Introduction to Programming Utilities

Lecture 1: Linux and The Command Line

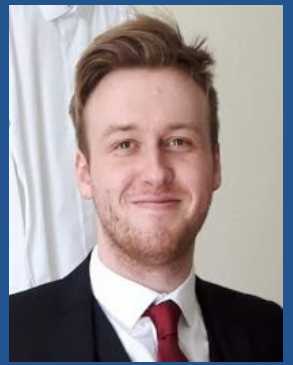
What is this course about and why it would be useful?

- "***Programming Utilities***"?
  - "*Auxiliary tools that complement programming and are useful, if not necessary, for creating softwares in a personal, educational and professional setting*"
- "***Introduction to Programming Utilities***"?
  - Teach various different programming utilities
    - Linux, Text editors, Compilers, Git, GitLab, IDEs, etc.
  - Give advice and tips **relevant to the course**
- Why useful?
  - You **will** use these utilities for programming
  - Exotic and confusing to use at first
  - Some tools **very hard** to understand at first without guidance

Who even are we?

- Philip Koo
- 2nd Year Computing
- Academic Events Coordinator of DoCSoc (Department of Computing Society) 20-21
- Deliver lectures

- Jamie Willis
- Graduate student
- GTA (Graduate Teaching Assistant)
- Run Q&A sessions (for 2020-21)



Course Syllabus

1. **Linux** and **The Command Line** (*This Lecture*)
2. **Text editor** and **Compiler**
3. Basics of **Git** and **GitLab**
4. Integrated Development Environment (**IDE**)
5. **Advanced Git** for Group Projects

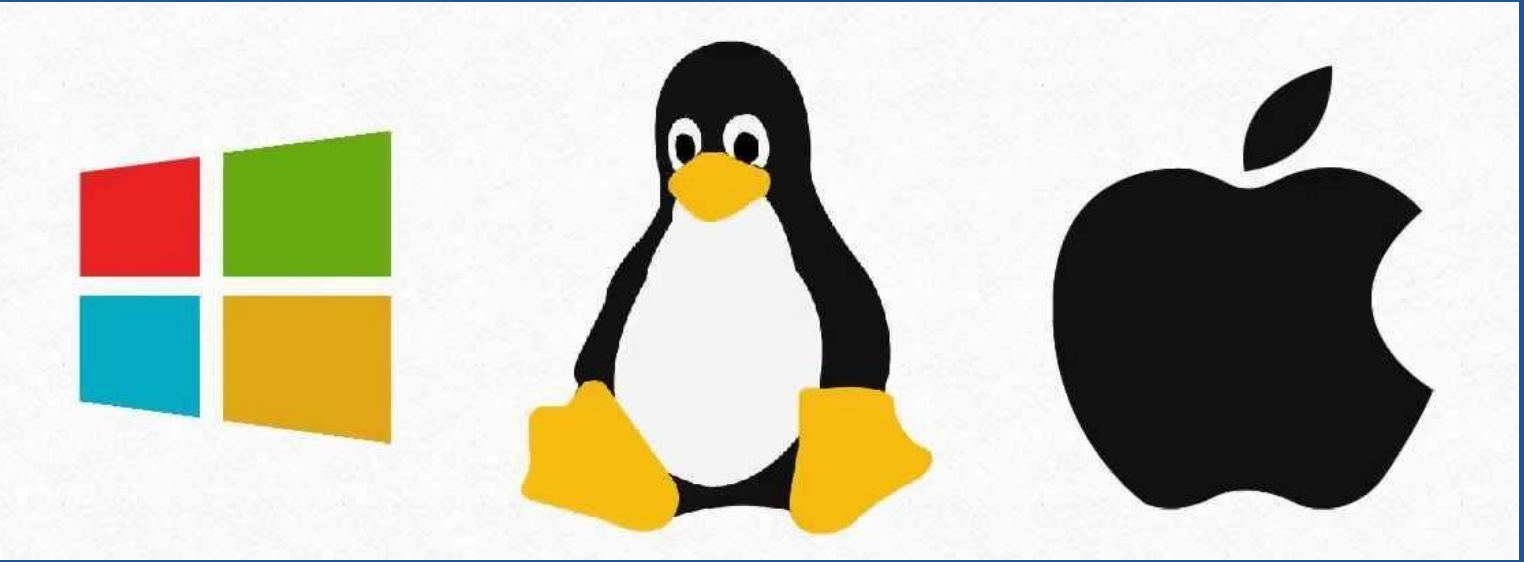
Some Logistics (for 2020-21)

- Three Support and Q&A session in **Week 1**
  - 6th Oct (**Tue**) 11:00am~12:00pm BST -> Lecture 1: Linux and The Command Line
  - 7th Oct (**Wed**) 09:00am~10:00am BST -> Lecture 2: Text Editors and Compilers
  - 9th Oct (**Fri**) 11:00am~12:00pm BST -> Lecture 3: Basics of Git and GitLab
- Lecture 4 (IDE) will run in **Week 7**
  - Exact date & time TBC
  - Lecture pre-recorded w/ Q&A session
- Lecture 5 (Advanced Git) will run sometime in the **Spring Term**
  - Exact date & time TBC
  - Format TBD
- Also ask questions on Piazza
- COMPM0101 *Introduction to Remote Learning*?

This lecture: **Linux** and **The Command Line**

- What even is “Linux”?
  - Do you need to use Linux for programming?
- Ways to install Linux distros on your own computer
- The Command Line
  - The Terminal app
  - Directory hierarchy
  - Useful commands for navigating the command line
  - File permissions
  - Package managers
- Advanced concepts in command line
  - Wildcards
  - stdin, stdout, stderr
  - I/O Redirection and Pipeline
  - SSH

What is Linux?

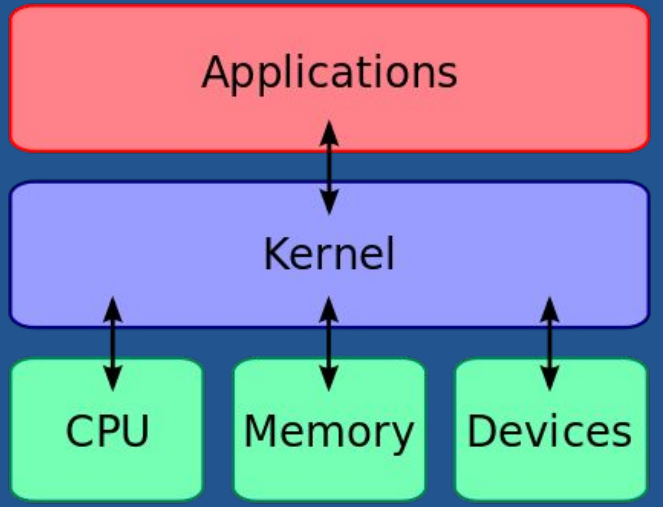


The Three Major Operating Systems?

Linux is NOT an Operating System

The Linux Kernel

“The kernel’s job is to talk to the hardware and software, and to manage the system’s resources as best as possible.”



Various Linux Distributions (“distros”)

But seriously, which distro is the “best”???

- Lab machines use Ubuntu
  - Maximum compatibility
  - Best support from lab helpers (UTAs, GTAs)
- Which version? (as of 2020)
  - Ubuntu **20.04 LTS** “*Focal Fossa*”
  - In general: match the version of your computer’s OS with that of the lab machines
- All demos for this course will be done on Ubuntu 20.04

Do you REALLY need Linux?

- No, actually!
- But it sure helps a lot!
  - Required skill for your degree here AND in industry
  - Some tools only natively supports Linux
  - Have to use it anyway if you want to use the lab computers

How do I install Linux?

1. Dual Booting

Installing two operating systems on a single partitioned disk.

Advantages

- Performance
- Battery Life
- Authentic
- Linux now good for everyday use

Disadvantages

- Inconvenient as reboot required to switch OS
- Less flexible storage space

2. Virtual Machine

Running an emulated version of Linux on top of your main host OS using apps like VirtualBox

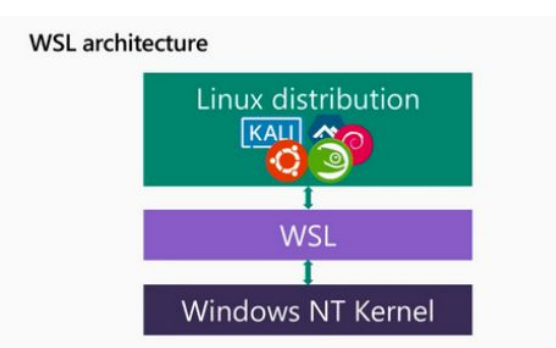
Advantages

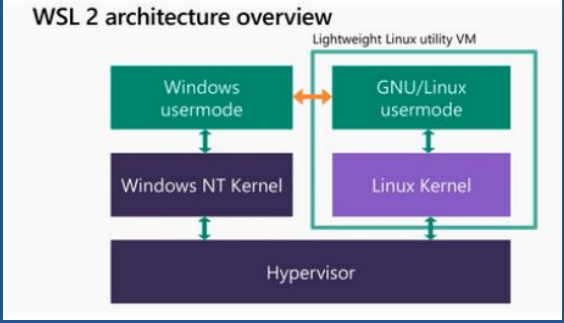
- “Seamless” if you treat Linux as an app for Computing work
- Easier file transfer between host and guest OS
- More flexible storage space

Disadvantages

- VERY slow compared to dual booting
- Significantly worse battery life
- *May* experience issues with some apps like SSH

3. (for Windows) Windows Subsystem for Linux (WSL/WSL 2)





- "*Compatibility layer for running Linux binary executables natively on Windows 10*"
- Use Linux through a terminal just like a 'real' Linux computer
- WSL 1
  - Emulates Linux Kernel on top of Windows NT Kernel that apps run on top of
- WSL 2
  - Real Linux Kernel alongside Windows NT Kernel
- WSL 2 (in general) is better

3. (for Windows) Windows Subsystem for Linux (WSL/WSL 2)

Advantages

- Provides the best of both worlds of Dual booting (performance, battery life, etc.) and Virtual Machine (convenience, file interoperability, etc.)
- Use Windows apps in conjunction to Linux apps
- Actually easier to install than dual booting

Disadvantages

- No GUI apps (yet)
  - X11 shell forwarding?
  - May change in the near future
- Harder to use for beginners
- File hierarchy is a bit weird
  - On Windows: the entire WSL drive is a subfolder in Windows
  - On Linux: Windows drive is mounted as an external drive that you can navigate to

3. (for Mac) Just use macOS (NOT Recommended)

- macOS shares root with Linux (Unix Kernel)
- So some tools like Command Line are already available on macOS
- But often NOT recommended

Advantages

- No setup needed

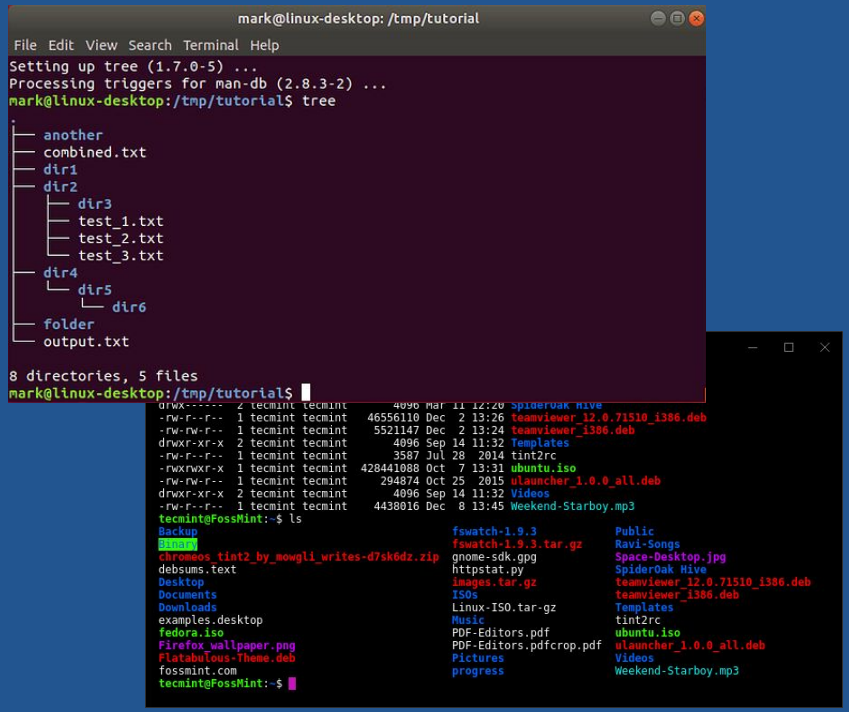
Disadvantages

- Some critical differences
  - E.g. LLVM/Clang instead of GCC, no package manager, etc.
  - macOS based on FreeBSD, not Linux
- Historically LOTS of technical difficulties come from using macOS instead of Linux

Now go and install Linux on your computer!

The Command Line

What is The Command Line?



- “*Command Interpreter that uses a Command Line Interface (CLI) instead of a Graphical User Interface (GUI)*”
- The heart of Linux
  - Many tools only work on command line
- Necessary to learn how to use it for serious productive work (esp. in programming)
- The ‘Terminal’ App
  - App for using the command line
  - `ctrl + alt + t`

Basics of Directories

- Directories == Folders
- *<folder-name>/*
- Every command runs with respect to a certain directory
- **Working Directory**
  - The current directory that the terminal window is working on
- **Root Directory** (**/**)
  - The top directory in Linux hierarchy
  - Require **root privileges** to modify (details later)
  - Should not edit files here (unless you know what you're doing)
- **Home Directory** (**/home/*<your-username>*/** OR **~/**)
  - Main folder for individual users (with Desktop, Documents, Downloads etc.)
  - Default location for all your personal files
  - Also default working directory when opening up terminal

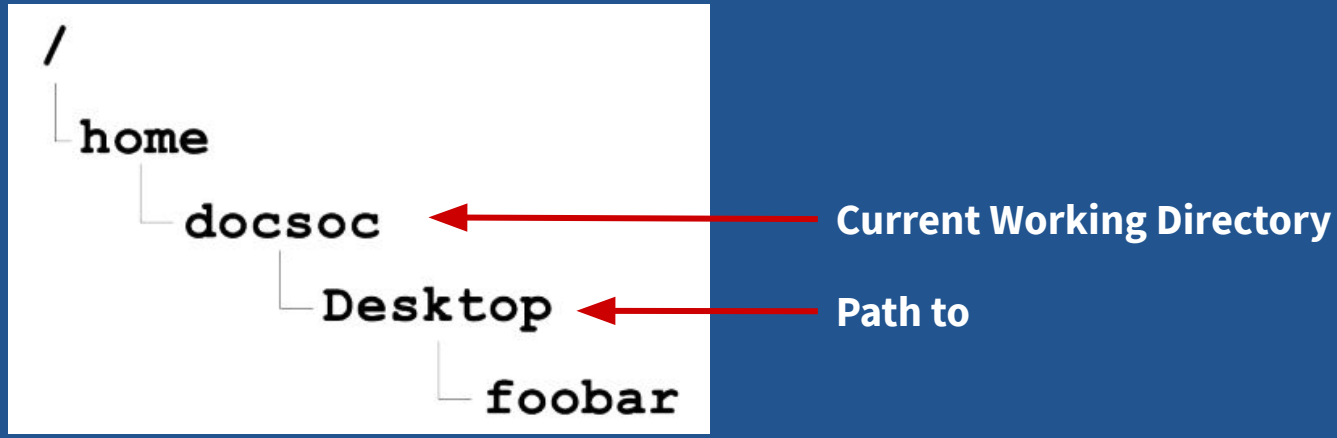
Commands: Directory Navigation

- **`pwd`** : print working directory
- **`ls`** : list items in current directory
  - `ls` **`-a`** : list all files (including hidden files)
  - `ls` **`-l`** : give a detailed list w/ file permissions, owner name, file size, date last modified, etc.
- **`cd`** *<path>* : change directory
  - **`./`** : Current working directory
  - **`../`** : Parent directory (relative to current working directory)

Relative vs absolute paths

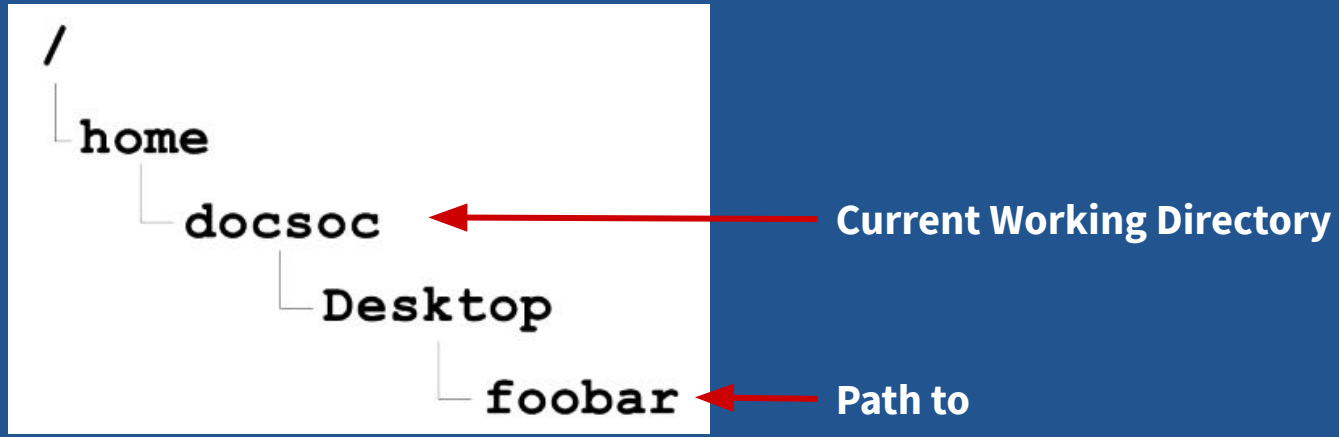
- **Absolute path**
  - Given with reference to the root directory
- **Relative path**
  - Given with reference to the current working directory

Relative vs absolute paths - example



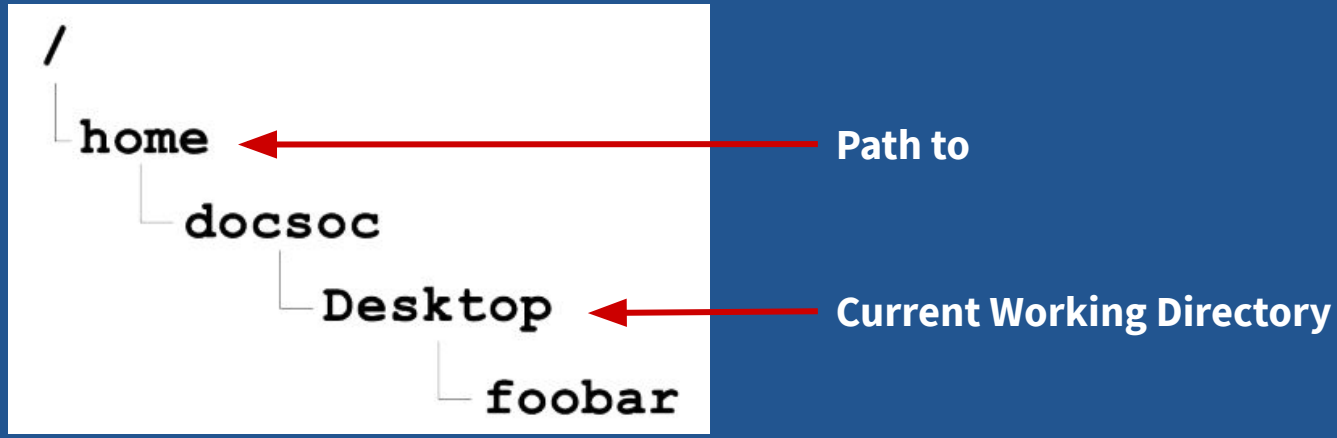
- Absolute Path: `/home/docsoc/Desktop/`
- Relative Path: `Desktop/`

Relative vs absolute paths - example



- Absolute Path: `/home/docsoc/Desktop/foobar`
- Relative Path: `Desktop/foobar`

Relative vs absolute paths - example



- Absolute Path: `/home`
- Relative Path: `../../`

Demo 1: Directories and Paths

Commands: Creating and Destroying Files

- **`touch`** *`<file-name>`* : create a new empty file
- **`mkdir`** *`<directory-name>`* : create a new directory
- **`cat`** *`<file-path>`* : display the contents of the file
- **`rm`** *`<file-path>`* : remove the file
  - rm **-r** : delete **recursively** - search through subfolders as well
  - rm **-f** : **force** delete - don't ask for user's permission
- **`cp`** *`<src-file> <dest-file>`* : copy file/directory
- **`mv`** *`<src-file> <dest-file>`* : move file/directory
  - Also used for renaming files

Demo 2: Creating and Destroying Files

Commands: Miscellaneous

- **sudo** *<command>* : allows root privileges
  - **root** : user with highest authority; can read, write, and execute anything
  - Necessary for some commands (like apt-get)
  - But VERY dangerous for some operations (e.g. “sudo rm -rf /”)
- **echo** *“string”* : prints the string on the terminal
  - Useful with Redirection and Pipelining (details later)
- **grep** *“string” <file-path>*
  - print out every instance where *“string”* occurs in file
  - grep **-n** : print the line number next to string
  - grep **-r** : search within a directory instead
- **man/help** *<command>* : shows a manual for the command
- **./<executable>** : run the executable

File permissions and chmod

```
drwxr-xr-x  2 hk619 hk619 4096 Oct  2 04:17 Desktop
```

- **d** : the object is a **directory** **-** : the object is a **file**
- Next 9 characters indicate the individual permissions of a file
  - **r** : file/directory is **readable**
  - **w** : file/directory is **writable**
  - **X** : file/directory is **executable**
- First 3 characters -> permissions for the **owner**
- Next 3 characters -> permissions for the owner's **user group**
- Last 3 characters -> permissions for **everyone else**

File permissions and chmod

- **chmod** *<mode> <file/directory>* : change permission
  - mode argument given as a 3-digit number, each digit in the range 0-7

Digit	Shortcode	Permissions
7	rwx	read, write and execute
6	rw-	read and write
5	r-x	read and execute
4	r--	only read
3	-wx	write and execute
2	-w-	only write
1	--x	only execute
0	---	none

Package Managers

- Download programs off the internet from a trusted central server
- Most of the programs will probably be available on package managers
- All software are verified securely before uploaded onto the server
- Different for each distro
  - **`apt/apt-get`** : For Debian-based distros (Ubuntu, Mint etc.)
  - **`yum`** : For Fedora-based distros (centOS, Red Hat etc.)
  - **`brew`** : For macOS (Unofficial)
  - etc.

Commands: apt-get

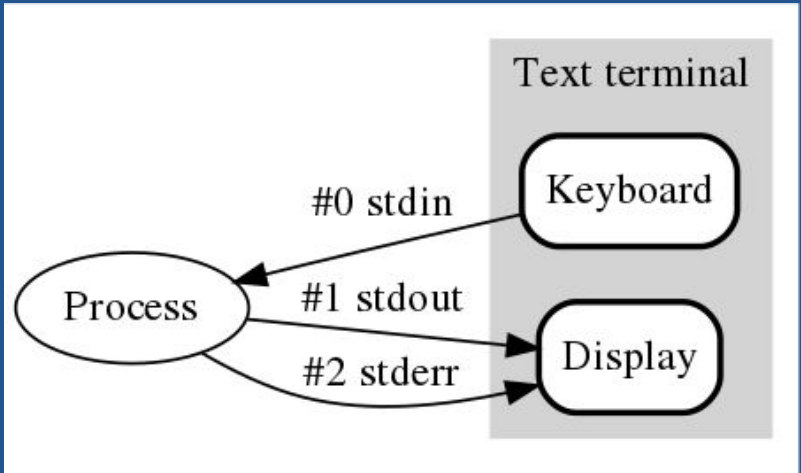
- **`sudo apt-get update`**
  - Update the list of softwares from repositories
- **`sudo apt-get upgrade`**
  - Update (or upgrade) the softwares installed on the operating system to the latest version available on the repositories
- **`sudo apt-get install`** *<package>*
  - Install application
- **`sudo apt-get remove`** *<package>*
  - Uninstall application

Advanced Concepts of The Command Line

Wildcards

- A set of symbols that stand in for other characters
  - Used to substitute for any other character(s) in a string
  - Can be used for arguments to many Linux commands
- **?** : matches any single character
  - e.g. “**e??**” matches “**egg**”, “**eel**” but not “**chicken**”, “**eggs**”
- ***** : matches any character or a set of characters, including no characters
  - e.g. “e*” matches “**e**”, “**eel**”, “**eggs**” but not “**cggs**”
- **[]** : match characters enclosed in square brackets
  - e.g. “e[acu]d” matches “ead”, “ecd”, “eud” and “e[a-c]” matches “ea”, “eb”, “ec”

`stdin`, `stdout` and `stderr` streams

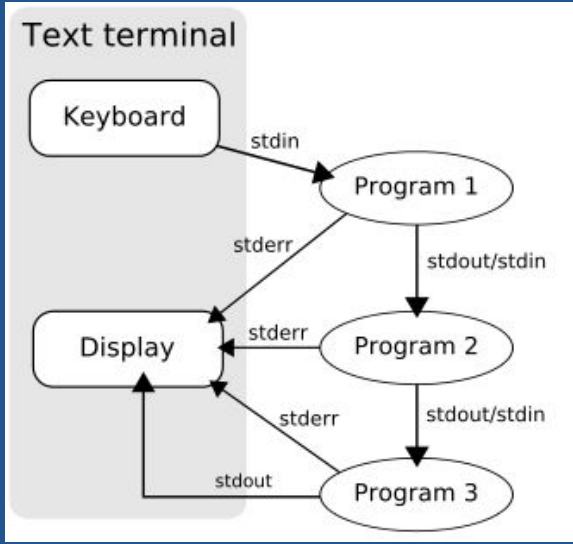


- Streams: sequence of data
- Virtual input and output channels into and out of the command line
- **`stdin`** : **st**an**d**ard **in**put
  - Handles input from the user's keyboard to a command
- **`stdout`** : **st**an**d**ard **out**put
  - Handles output from a command to the user's display
- **`stderr`** : **st**an**d**ard **err**or
  - Handles displaying error messages from a command

I/O Redirection

- Pass output of a command to a **file** or pass file as an input of a command
- ***command > file*** : Output Redirection
  - Redirect the output of a command to a file
  - e.g. `echo "Hello, world!" > hello.txt`
- ***command >> file*** : Output Redirection
  - Same as > except the output is appended to the end of the file
  - e.g. `ls -al >> list-append.txt`
- ***command < file*** : Input Redirection
  - Use the given file as the input stream for the command
  - e.g. `./pretty-printer < input.txt`

Pipeline



- Joining the standard output of one command to the standard input of another command
- Analogy: a factory and a conveyor belt
  - Programs -> workers
  - Pipeline -> a conveyor belt
  - Input from stdin -> Products
- **`program-1 | program-2 ...`**
  - e.g. **`cat input.txt | grep "world" | ./printer`**

SSH (Secure Shell) and SCP (Secure Copy)

- SSH
  - Allows you to remotely access the terminal of any other public Linux computer on the internet, including DoC lab computers
- **`ssh`** `<username>@<computer>`
  - Username: name of the user to remotely access the computer
  - Computer: either the hostname or the IP address
- SCP
  - Allows a secure file transfer between a public computer and your local directory, notably file transfer from DoC computers to your own computer
- **`scp`** `<username>@<computer>:<file-path> <local-path>`
- Detailed guide can be found on the CSG website

Department of Computing Society

Imperial College London

docsoc.co.uk | fb.com/icdocsoc | twitter.com/icdocsoc