# UNIX

Things to do

# Unix Philosophy

- This lecture is about the setup and philosophy of Unix
- Lecture Plan
  - Basic Philosophy
  - File System
  - Core Commands
  - Redirection
  - Pipes
  - Processes
  - Shells
  - System Interface to C Programs
  - Next Things

# Basic Unix Philosophy

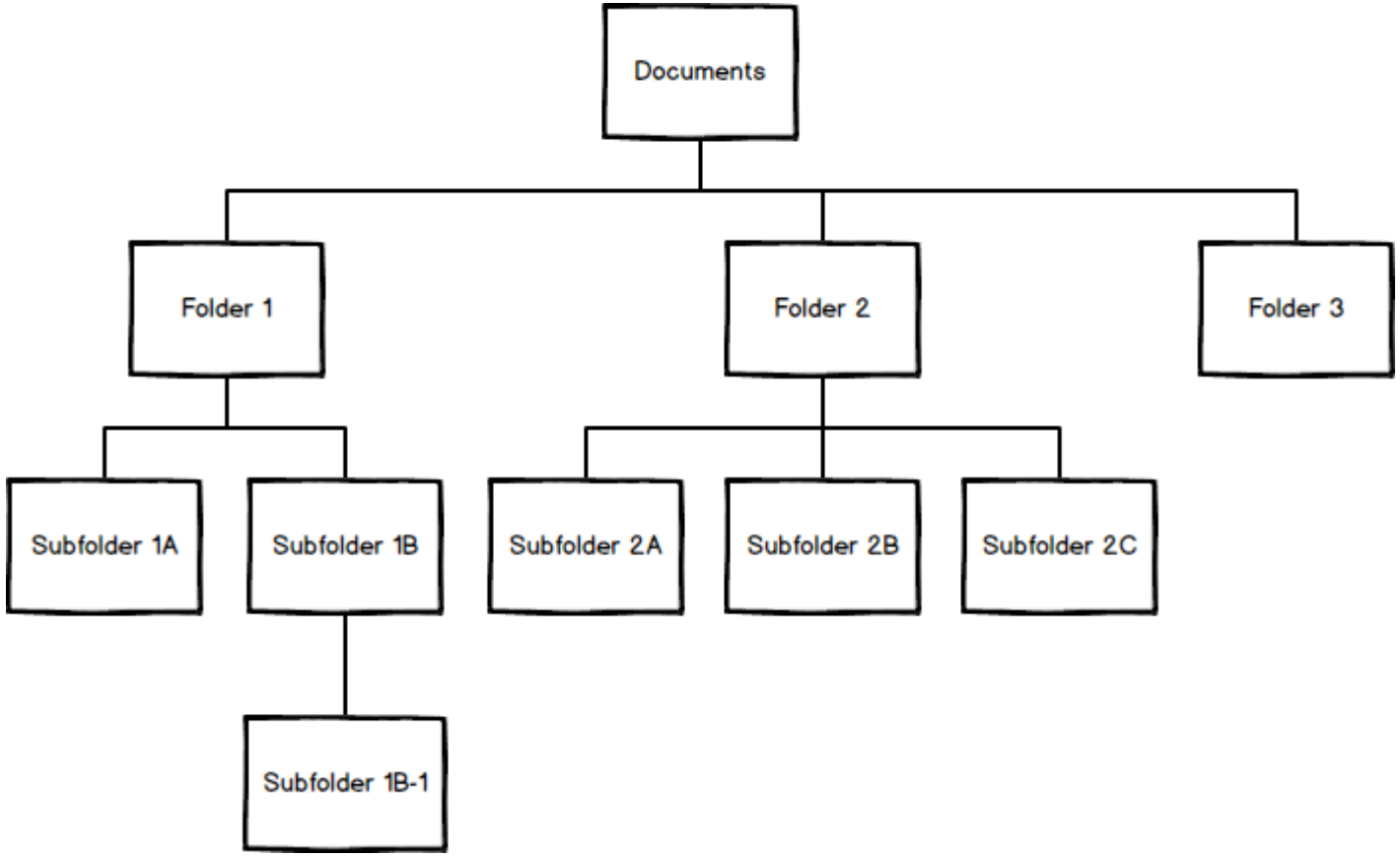
EVERYTHING IS A FILE!!

- Directories are files
- Executables are files
- Devices are files
- Files are… files

# File System

Consists of:

- Files
- Directories (which contain more files)



# Navigating the Tree

Can be done by direct path:

- /home/bernard/dir1/dir2/file1.file

Or relative path:

- If I am in /home/bernard
- dir1/dir2/file1.file

**Cue: Do a DEMO**

# Permissions

3 Kinds

- Read, Write, Execute

Also 3 Kinds

- User, Group, Global

Can be seen using: **ls –l**

Can be fiddled using: **chmod**

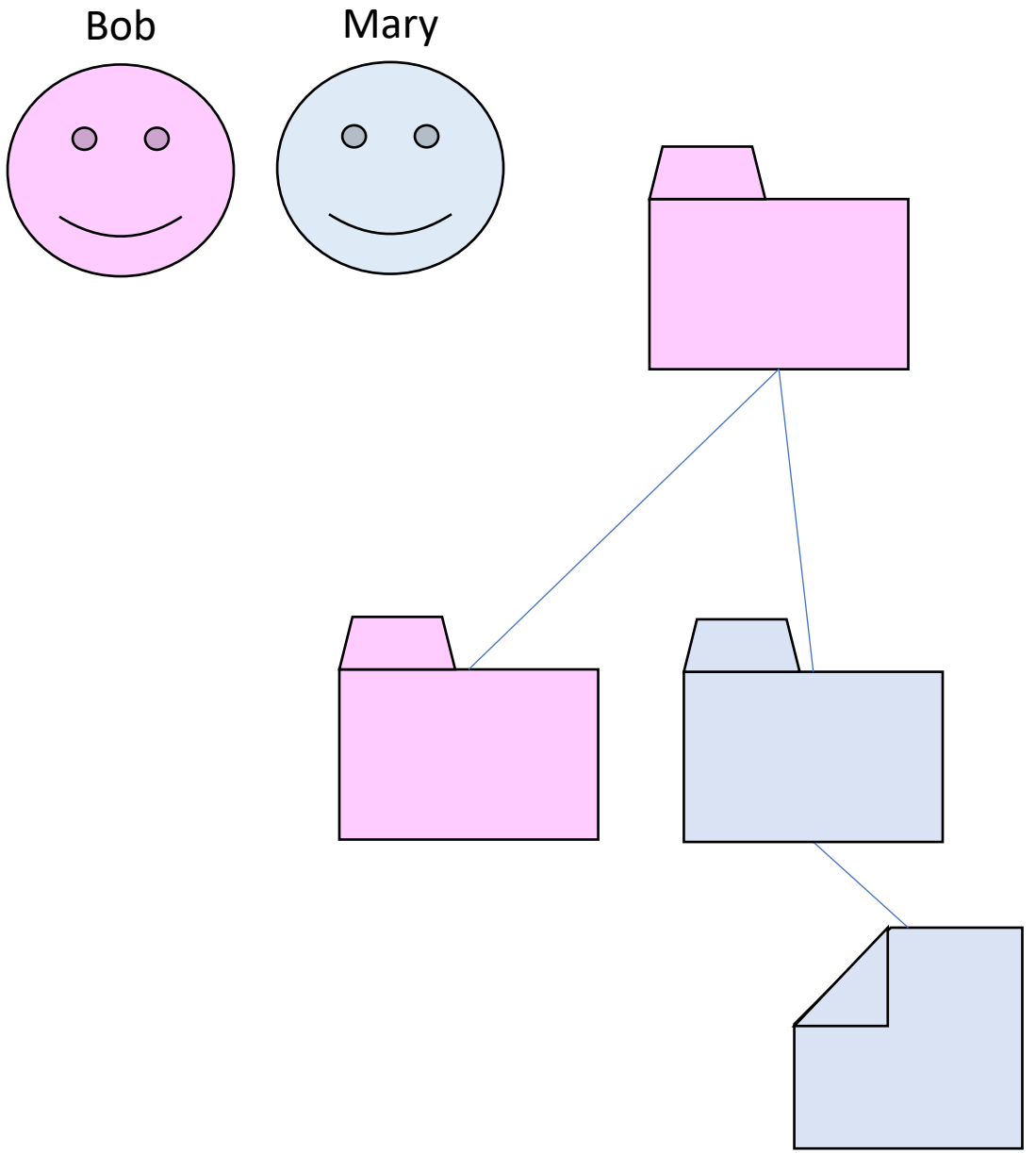
Unix Examples:

- cantread.sh Can’t copy it
- cantwrite.sh Can’t write to it
- cantexe.sh Can’t execute

# Users

A user is basically a set of **files and/or directories** that belong to them or they have **permission** to access.

Users can interact with the system by running a process called a **shell.**



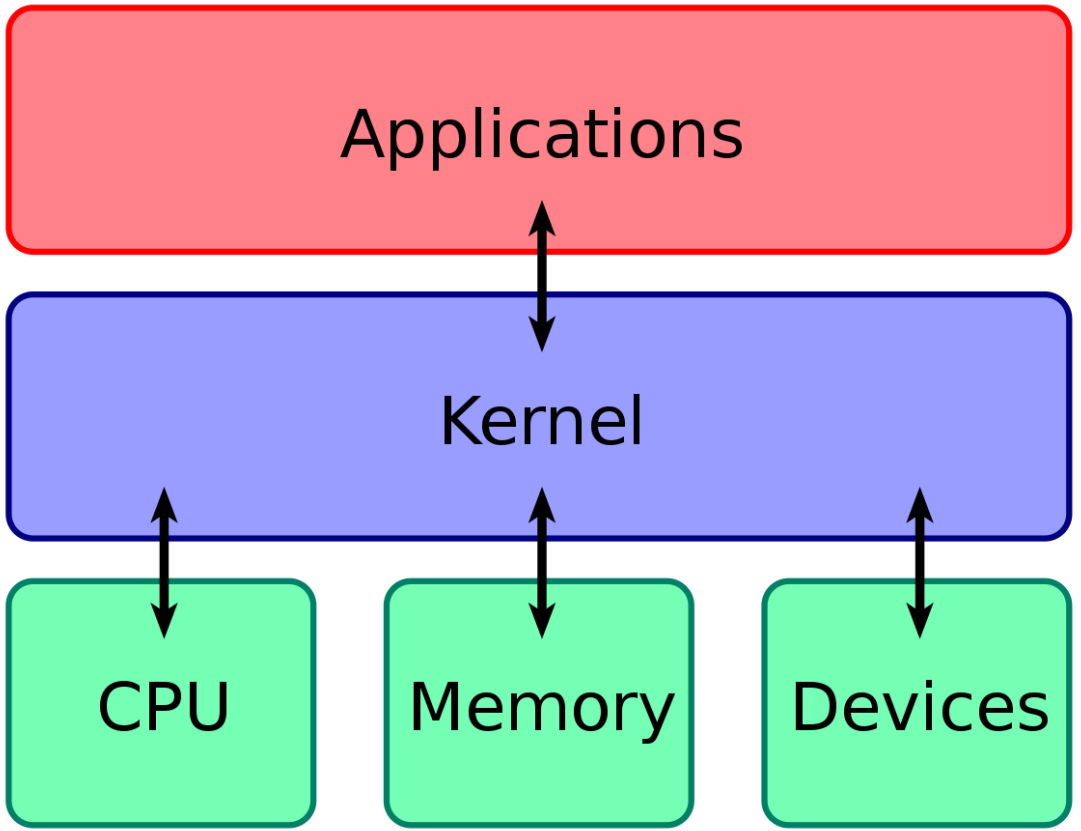
# The Kernel

The Kernel

- Is a program
- It is always there

The Kernel manages:

- Files
- Users
- Devices
- Processes (not technically files… but kind of look like files)



# Devices

Devices include things like:

- Terminal
- Network Port
- Keyboard
- Disk
- RNG

Can be viewed using:

- lsblk, lspci, lsusb etc

# Processes

Processes:

- Programs that are running
- Are owned by a user
- Can launch other processes
  - A **parent** process launches a **child** process
- Generally, if the parent dies, the child dies
  - This can be worked around

# Shell

The Shell

- Is an interactive process that allows a user to run new **processes**
- All commands seen thus far are **programs** turned into **processes**

When the user logs out, the shell is gone

- This includes processes
  - This can be avoided

Not all processes are spawned from a shell

- The Kernel is its own thing (starts processes on start up)
- These processes tend to respond to external events (timers, messages etc)

# Basic Commands

Lots:

- ls, cp, mv, rm
- less
- pwd, cd
- cat
- grep, egrep
- head, tail, wc
- cut
- paste
- ps, kill

# Redirection

Unix allows output to be redirected

- Putting into a file: >
- Adding to the end of a file
- For piping to other programs

We can redirect not just to our standard input and output but too other file descriptors

- This will be covered in detail later

# C Programming Interface

C

- Has a special relationship with Unix
- It is the first programming language in Unix
- Very Low Level
- There are many commands that C programs can use to interact with the system (all of these are in section 3 of the unix manual)

The two very basic parts are:

```
int main(int argc, char * argv[])
{
    return 0;
}
```

**These interact as input arguments**

**This is the return status**

# Housekeeping

Workshops

- Your first workshop is this week

Assignment #1 is out.

- It is in BASH, learn BASH

Practice:

- Write some bash scripts that do things (move files, create files, calculate numbers)