Lec-01-2
Linux Commands

Dr Syed Faisal Hasan and Dr. Hymie Latif
Computing and Information Technology
College of Enterprise and Development
Otago Polytechnic
Dunedin, New Zealand

Bachelor of Information Technology
IN616 – Operating Systems Concepts
Semester 1, 2020

Schedule

- Recap
- History
- BASH, Shells, Terminals
- Linux commands
- Linux file system structure
- Linux help system
- Metacharacters
- Environment variables
- User privilege

Linux: Market Share

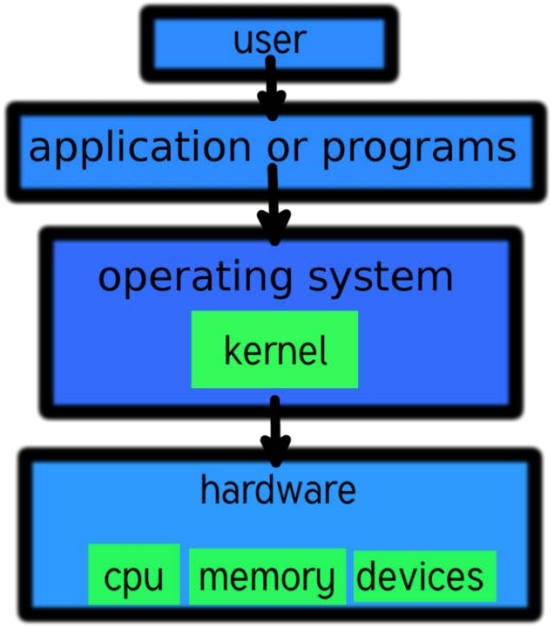
- **Mobile:**
  - Android with 71%
  - iOS is 26%
- **Desktop:**
  - Linux with 1.6%
  - Windows is 89% and Mac is 9.5%
  - www.netmarketshare.com
- **Web servers**
  - Linus accounts for 36% of all webservers
  - http://stackoverflow.com/research/developer-survey-2016#technology-desktop-operating-system
  - All UNIX-like account for 67% of all webservers
  - http://w3techs.com/technologies/overview/operating_system/all
- **Supercomputers:**
  - Linux with 99.4%
  - http://www.top500.org/statistics/overtime

Why is Linux not Popular in Desktop?

- **Technical challenges**
  - Let's hear what Linus says https://www.youtube.com/watch?v=ZPUk1yNVeEI
- **Other Challenges**
  - Business
  - Applications
  - Service Level Agreements

Linux: **Kernel** or OS or Distribution

- **Linux**
- Only kernel without software ecosystem
- Core functionality:
  - Architecture dependent code (32/64-bit)
  - Interface for drivers (not actual drivers)
  - System call hooks for applications
  - Task scheduling
  - Memory management
  - Network functionality



Linux: Kernel or OS or **Distribution**

- **Distribution**
  - Linux Kernel + GNU Utilities (cp, mv, ls, bash) + code compilers + editors + applications
  - Focuses on purpose of distribution
  - www.distrowatch.com

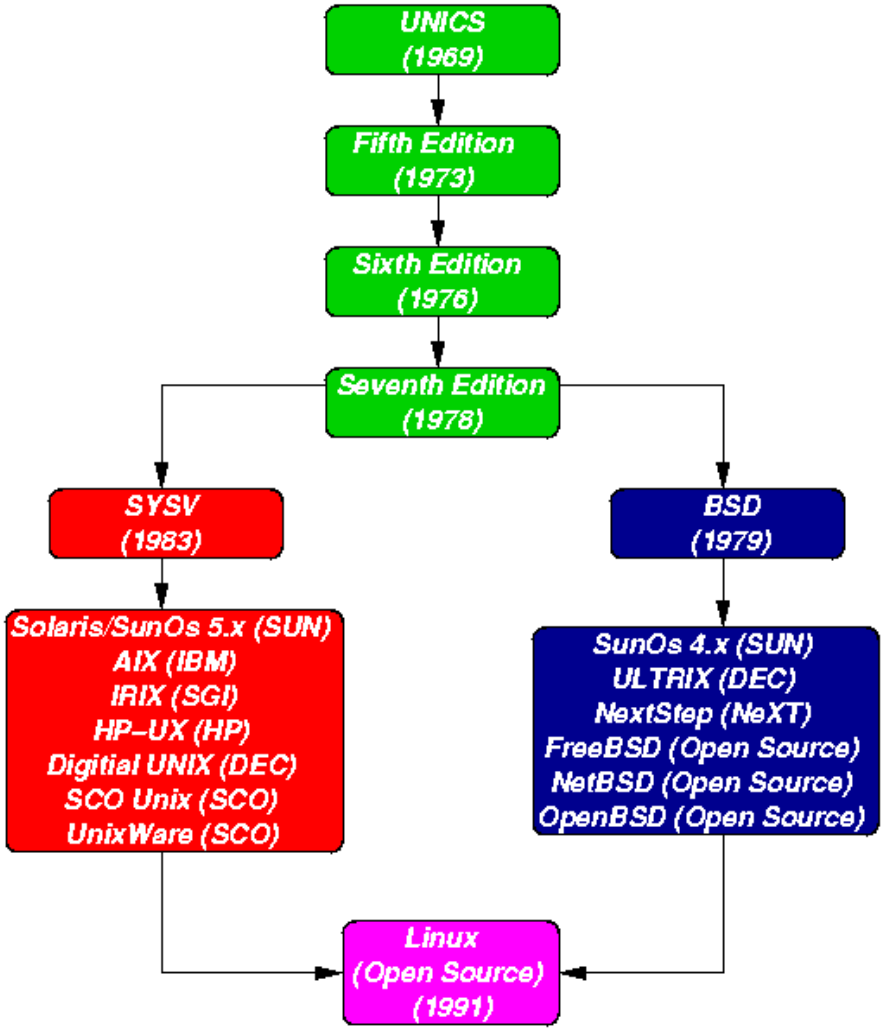
TOPIC:

History

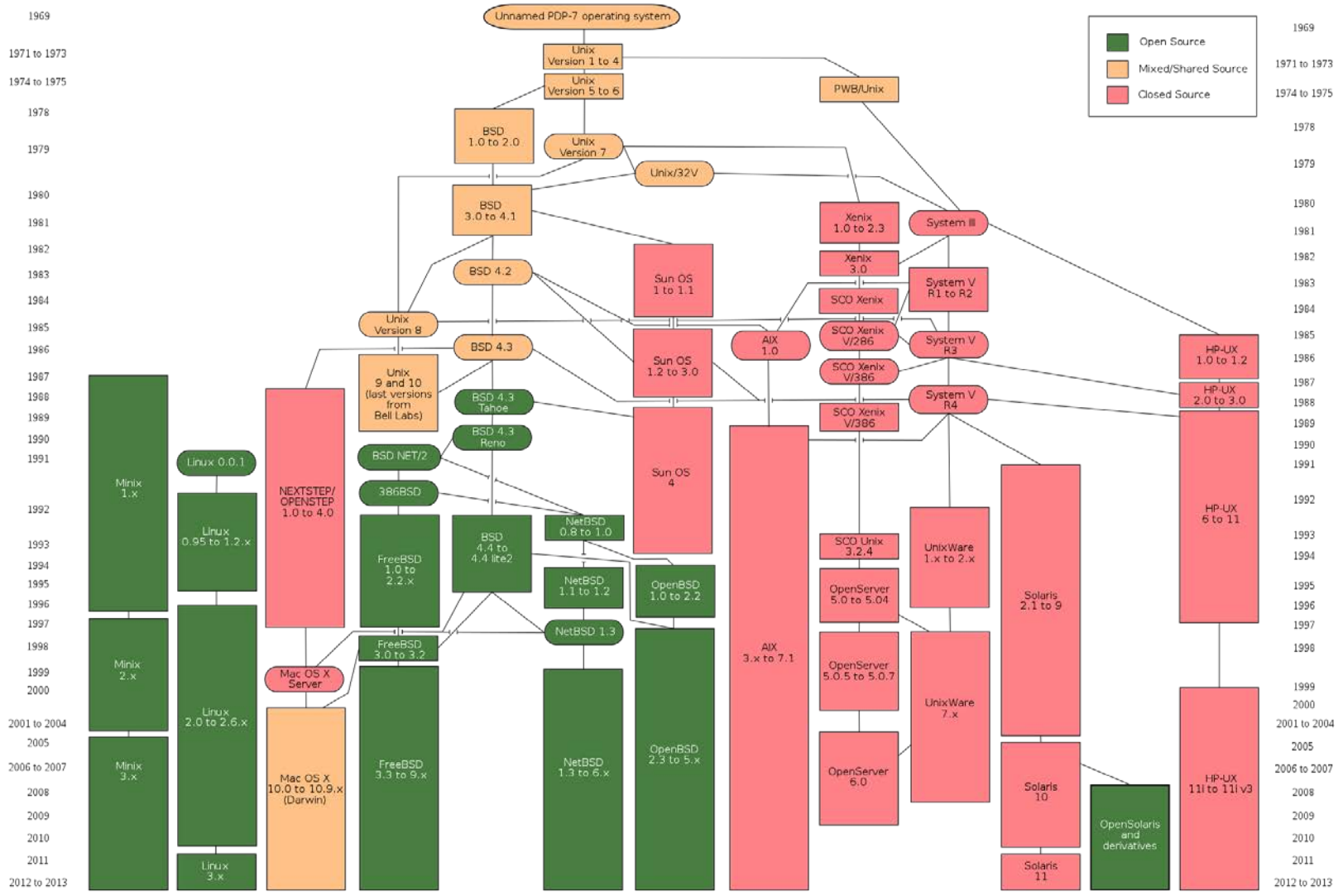
From UNICS (UNIX) to Linux

- **AT&T developed UNIX in 1969**
- **1970s**
  - Ken Thompson & Dennis Ritchie (inventors of C) joined the development of UNICS
  - UNICS = Uniplexed Information and Computing System
  - UNICS was later known as UNIX
- **Objectives**
  - Portability
    - Separation of hardware and software
    - Developed in C with minimal amount of machine code
  - Higher efficiency
    - Limiting memory consumption of OS itself
    - Short commands: ls, man, cp, mv, etc.
- **Split into two branches:**
  - System 5 (AT&T and others) → commercial
  - Berkeley Software Distribution (BSD) (University of California) → academic

From UNICS (UNIX) to Linux



From UNICS (UNIX) to Linux



Linux: And GNU project

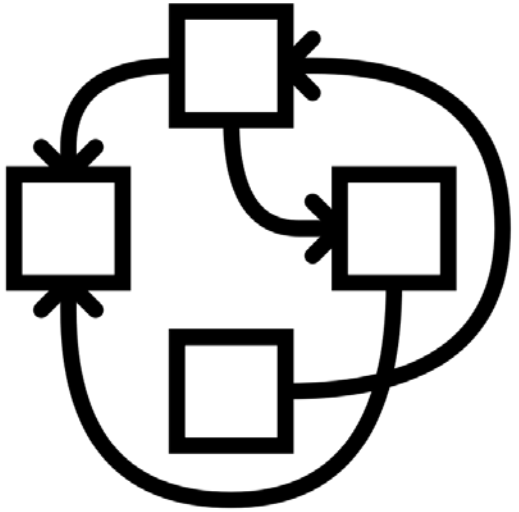
- **GNU: *GNU's not UNIX***

- **Richard Stallman**
  - Has promoted open source software since 1984
  - Founder of GNU Foundation
  - The author of the GNU General Public License (GPL)
  - Creator of Emacs

- **GNU Project**
  - Collection of open source UNIX-compatible software
  - User-level applications
    - http://gnu.uberglobalmirror.com/
    - https://en.wikipedia.org/wiki/List_of_GNU_packages
- Torvalds adopted GPL in 1992 – Linux was de facto GNU kernel

Linux: And GNU project

- **GNU is missing a kernel !!**
- **GNU Hurd**
  - Been under development since 1990
  - Designed to be a replacement for the UNIX kernel
- **Linux + GNU**
  - Most Linux *distributions* use GNU coreutils
  - Commonly known as GNU/Linux
    - http://gnu.uberglobalmirror.com/
    - https://en.wikipedia.org/wiki/List_of_GNU_packages



TOPIC:

Introduction to BASH

Shells ???

- **Shell**
  - User interface
  - Provides ability to interact with Linux kernel

- **Provides a command line interface**

- **BASH**
  - A type of shell
  - **B**ourne **A**gain **SH**ell
  - No… not that dude →

BASH

- **Default shell used in most Linux systems**
- **Different users can have different shells**
  - By installing/specifying multiple shells
- **Same user can have multiple shells**
  - By logging on multiple times

Other Shells ???

- **Alternative shells?**
  - Tcsh
  - Sch
  - Ksh
  - Zsh
  - Fish

- **Short overview of other shells**
- https://www.tecmint.com/different-types-of-linux-shells/

Terminals

- Device which user connects to the shell

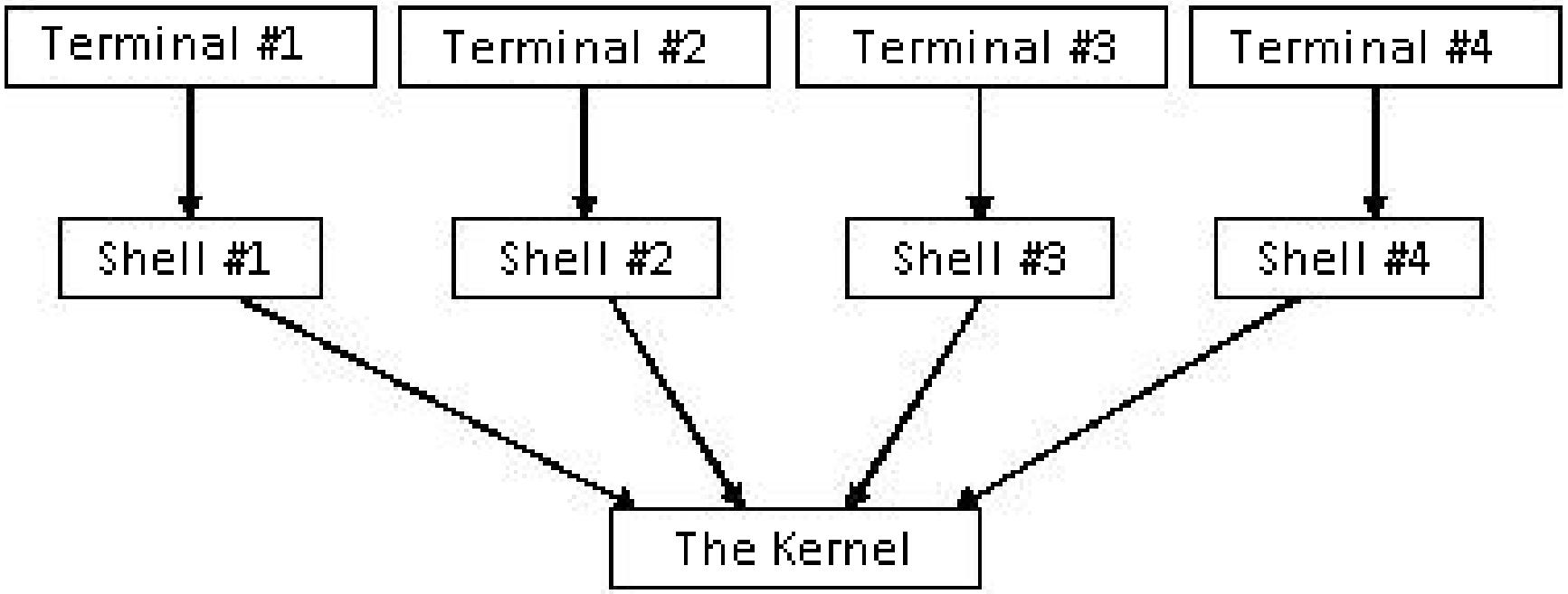
- **Can be local**
- We can use multiple local at the same time
  - Alt + F1 → tty1
  - Alt + F2 → tty2
  - … (F1 → F6)
  - Alt + F7 → tty7 (this is the GUI – but we do not have one ☹)

- **Can be remote** – anyone know how?

Terminals

```
Ubuntu 16.04.2 LTS opstudent-host tty1

opstudent-host login: user
Password:
Last login: Tue Jul 18 08:17:13 NZST 2017 on tty1
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-62-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

120 packages can be updated.
57 updates are security updates.


user@opstudent-host:~$
```

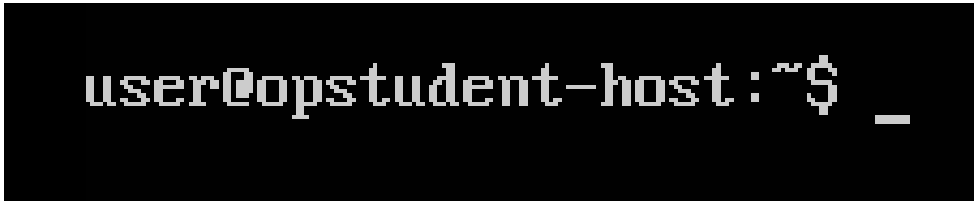
Terminals


Terminal #1
Terminal #2
Terminal #3
Terminal #4
Shell #1
Shell #2
Shell #3
Shell #4
The Kernel

Quick Summary

- **Shell**
  - Another word for a User Interface of an operating system
  - It is software-based
- **BASH**
  - A particular type of command line shell
- **Terminal**
  - Text-based input/output environment
  - Technically a *device,* but can be software
  - Also can be emulated

Terminal Structure



- Format: `username@hostname:directory$`
- username = `user`
- hostname = `opstudent-host`
- Present working directory: `~` (home dir)

- $ at end → normal user account
- # at end → super user account (not seen much)

Summarise these:

```
root@opstudent-host:/etc/network#
```

```
user@opstudent-host:/boot/grub$
```

Username?
Hostname?
Working directory?
Privilege?

TOPIC:

Linux Commands

Commands

- Generic command syntax:

  <command> <options> <arguments>

- Options are not mandatory
- Arguments are not mandatory
- Examples:

  ls
  ifconfig
  pwd
  hostname
  cat

Commands (ls): Options

- Generic command syntax:

  <command> <options> <arguments>

- Options modify command behavior
  - Prefixed with a dash symbol: **-**
  - For example: **-i**
  - Full-word options with double dash: **--**
  - For example: **--inode**

- Examples:

```
ls -i
ls --inode
```

Commands (ls): Arguments

- Generic command syntax:

  <command> <options> <arguments>

- Arguments are input to command

- No argument lists current working directory:

  ls

- Supplied argument lists specified directory:

  ls /etc

- What arguments do depend on the command

Commands (ls): Full example

- Generic command syntax:

  <command> <options> <arguments>

- Examples:

  ls –i /etc/ssh/

  ls --inode /etc/ssh/

Command Examples

- **`ls -l /home/user`**
  - List files in home directory for user account
- **`cat file.txt`**
  - Display file.txt on standard output
- **`mkdir -p sales/june`**
  - Make a directory (june) and parent directory (sales)
- **`touch tuesday.log`**
  - Create file/update timestamp on file
- **`clear`**
  - Clear the screen (standard output)

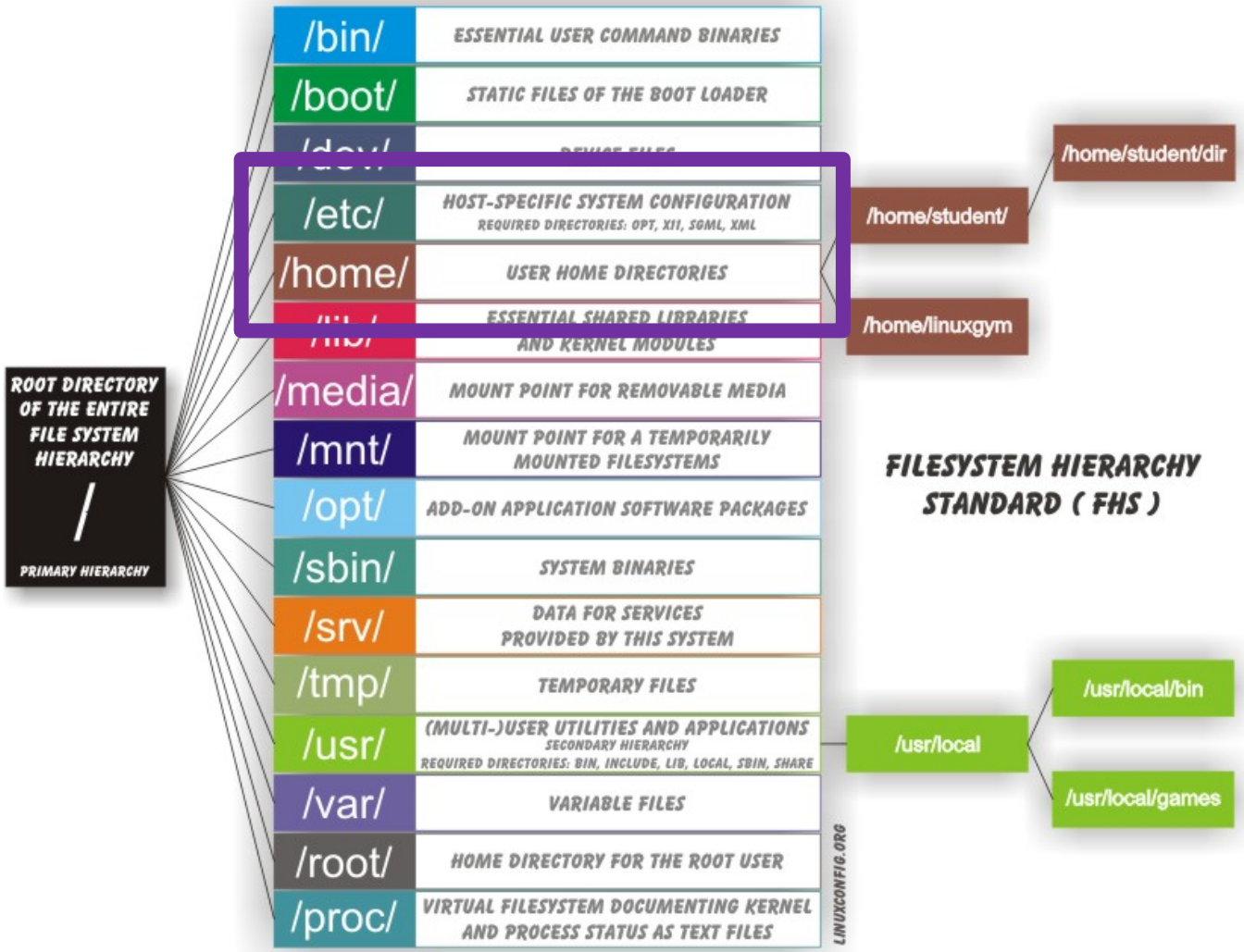
File System Hierarchy Principles

- **Difference to Microsoft Windows?**
- **There is no C:**
- **Instead, Linux has a single root directory → /**

```
/ == C:\
```

- **Linux file systems**
  - Tree structure (like all file systems)
  - Top level directories have a specific purpose
  - Relatively consistent across all Linux distributions

Linux File System



Linux File System: Resources

- **Good Introduction to File System Hierarchy in Linux:**
- Linux compared to Windows
  - http://codeidol.com/community/nix/understanding-the-linux-file-system/5302/

- **Full specification reference:**
- Food to find out purpose of specific directories
  - http://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html

Absolute VS Relative paths

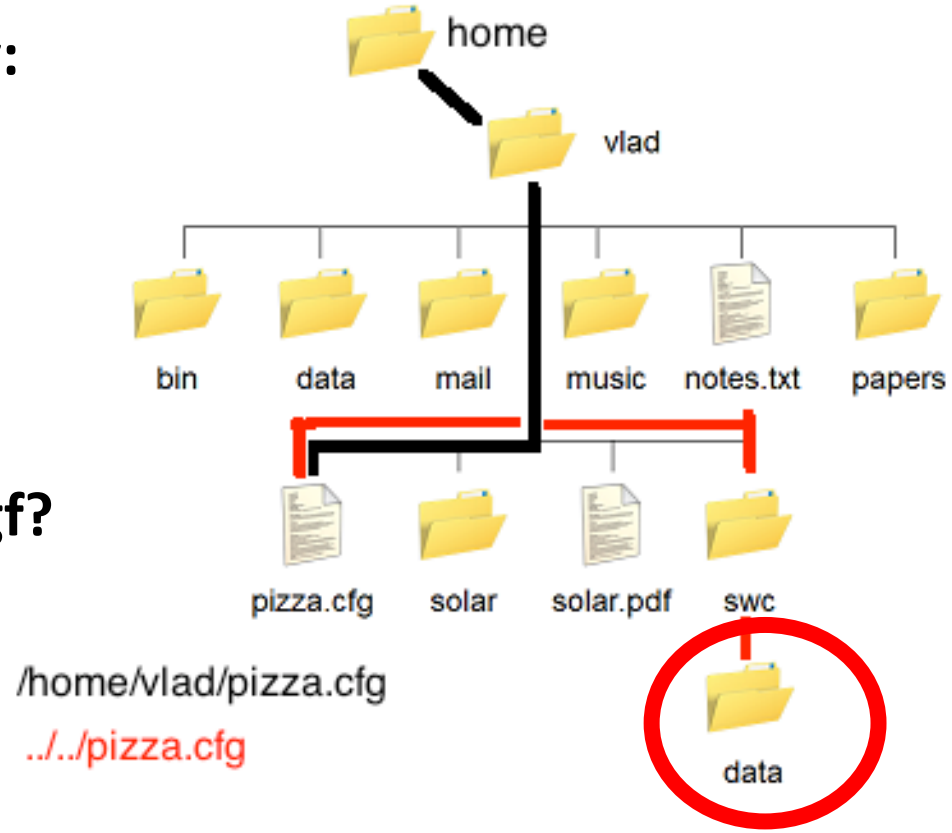
- **Absolute:**
- Always starts at the root directory
  - `C:\Users\student\Desktop\notes.txt`
  - `/home/student/Desktop/notes.txt`

- **Relative:**
- The path that is *relative* to the current directory
  - If we are in `C:\Programs Files`
  - And want to access `C:\Users\student\Desktop\notes.txt`
  - `..\Users\Tom\Desktop\notes.txt`

Absolute VS Relative paths

Current working directory:
/home/vlad/swc/data

From current directory…
How do we get to pizza.cgf?



/home/vlad/pizza.cfg

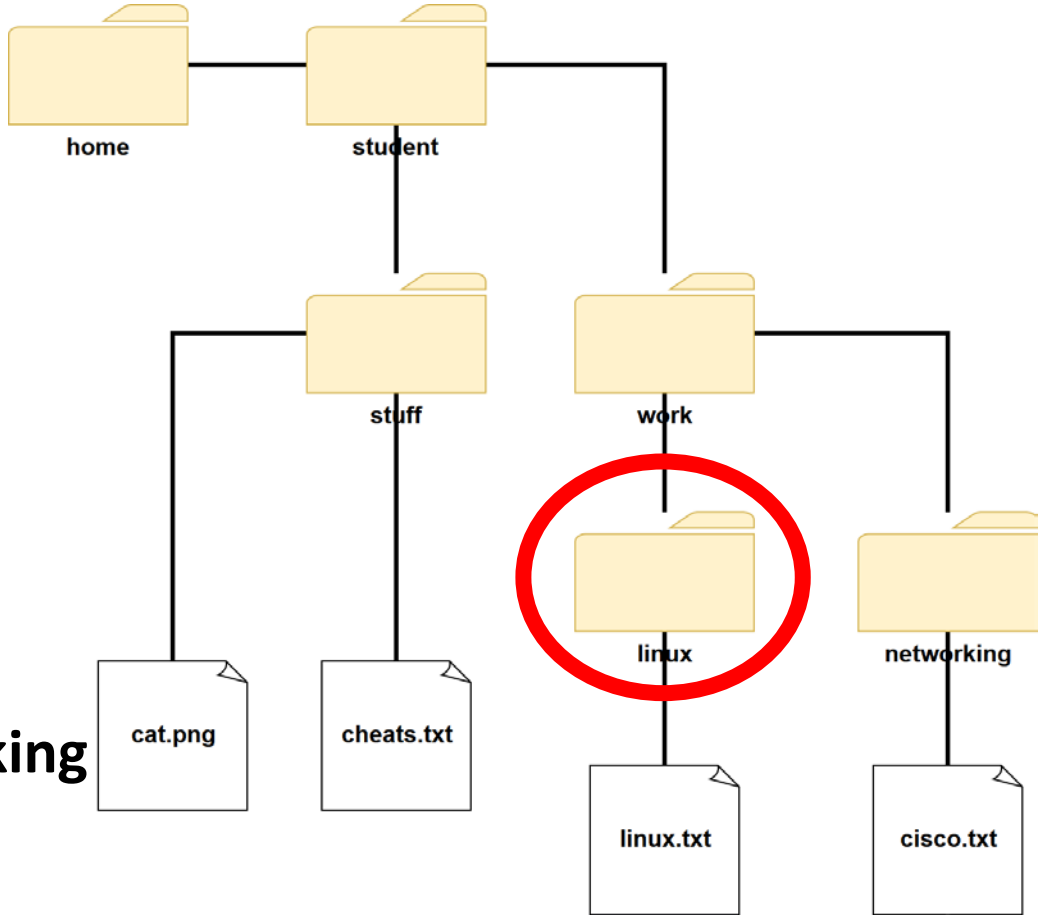
../../pizza.cfg

Absolute VS Relative Paths

Current working directory:
/home/student/work/linux

From current directory…
How do we get to networking?

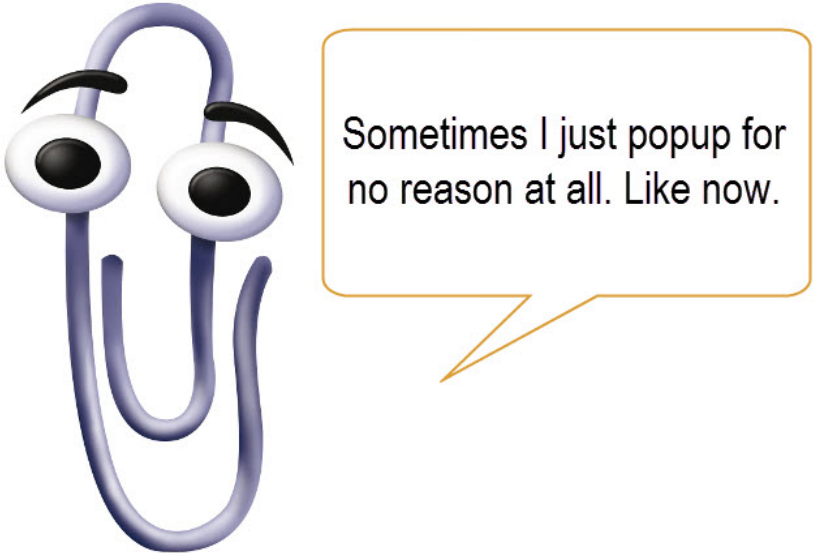
../networking
/home/student/work/networking



Linux Help System

Linux Help System

- **It is built into the machine (or online)**
- **`man <command>`**
  - man stands for manual
  - It is the system manual for command
  - Type **`q`** to exit the manual
- **`<command> --help`**
  - Not always consistent
  - Provides overview of arguments
  - **`ls --help`**
  - Encoded in program itself



- **You can also check manpages online: http://manpages.ubuntu.com/**

Man: organised by sections

- **Output of: `man man`**

```
1   Executable programs or shell commands
2   System calls (functions provided by the kernel)
3   Library calls (functions within program libraries)
4   Special files (usually found in /dev)
5   File formats and conventions eg /etc/passwd
6   Games
7   Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
8   System administration commands (usually only for root)
9   Kernel routines [Non standard]

A manual page consists of several sections.

Conventional section names include NAME, SYNOPSIS, CONFIGURATION, DESCRIPTION, OPTIONS,
EXIT STATUS, RETURN VALUE, ERRORS, ENVIRONMENT, FILES, VERSIONS, CONFORMING TO, NOTES,
BUGS, EXAMPLE, AUTHORS, and SEE ALSO.
```

- **`man ls`**

Home work: Realize Setup

- **vRealize setup:** https://fthvra01.op.ac.nz/vcac/
  - Check you have access (can log in)
- Request the *TrainingVM*
  - Catalog → IN616 TrainingVM (Request one!)

Lab_01_2 – Start

- **TOPICS:**
- **Browsing the Linux file system**
- **Using Bash commands**
- **Using the help system**

- **Complete up to section 4**
- **We will discuss more, before the next lab sections**

Metacharacters

- Special characters used in commands
- They have special meanings

- $ in BASH denotes a variable
  - $name
  - $age

- In this context, $ is a metacharacter

- Avoid metacharacters in command line arguments!

Metacharacters

Metacharacter(s)	Description
$	Shell variable
~	Special home directory variable
&	Background command execution
;	Command termination
< << > >>	Input/Output redirection
|	Command piping
* ? [ ]	Shell wildcards
' " \	Metacharacter quotes
` `	Command substitution
( ) { }	Command grouping

http://faculty.salina.k-state.edu/tim/unix_sg/shell/metachar.html

http://tldp.org/LDP/GNU-Linux-Tools-Summary/html/x11655.htm

Environment variables

- We can display variables using
  - `echo $var`
  - `echo $name`
- Run `env` to show all environment variables

Command	Meaning
$SHELL	Shell binary used in current terminal
$PATH	Path variable contains locations for commands
$HISTSIZE	Size of history
$TERM	Returns terminal type (limited use nowadays)

- Add you own variables into `.bashrc`

Using metacharacters (if you have to)

- Prevent interpretation of metacharacters
  - Single quotes: ‘’
  - Double quotes: “”
  - Try difference between
    - `echo ‘The price is $100’`
    - `echo “The price is $100”`
  - Double quotes preserve: `$, \, ‘`
  - Enforce interpretation of command (command substitution):
  - Backtick or grave: `` ` ``
    - `echo The date is` `` `date` ``

Superuser, su, root

- In Linux, superuser has:
- *Privilege to receive all privilege*
  - insert metaphor →
- Doesn't mean the account has all privilege
- You need superuser privilege to access specific folders/files
- Rules:
  - Everyone has a normal, non-superuser account
  - Only elevates privilege when required (prompted)

Elevating privilege

- In Ubuntu, login as **root** is disabled (by default)
- Elevation of account is also usually disabled (`su root`)

- Instead use `sudo`
  - user must be part of sudo group

- Approach 1: `sudo <command>`
  - Run command with su privilege
  - Will ask you for password (user password, not root password)
- Approach 2: `sudo !!`
  - Runs previous command with su privilege
  - Convenient, no need to re-run command