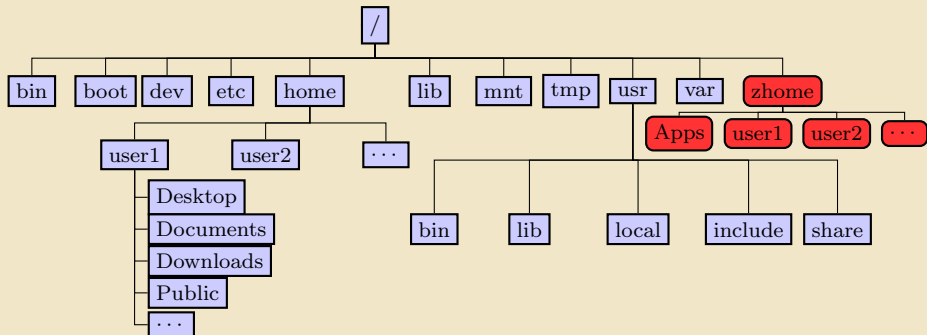


- UNIX like OS's are designed for multi user environments i.e. multiple users can exist on the system.



Directory Structure

- All files are arranged in a hierarchical structure, like an inverted tree.
- The top of the hierarchy is traditionally called **root** (written as a slash /)



Important Directories

`/bin`: contains files that are essential for system operation, available for use by all users.

`/lib`,`/lib64`: contains libraries that are essential for system operation, available for use by all users.

`/var`: used to store files which change frequently (system level not user level)

`/etc`: contains various system configurations

`/dev`: contains various devices such as hard disk, CD-ROM drive etc

`/sbin`: same as `bin` but only accessible by **root**

`/tmp`: temporary file storage


`/boot`: contains bootable kernel and bootloader

`/usr`: contains user documentations, binaries, libraries etc

`/home`: contains home directories of all users. This is the directory where you are at when you login to a Linux/UNIX system.

- Installing your own OS: `/bin`,`/lib{64}`,`/etc`,`/dev` and `/sbin` must be on the same partition.

- UNIX like OS's are designed for multi user environments i.e. multiple users can exist on the system.
- Special user called **root** is the administrator and has access to all files in the system.
- In *nix, users are organized into groups.
- Each user is in atleast one group.
- Group membership makes it easier to share files with members of your group.

Type **groups**  to find your group membership.

- All files are *case sensitive*,
- ◆ myfile.txt, Myfile.txt and myfile.TXT are three different files and can exist in the same directory simultaneously.

Relative & Absolute Path

- *Path* means a position in the directory tree.
- You can use either the *relative path* or *absolute path*
- In *relative path* expression
 - . (one dot or period) is the current working directory
 - .. (two dots or periods) is one directory up
 - You can combine . and .. to navigate the file system hierarchy.
 - the path is not defined uniquely and does depend on the current path.
 - ../../tmp is unique only if your current working directory is your home directory.
- In *absolute path* expression
 - the path is defined uniquely and does not depend on the current path
 - /tmp is unique since /tmp is the *absolute path*

Variables

Variables I

- *nix also permits the use of variables, similar to any programming language such as C, C++, Fortran etc
- A variable is a named object that contains data used by one or more applications.
- There are two types of variables, Environment and User Defined and can contain a number, character or a string of characters.
- Environment Variables provides a simple way to share configuration settings between multiple applications and processes in Linux.
- By Convention, environmental variables are often named using all uppercase letters

e.g. `PATH`, `LD_LIBRARY_PATH`, `LD_INCLUDE_PATH`, `TEXINPUTS`, etc

- To reference a variable (environment or user defined) prepend `$` to the name of the variable

e.g. `$PATH`, `$LD_LIBRARY_PATH`

Variables II

- The command `printenv` list the current environmental variables.
- ★ Type `printenv` on your command prompt to list all environment variables in your current session.
- The command `env` is used to either print a list of environment variables or run another utility in an altered environment without having to modify the currently existing environment.
- ★ Type `env SHELL=/bin/tcsh xterm` to start an xterm session in tcsh
- ◆ To execute the above command successfully, you need to be in GUI mode on the virtual OS or logged into a remote systems with X-Forwarding enabled.

Variables III

PATH: A list of directory paths.

HOME: indicate where a user's home directory is located in the file system.

PWD: contains path to current working directory.

OLDPWD: contains path to previous working directory.

TERM: specifies the type of computer terminal or terminal emulator being used

SHELL: contains name of the running, interactive shell.

PS1: default command prompt

PS2: secondary command prompt

LD_LIBRARY_PATH: colon-separated set of directories where libraries should be searched for first

HOSTNAME: The systems host name

USER: Current logged in user's name

DISPLAY: Network name of the X11 display to connect to, if available.

Variables IV

- You can edit the environment variables.
- Command to do this depends on the shell
- ★ To add your bin directory to the **PATH** variable
 - sh/ksh/bash: `export PATH=${HOME}/bin:${PATH}`
 - csh/tcsh: `setenv PATH ${HOME}/bin:${PATH}`
- ★ Note the syntax for the above commands
- ★ sh/ksh/bash: no spaces except between **export** and **PATH**
- ★ csh,tcsh: no = sign, just a space between **PATH** and the absolute path
- ★ all shells: colon(:) to separate different paths and the variable that is appended to
- **Yes, the order matters.** If you have a customized version of a software say perl in your home directory, if you append the perl path to **PATH** at the end, your program will use the system wide perl not your locally installed version.

Variables V

- Rules for Variable Names

- ① Variable names must start with a letter or underscore
- ② Number can be used anywhere else
- ③ DO NOT USE special characters such as @, #, %, \$
- ④ Case sensitive
- ⑤ Examples
 - Allowed: VARIABLE, VAR1234able, var_name, _VAR
 - Not Allowed: 1VARIABLE, %NAME, \$myvar, VAR@NAME

- Assigning value to a variable

Type	sh,ksh,bash	csh,tcsh
Shell	name=value	set name = value
Environment	export name=value	setenv name value

- **sh,ksh,bash** THERE IS NO SPACE ON EITHER SIDE OF =
- **csh,tcsh** space on either side of = is allowed for the **set** command
- **csh,tcsh** There is no = in the **setenv** command

Exercise

- Create two shell variables containing
 - ① your name
e.g. MYNAME=Alex
 - ② a standard greeting
e.g. Greet=Hello
- We'll make use of these variables in a few slides when we learn some basic commands.



Basic Commands

Basic Commands




What is a command and how do you use it?

- **command** is a directive to a computer program acting as an interpreter of some kind, in order to perform a specific task.
 - **command prompt** (or just **prompt**) is a sequence of (one or more) characters used in a command-line interface to indicate readiness to accept commands.
 - Its intent is to literally prompt the user to take action.
 - A prompt usually ends with one of the characters \$, %, #, :, > and often includes other information, such as the path of the current working directory.
- ★ Virtual Image: `[user@localhost ~]$`
- ★ Mac OSX in `tcsh`: `[c8-bc-c8-ee-b8-9e:~] apacheco%`
- Each **command** consists of three parts: name, options, arguments
- `[user@localhost ~]$ command options arguments`

How to get more information with Linux

- `man` shows the manual for a command or program.
- The manual is a file that shows you how to use the command and list the different options for the command in question.
- Usage: `man [command]`
- Example: `man ls` 
- `apropos` shows you all of the man pages that may shed some light on a certain command.
- Usage: `apropos [keyword]`
- Example: `apropos editor` 

Input & Output Commands I

- The basis I/O statements are **echo** for displaying output to screen and **read** for reading input from screen/keyboard/prompt
- The **read** statement takes all characters typed until the  key is pressed and stores them into a variable.
- Usage: **read** <variable name>
- Example: **read** name 
Alex Pacheco 
- In the above example, the name that you enter is stored in the variable **name**.
- The **echo arguments** command will print **arguments** to screen or standard output.
- **arguments** can be a (single or multiple) variable, string of characters or numbers.

Input & Output Commands II

- Examples:

- ① `echo $LD_LIBRARY_PATH $LD_INCLUDE_PATH` 

- ② `echo Welcome to HPC Training` 

- By default, `echo` eliminates redundant whitespace (multiple spaces and tabs) and replaces it with a single whitespace between arguments.

- To include redundant whitespace, enclose the arguments within double quotes

e.g. `echo "Welcome to HPC Training"` 

Input & Output Commands III

Exercise

- Print out the variable you created a few slides back

```
echo $MYNAME 
```

```
echo $Greet 
```

- Read a variable for greeting message

```
read message 
```

```
Welcome to HPC 
```





- Combine and print your name, the greeting and the message

```
echo $Greet $MYNAME $message 
```


- What is the output of the following command?

```
echo $Greet $MYNAME, $message Training 
```

Commands: pwd & cd

- `pwd` command prints the current working directory.
- Usage: `pwd`
- Example: `pwd` 
- `cd` command allows one to change directory
- argument is the path (relative or absolute) of the directory you want to change to
- Usage: `cd [destination]`
- Example: `cd /tmp` 
- The default destination directory is your home directory.
- i.e. If you type `cd`  , you will end up in your home directory.
- If you want to go back to the previous directory, type `cd -` 

Command: ls

- `ls` command lists the contents of a directory.
- Usage: `ls <options> <path>`
- Example: `ls` 
- The current working directory is the default path.
- To list contents of another directory specify the path, relative or absolute
- Common options to the `ls` command
 - l: show long listing format
 - a: show hidden files
 - r: reverse order while sorting
 - t: show modification times
 - h: use file sizes in SI units (bytes, kilobytes, megabytes etc) default is bytes

Command: alias

- **alias** is a command to create a shortcut to another command or name to execute a long string.

- Usage

```
bash/sh/ksh: alias <name>="<actual command>"
```

```
csh/tcsh: alias <name> "<actual command>"
```


- Example:

```
bash/sh/ksh: alias lla="ls -al"
```

```
csh/tcsh: alias lls "ls -al"
```

- The **alias** command is very useful tool to create shortcuts to other commands and is most often used by paranoid users to prevent accidental deletion of files.
- **unalias** is a command to remove an alias.
- Usage: **unalias** <name>
- Example: **unalias lla** will remove the shortcut to **ls -al**

Command: mkdir


- `mkdir` is a command to create a directory
- Usage: `mkdir <options> <directoryname>`
- Example: `mkdir -p $HOME/test/testagain` 
- By default, the directory is created in the current directory or in a path relative to the current directory
- The `-p` option will create intermediate directories if they do not exist.


e.g. If the directory `test` does not exist in `$HOME`, then

`mkdir $HOME/test/testagain` will fail.

The `-p` option will create the `test` directory within `$HOME` and then create `testagain` within the newly created `test` directory

Command: cp


- `cp` is a command to copy a file or directory
- Usage: `cp <options> <source(s)> <destination>`
- Example: `cp $HOME/.bashrc ../../tmp` 
- Common options to `cp` command:
 - `r`: copy recursively, required when copying directories.
 - `i`: prompt if file exists on destination and can be copied over.
 - `p`: preserve file access times, ownership etc.
- If there are more than one source files, then the destination (i.e. last entry or file) must be a directory.
- If the source(s) is(are) a file(s) and the destination is a directory, then the file(s) will be copied into the directory

e.g. `cp file1 file2 dir1` 

`dir1` will contain the files `file1` and `file2`

If `dir1` is a file, then the above command will fail

Command: rm


- `rm` command removes or deletes a file or directory
- Usage: `rm <options> <file or directory>`
- Example: `rm $HOME/tmpfile` 
- Common options to `rm` command:
 - `r`: remove recursively, required when copying directories.
 - `i`: prompt if file really needs to be deleted
 - `f`: force remove overrides the `-i` option
- **BE CAREFUL WHILE USING THE `rm` COMMAND, DELETED FILES CANNOT BE RECOVERED**
- To be on the safe side, create an `alias` to the `rm` command and only use the `-f` option only if you are sure you want to delete the file or directory

```
sh/ksh/bash: alias rm="rm -i"

csh/tcsh: alias rm 'rm -i'
```
- delete empty directories using the `rmdir` command.

Command: mv

- `mv` command moves or renames a file or directory
- Usage: `mv <options> <source> <destination>`
- Example: `mv test test1`
- If there are more than one source file, then the last file is the destination and must be a directory.
- Use the `-i` option to prompt if a file or directory will be overwritten.
- If the source(s) is(are) a file(s) and the destination is a directory, then the file(s) will be copied into the directory.

e.g. `mv file1 file2 dir1` 

`dir1` will contain the files `file1` and `file2`

If `dir1` is a file, then the above command will fail

Pager Commands

- To display a file to screen, *nix provides three commands at your disposal
- **cat**: Show contents of a file.
- **more**: Display contents one page at a time.
- **less**: Display contents one page at a time but allow forward/backward scrolling
`less > more or less is more, more or less`
- Usage: `cat/more/less <options> <filename>`
- Example: `cat .bashrc`
- To scroll forward in **more** or **less**, use the space bar, **CNTRL-f/d** or "Page Down" key.
- To scroll backwards in **less** use **CNTRL-b/u** or "Page Up".
- A rarely used command, **tac** does the opposite of **cat** i.e. show contents of a file in reverse.

Other Commands I

passwd: change password

chsh: change default shell

df: report disk space usage by filesystem

du: estimate file space usage - space used under a particular directory or files on a file system.

sudo: run command as root (**only if you have access**)

mount: mount file system (**root only**)

umount: unmount file system (**root only**)

shutdown: reboot or turn off machine (**root only**)

top: Produces an ordered list of running processes

free: Display amount of free and used memory in the system

file: Determine file type

touch: change file timestamps or create file if not present

date: display or set date and time

find : Find a file

```
find /dir/to/search -name file-to-search
```

Other Commands II

wc: Count words, lines and characters in a file

```
wc -l .bashrc
```

grep: Find patterns in a file

```
grep alias .bashrc
```

awk: File processing and report generating

```
awk '{print $1}' file1
```

sed: Stream Editor

```
sed 's/home/HOME/g' .bashrc
```

set: manipulate environment variables

```
set -o emacs
```

ln: Link a file to another file

```
ln -s file1 file2
```

wait: wait until all backgrounded jobs have completed


which: shows the full path of (shell) commands

whatis: display manual page descriptions

Other Commands III


!name: rerun previously executed command with the same arguments as before, **name <args>**.

Note that you do not always have to type the full command **name**, just the minimum unique characters (no spaces) of **name** need to be entered.

If you had entered two commands **name <args>** and **nbme <args>**, then to rerun **name**, use the command **!na**  .


history: display a list of last executed commands. Optional argument **m** will list the last **m** commands.


All previously executed commands will be listed with a number **n**.


To rerun a command from history which has number **n**, run the command **!n** .


To learn more about these commands, type **man command** on the command prompt


- Login to a Linux machine and open a terminal
- Enter the following commands or carry out operations asked for.
- Understand what you are doing and ask for help if unsure. Some commands are incorrect or will fail, enter the correct


❶ echo hello world 

❷ pwd 


❸ whoami 

❹ cd /tmp 

❺ cd - 

❻ mkdir test/testagain 

❼ cd test/testagain 

❽ touch file 

❾ Go back to your home directory.

❿ Which shell are you using?

⓫ Review the commands you have just entered.

⓬ create an alias for removing files which prompt for confirmation and delete the file that you created.

⓭ From your home directory get a list of files and directory in long format in reverse order with file sizes listed in human readable format.

Exercises II

- 14 Find out the location of vi, emacs, firefox, google-chrome, thunderbird, latex, pdflatex, gnuplot, python, perl and matlab.