COMP 3430

Operating Systems

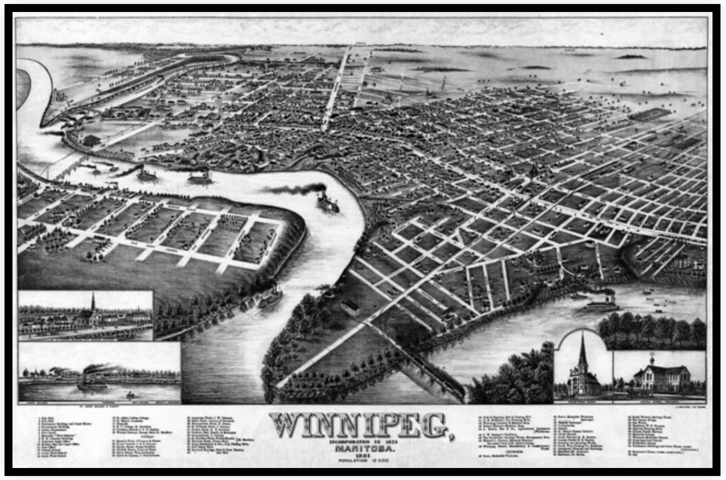
May 6th, 2019

Today

- Course overview.
- Responsibilities of an OS.
- Historical motivations.

Course overview

All course material will be posted to UM Learn.



Your environment

- *Everything* is done on Linux (`aviary.cs.umanitoba.ca`)
- Assignments **must**
  - … be submitted with `handin`
  - … work on `aviary`
  - … be written in C
  - … include a `README` and `Makefile`

Pixabay License

Your environment

- You can use *any* C compiler you want
  - … as long as it's on `aviary`.

My expectations for you

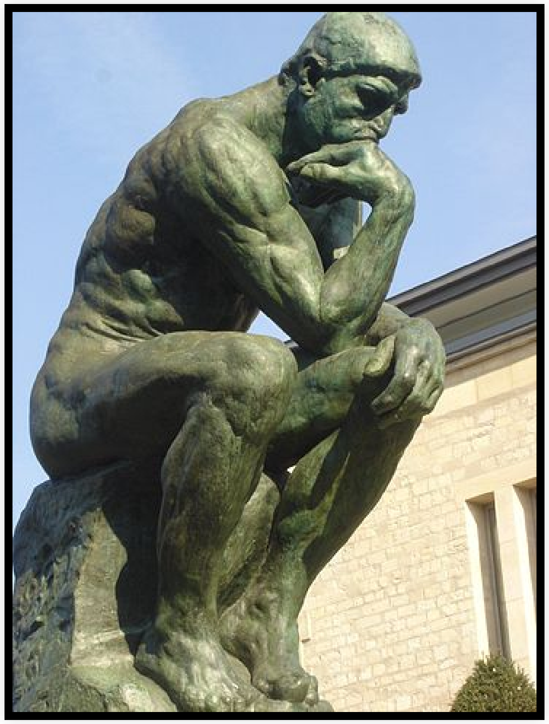
As a student in this class, I expect you to…

- Read. A lot.
- Participate in class!
- Get as much out as you put in.

Pixabay License

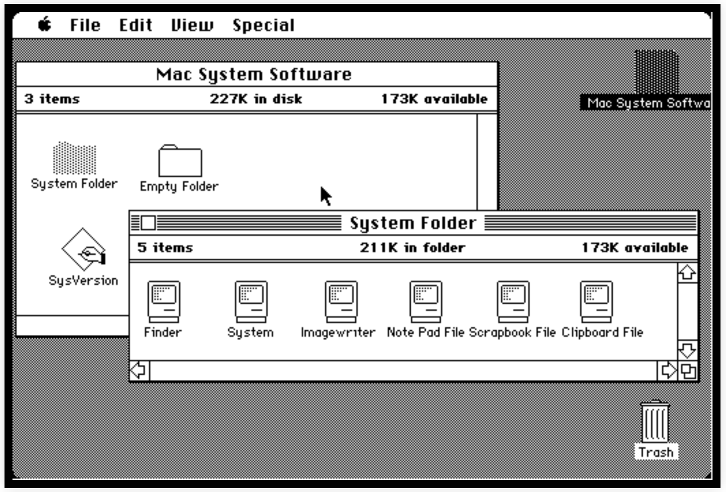
What do *you* want?

- What do *you* want to learn from this course?
- What are your expectations for *me*?



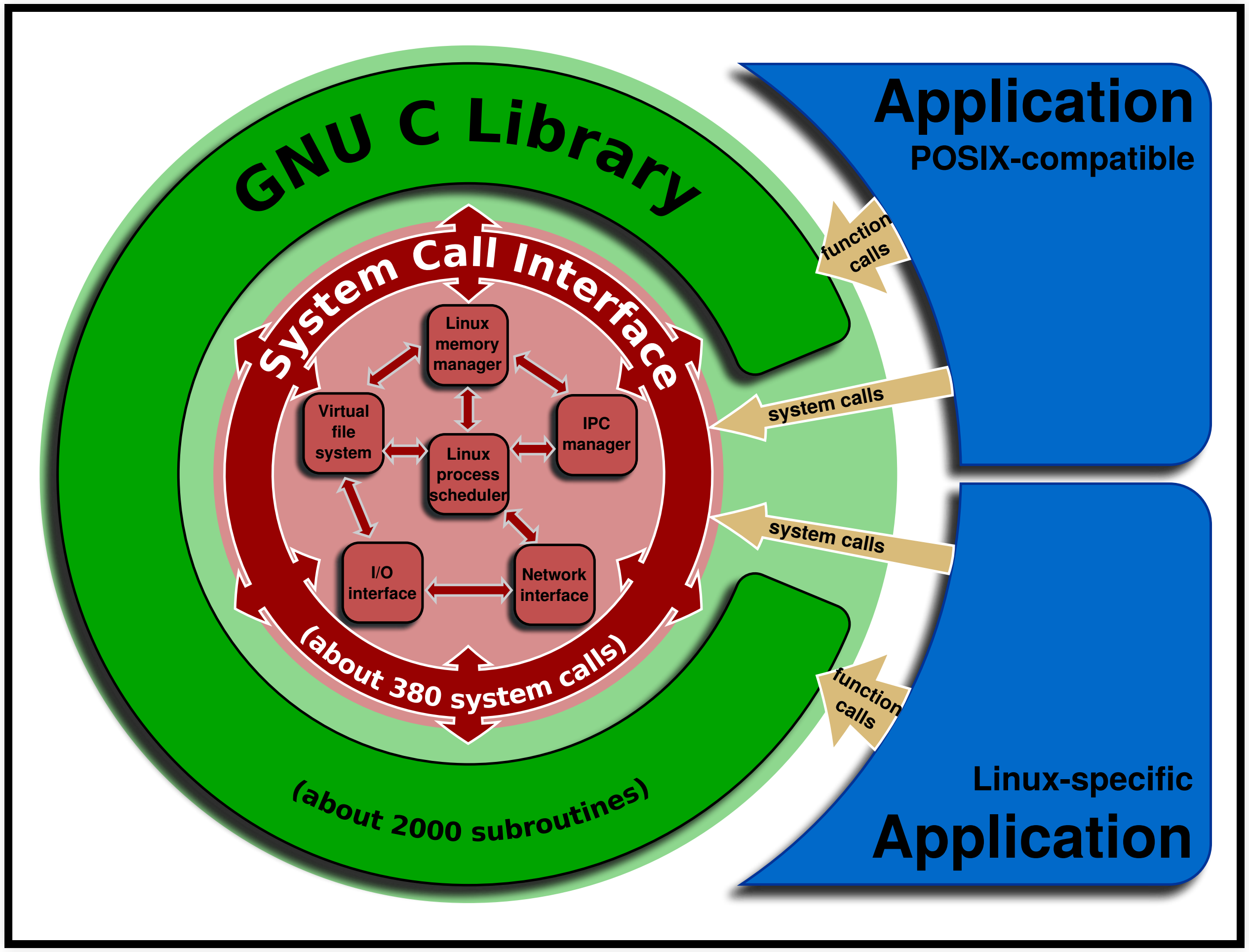
With the person beside you, answer the questions:

What *is* an OS?



Fair use

What *is* an OS?

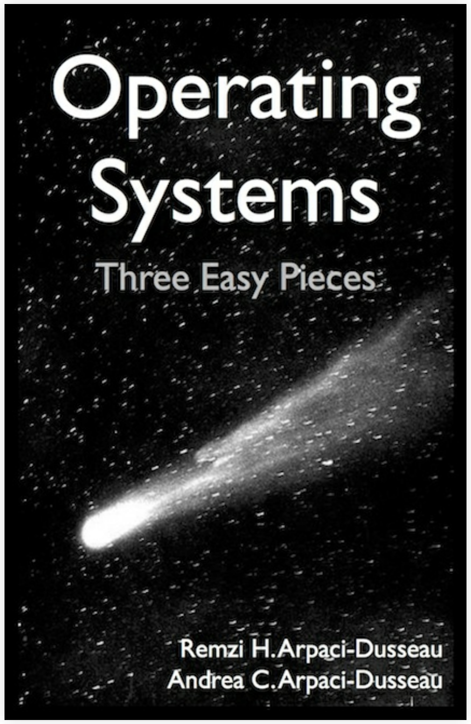

GNU C Library
(about 2000 subroutines)
System Call Interface
(about 380 system calls)
Linux memory manager
Virtual file system
IPC manager
Linux process scheduler
I/O interface
Network interface
function calls
system calls
system calls
function calls
Application
POSIX-compatible
Linux-specific
Application

What *is* an OS?

- An OS is *just* software. It’s code all the way down.
  - An OS is *not* magic   .
- At the interface of hardware and software.
  - Operating systems are *tightly coupled* with hardware.
  - Tightly coupled so *your* software doesn’t need to be.

Three easy pieces

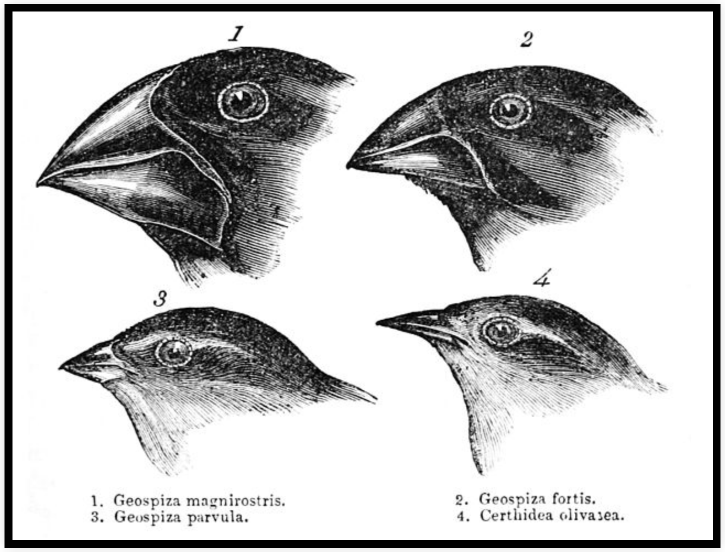
- We're going to look at *parts* of an OS
  1. Virtualization
  2. Concurrency
  3. Persistence
- Let's take a look at the text book.



© RH and AC Arpaci-Dusseau

How did we get here?

Let's take a brief (emphatically **not** comprehensive) tour through the history of computing and operating systems.



Public Domain

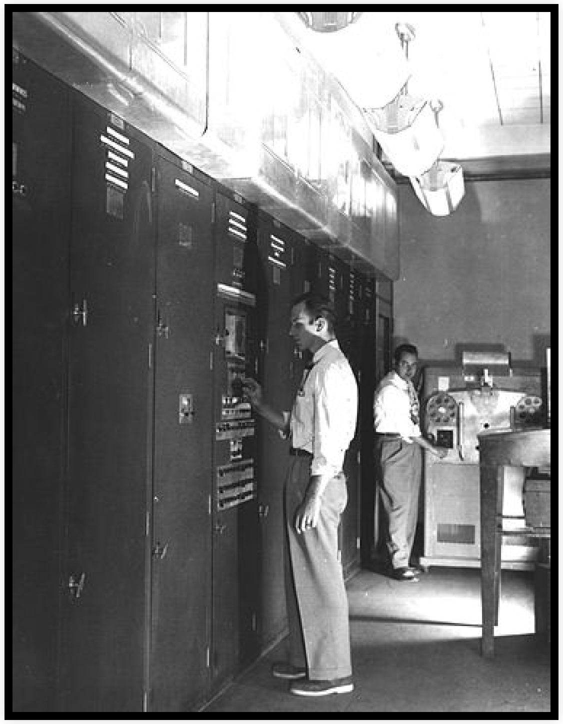
In the beginning…

- … maybe *too* far back.

Public Domain

In the 1940s

- EDVAC
  - About the size of a small house (size and weight)
  - Required 30 people per 8 hour shift to operate
  - ~5.5kb memory



Public Domain

EDVAC Software

- No OS.
- Was *not* portable – written specifically for the *hardware*.
  - Knew *everything* about the hardware.
- One program runs at a time.
  - Program entry is *entirely manual*.

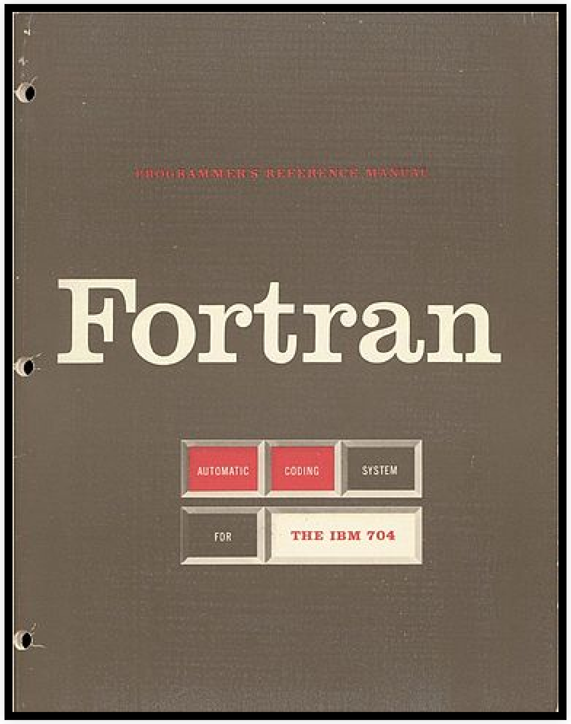
Responsibility

Given *your* experience with computing and the state of software on EDVAC, what responsibilities might a *modern* operating system have?

Pixabay License

1950s and 1960s

- Programming languages begin to abstract hardware
  - Lisp, FORTRAN, COBOL
- With languages come libraries (abstracting hardware)
- “Operating systems” appear (GM-NAA I/O)
  - Programs are automatically queued (but only one at a time)
  - **Not** interactive



Public Domain

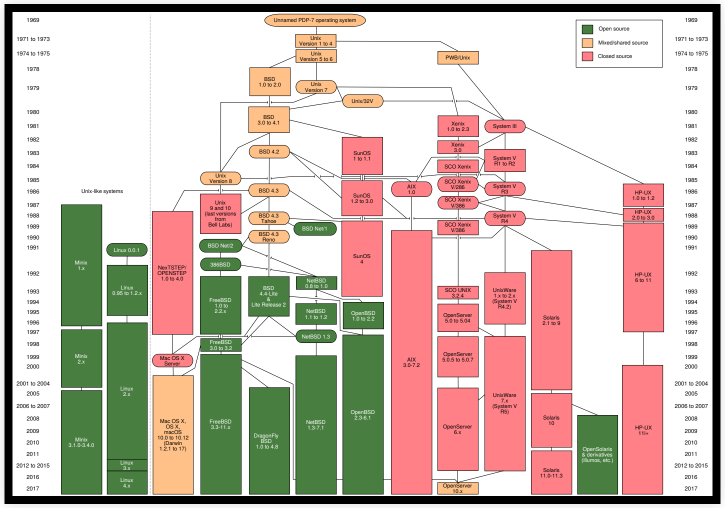
Responsibility

Given *your* experience with computing and the state of software in the 1950s and 1960s, what responsibilities might a *modern* operating system have?

Pixabay License

1970s

- Remote access and “Time-sharing”
- Real operating systems appear: MULTICS and UNIX



CC BY-SA 3.0

Responsibility

The OS is a **manager** of resources

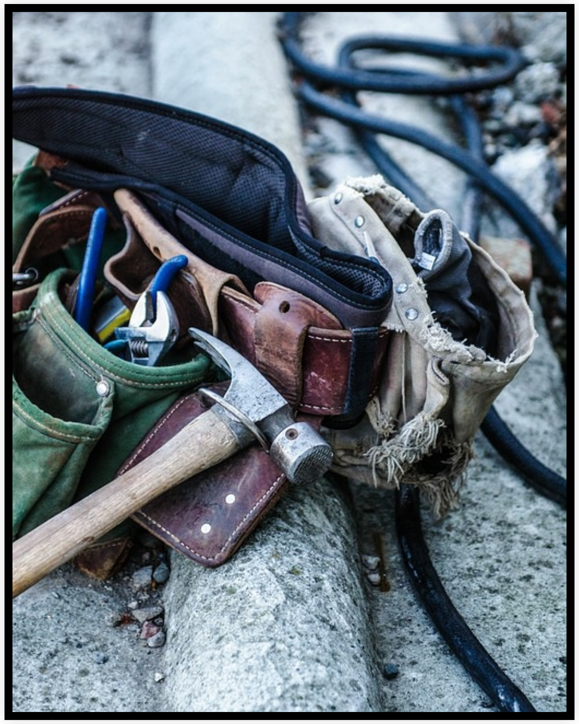
- Processors
- Storage & memory
- I/O devices
- Security/protection

Pixabay License

Responsibility

The OS is a **tool belt** for programmers

- Software libraries of functionality
- Hardware abstracted by “drivers”

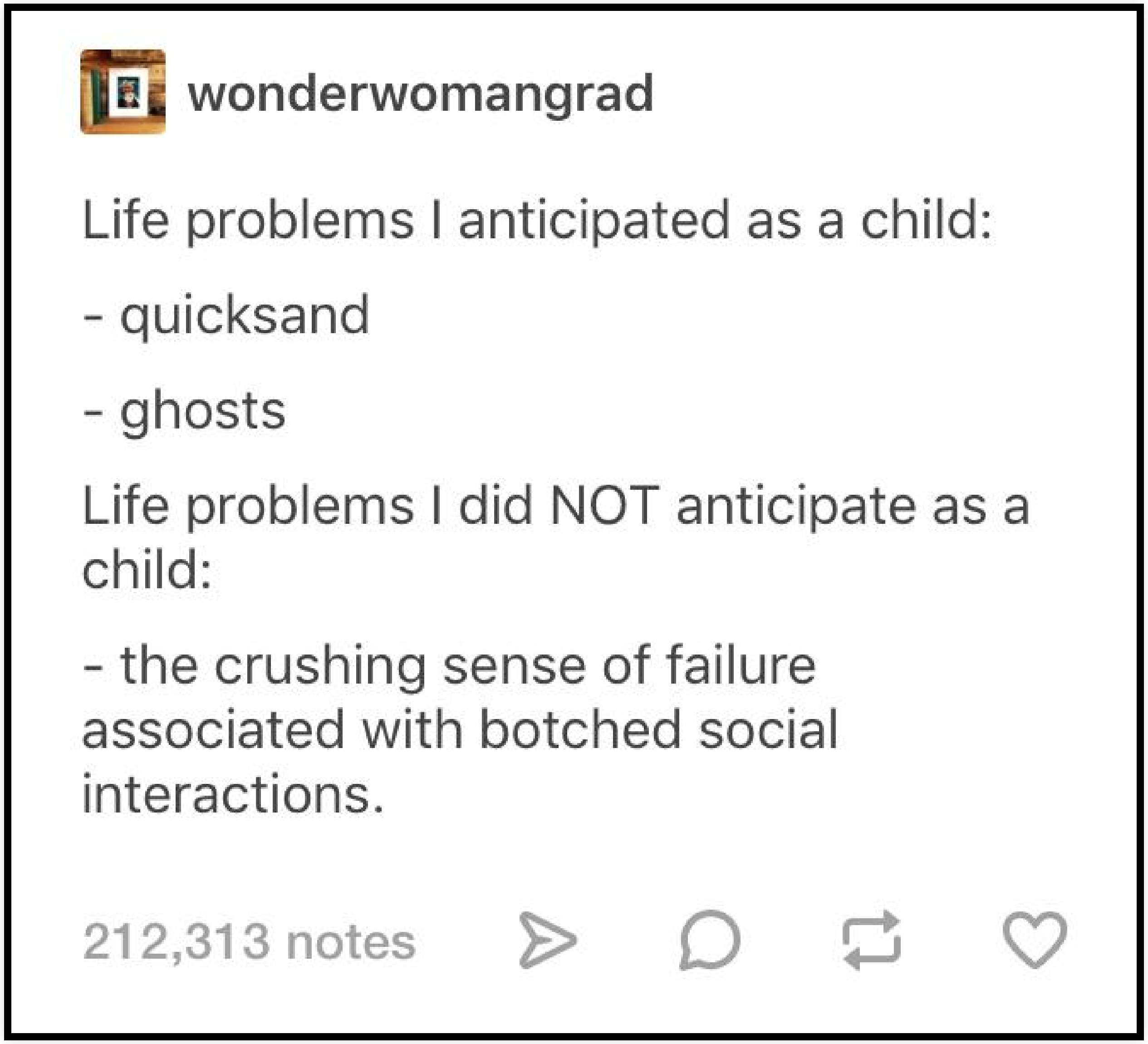


Pixabay License

Modern Operating Systems

- We're going to be looking mostly at Linux
- UNIX still lives (in your mac)
- Windows… exists.

Pixabay License



wonderwomangrad

Life problems I anticipated as a child:

- quicksand

- ghosts

Life problems I did NOT anticipate as a child:

- the crushing sense of failure associated with botched social interactions.

212,313 notes