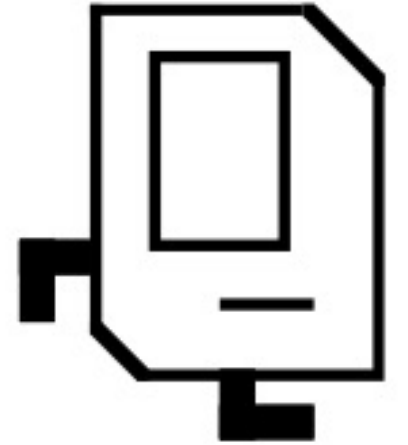


# Paths in Computer Science: Systems

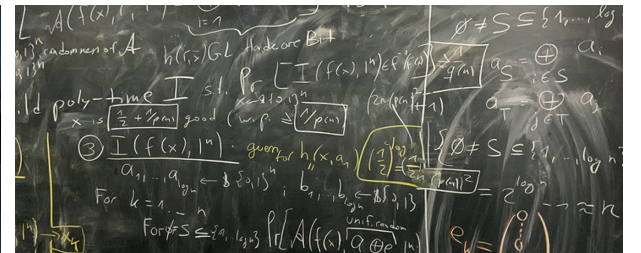
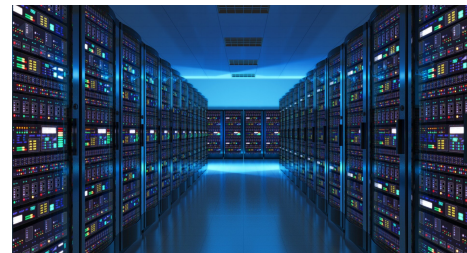
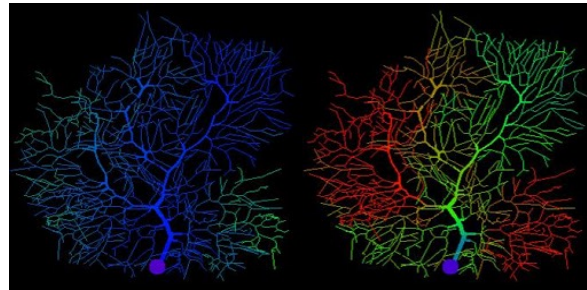
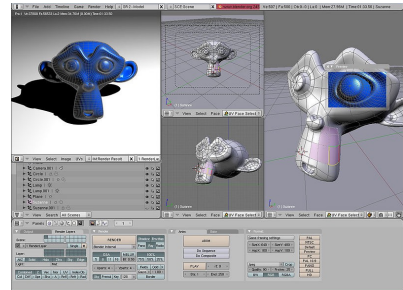
Tara Jones



# Introduction

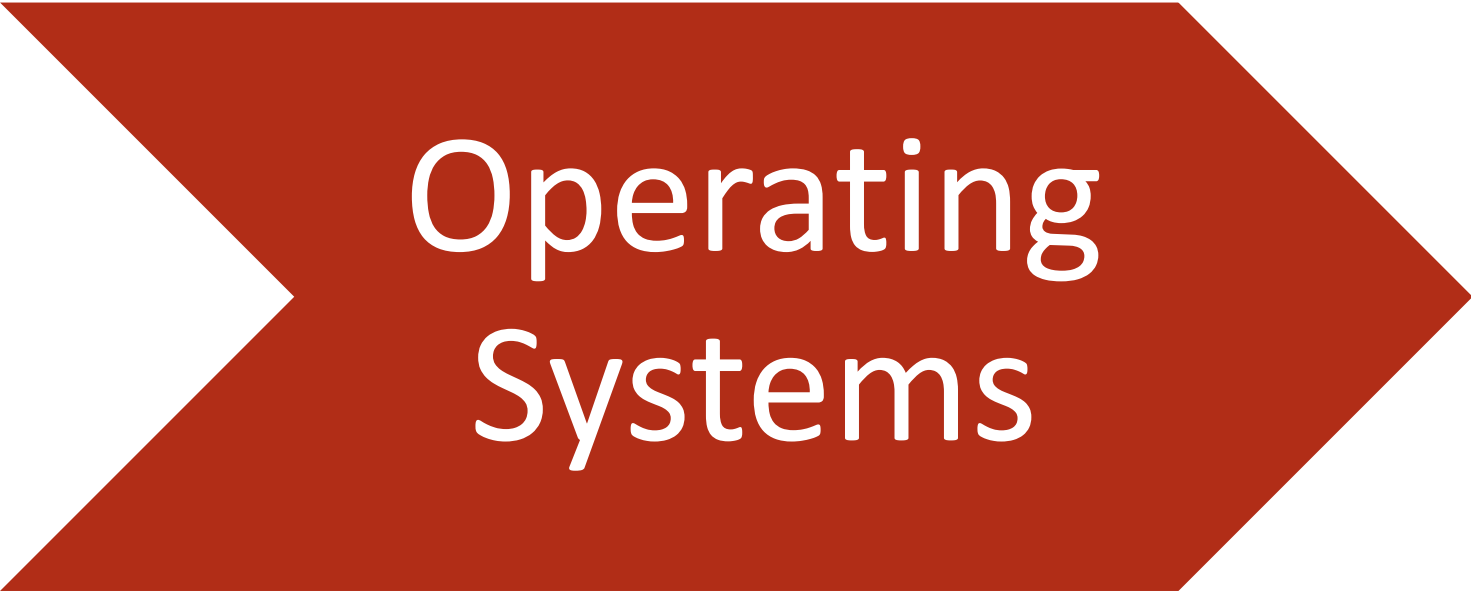


There are a lot of paths one can take when studying computer science



# What is Systems?

Systems is the study of the design and implementation of computer systems such as compilers, databases, networks, and operating systems.



Operating  
Systems

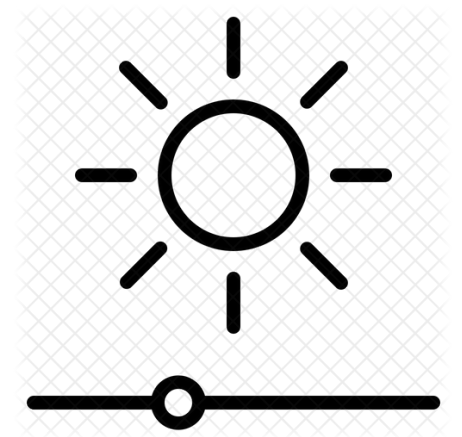
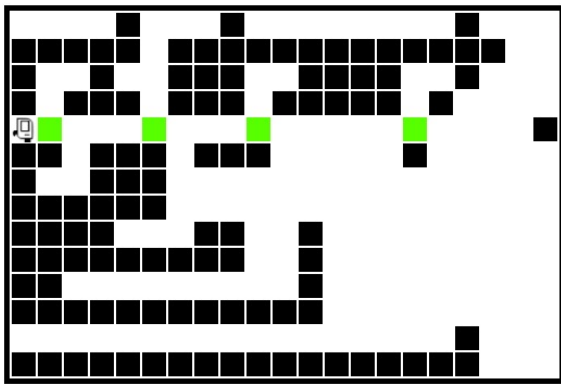
# Earliest Computers

- Around the 1940s computers were room-sized machines
- Programs were entered into punch cards and fed to the machine one at a time



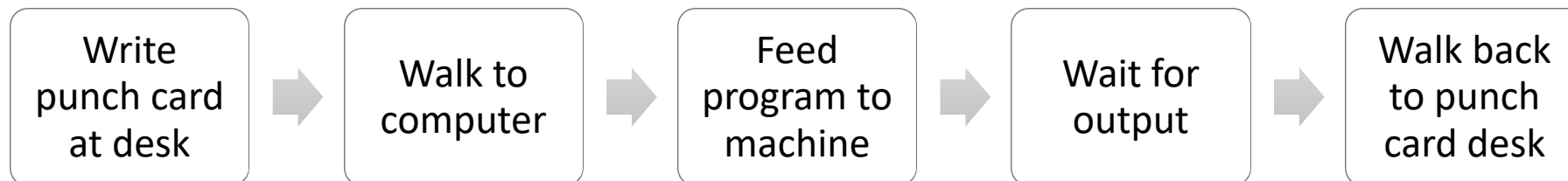
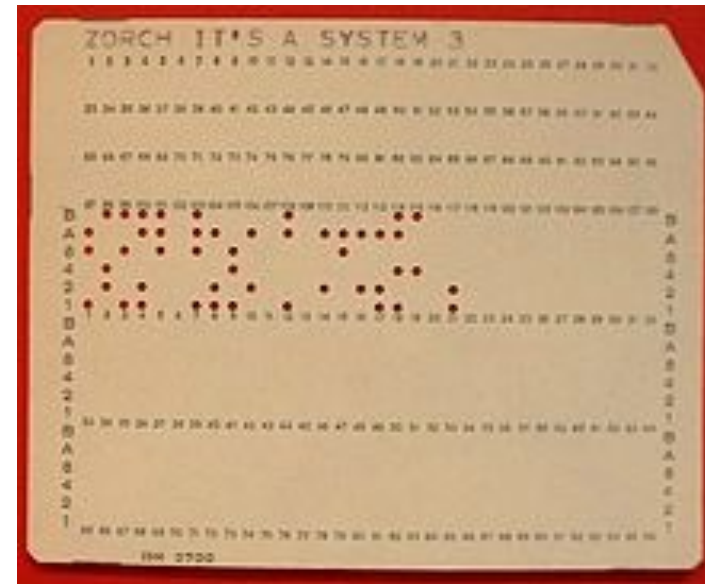
# Earliest Programs

- Early computers were used for calculations, similar to what we use a calculator for
- Modern day programs are anything that runs on a computer:



# The First Computer Programmers

---



# Why can a computer only run one program at a time?

- Picture you need to do a hard math problem:

$$21-3+18\div 6$$

- And then I gave you another math problem to do:

$$15-1(12\div 4+1)=?$$

- Your brain, like a computer, has a certain amount of processing power. It would be physically impossible for you to solve these problems simultaneously. Instead you would need to solve problem one, then problem two, or alternatively, solve a little of problem one, and then a little a problem two until done.

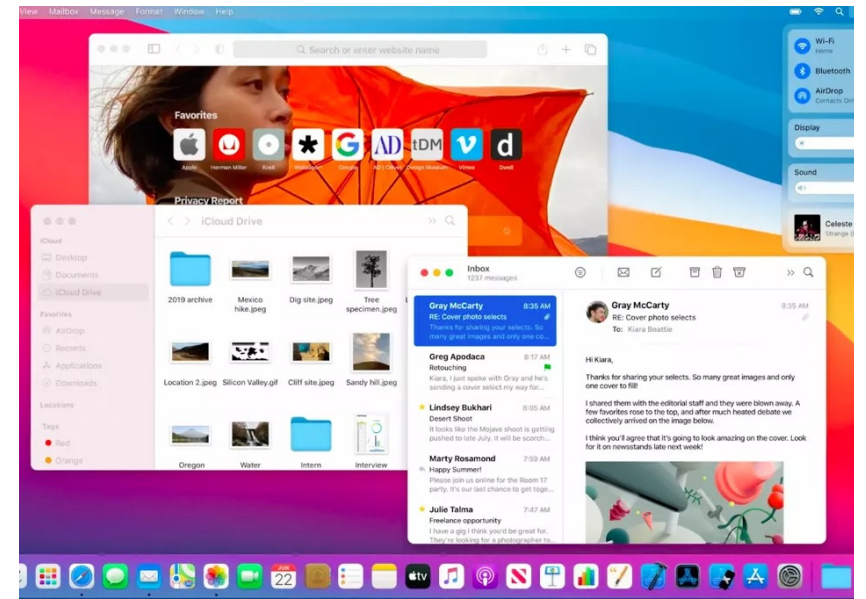


If a computer can only run one thing at a time, how did we get to where we are?





# OPERATING SYSTEMS

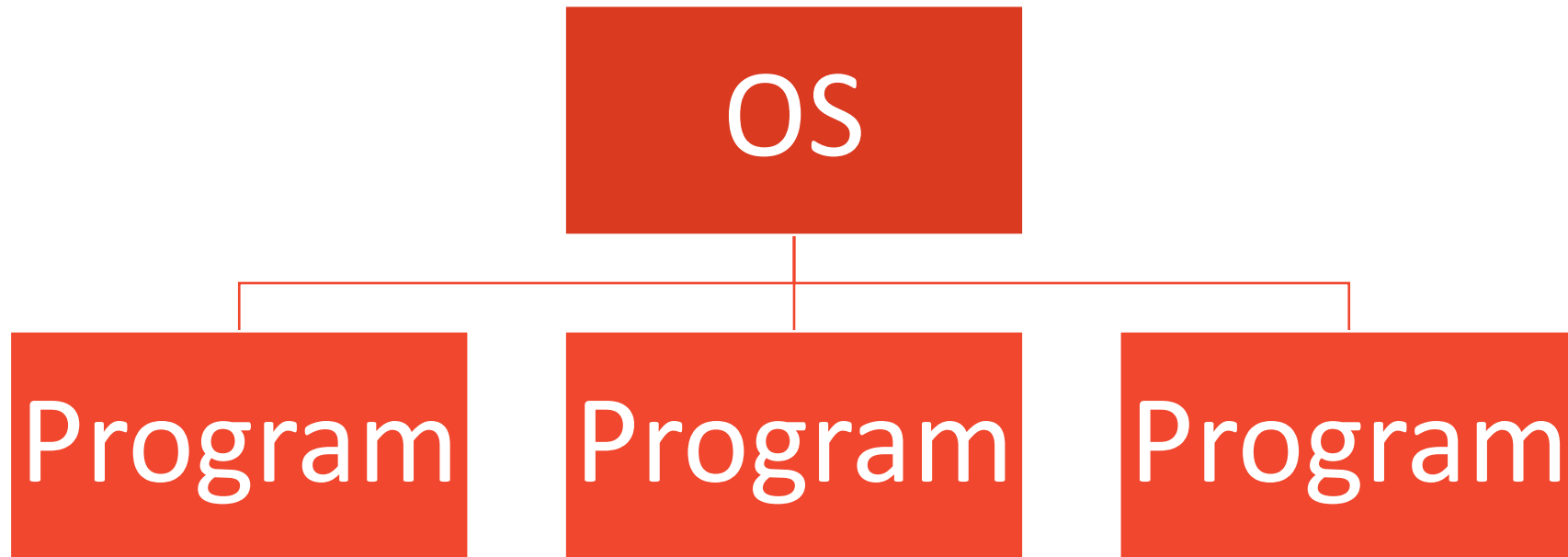


As the computers became extremely fast, humans loading in the programs become extremely inefficient.

What if instead, someone wrote a program to take the role of this engineer?

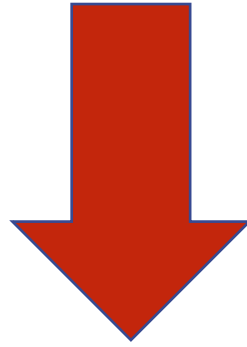


# This is where Operating Systems came from



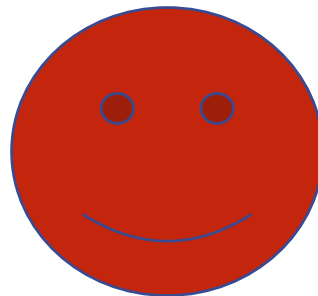
Instead of humans feeding a computer tasks one by one, the OS has a list of programs it wants to run and runs them in that order, waiting for one to finish before starting the other one

Let's dive into how this works...



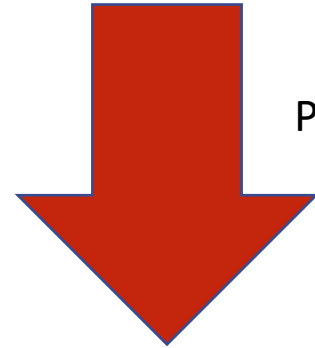
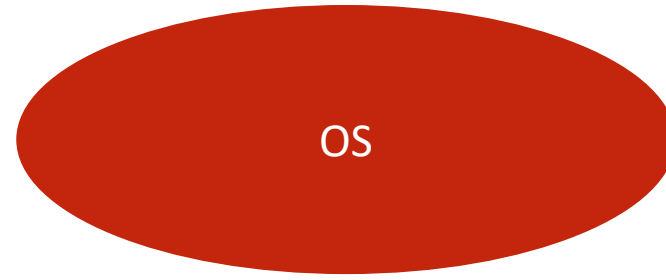
Program is told to run  
by the OS

... RUNNING ...



Program has output  
which tells the OS it  
is done, and ready to  
move on to the next  
program

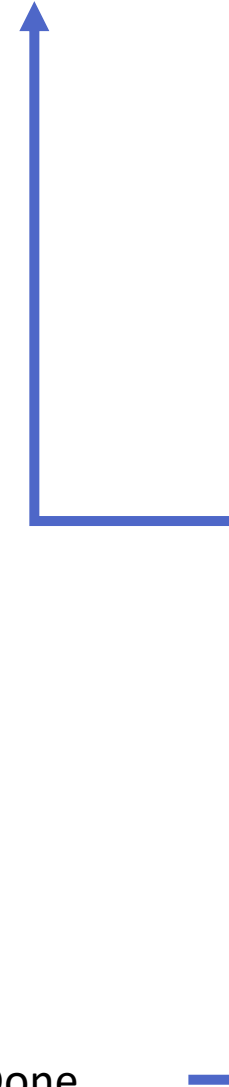
Example  
Programs:



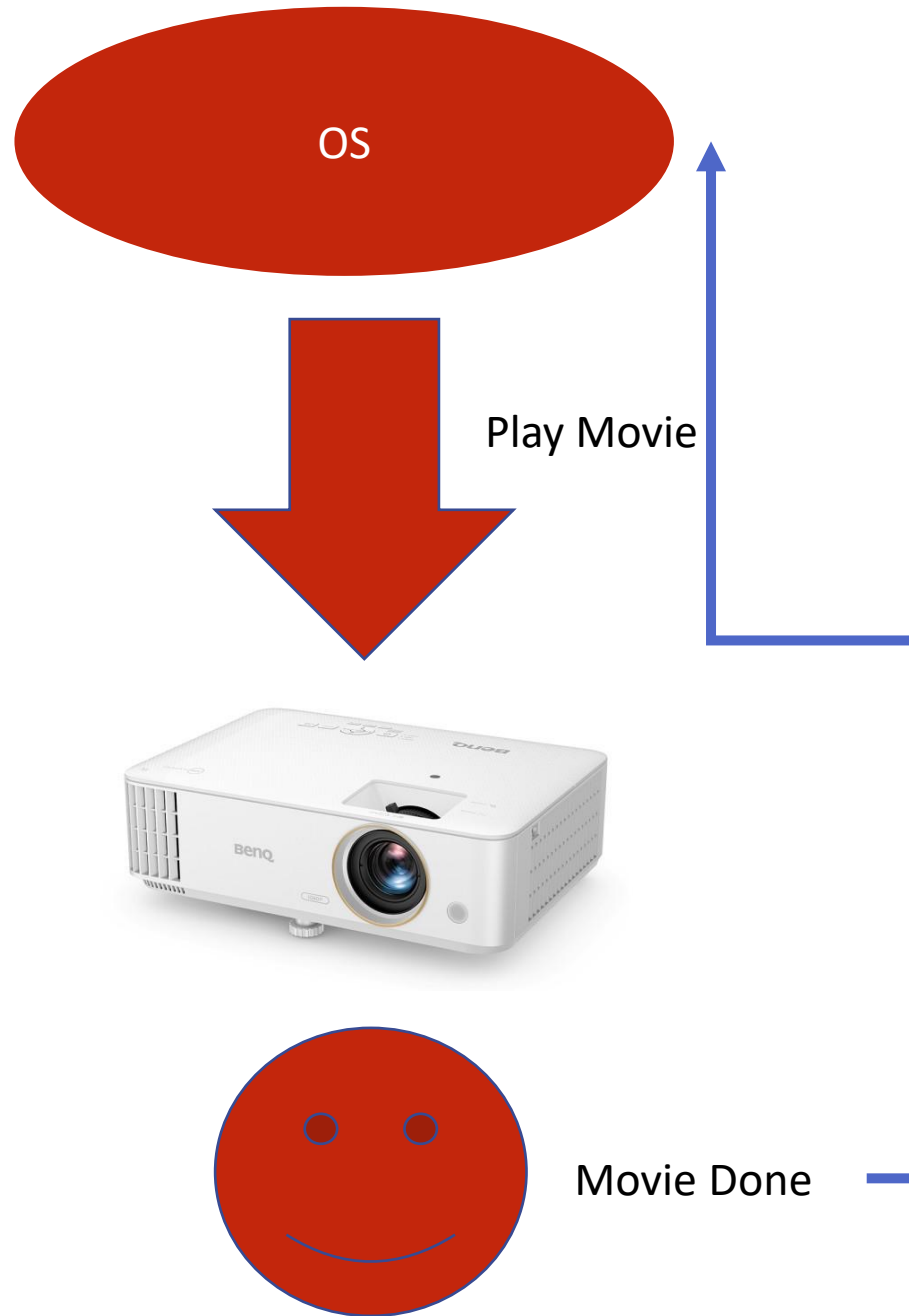
Print



Printer Done



# Example Programs:





# Challenge

## **Today we need to:**

- 1) Print out 120 pages for class
- 2) Watch the newest Netflix movie that came out

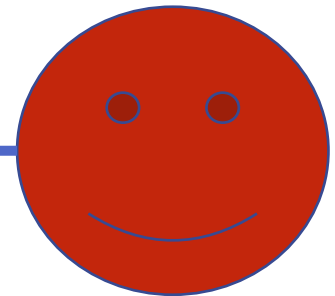
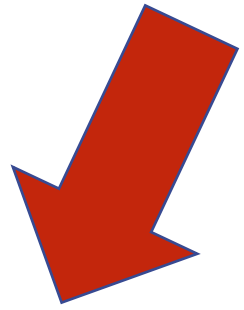
## **Stipulations:**

Each page takes 1 minute to print, giving us approximately two hours of printing

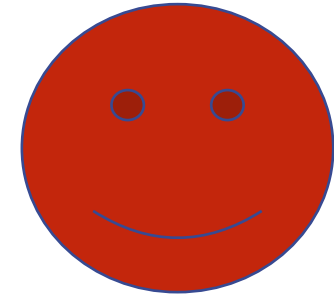
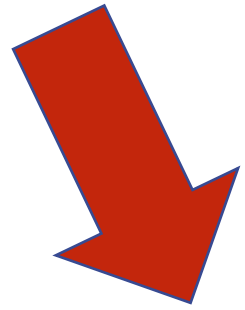
The movie is 2 hours long



TOTAL TIME  
4 HOURS



Printer Done



Movie Done



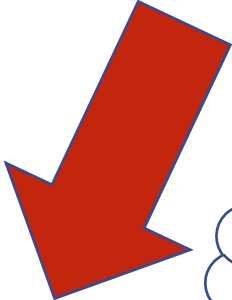
# Challenge

- An engineer's goal is to be more efficient ... how can we make this system better?

# Solution:

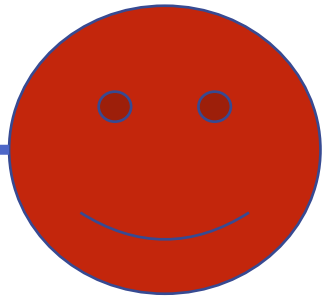
- Allow the OS to "sleep" certain programs, meaning stop monitoring them consistently, and instead check in every once in a while

Hey are you good?

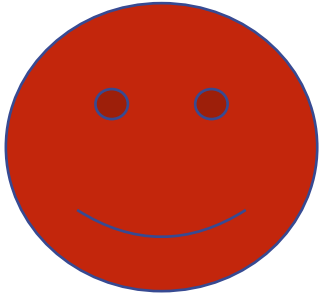


ZZZZZZZZ  
ZZZ

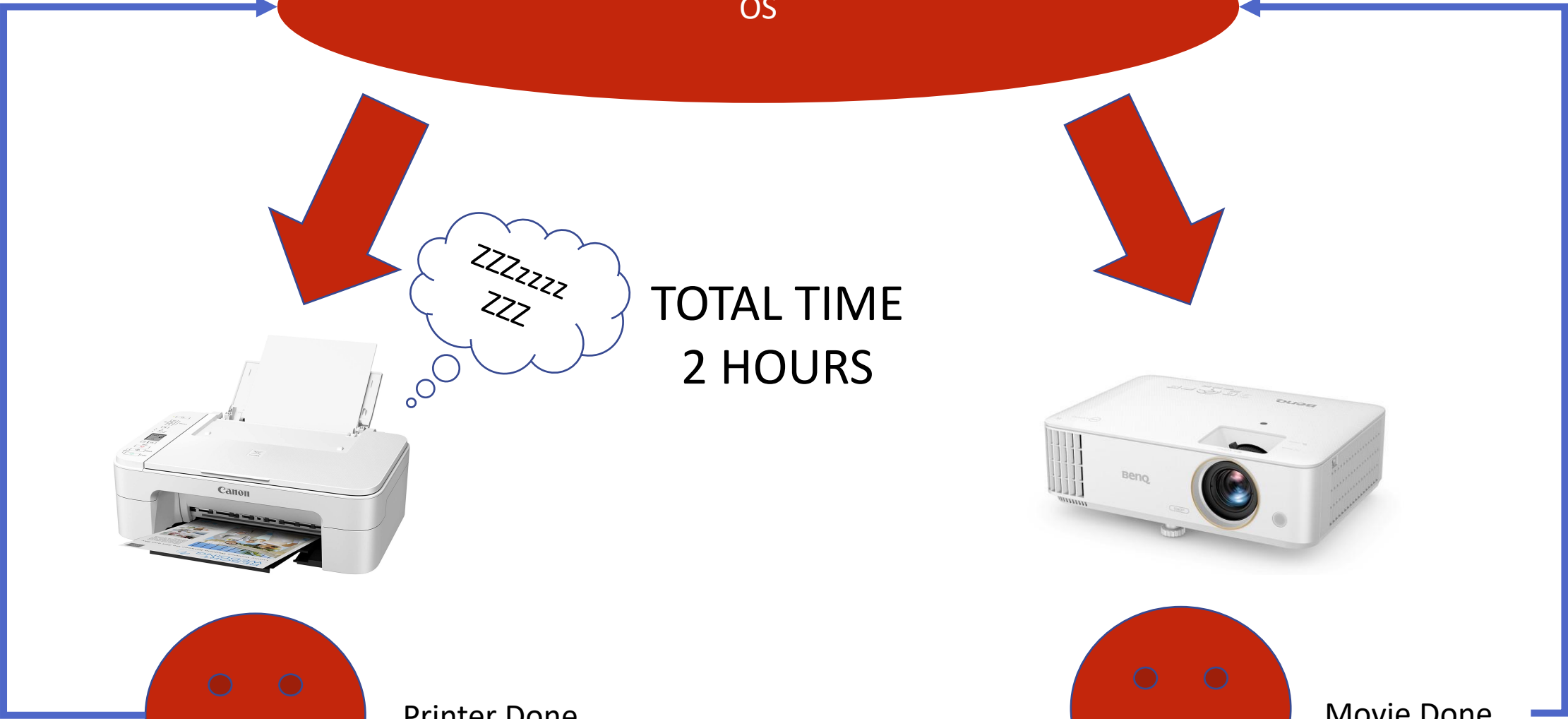
TOTAL TIME  
2 HOURS



Printer Done



Movie Done



This concept is called Multitasking and became used in OS around the 1960s.

Modern computers usually have 4 cores, meaning they have four mini-computers running inside.

Computers are now so great at multitasking, that it can seem that many things are happening at once, when in reality, each core can only run one program at a time



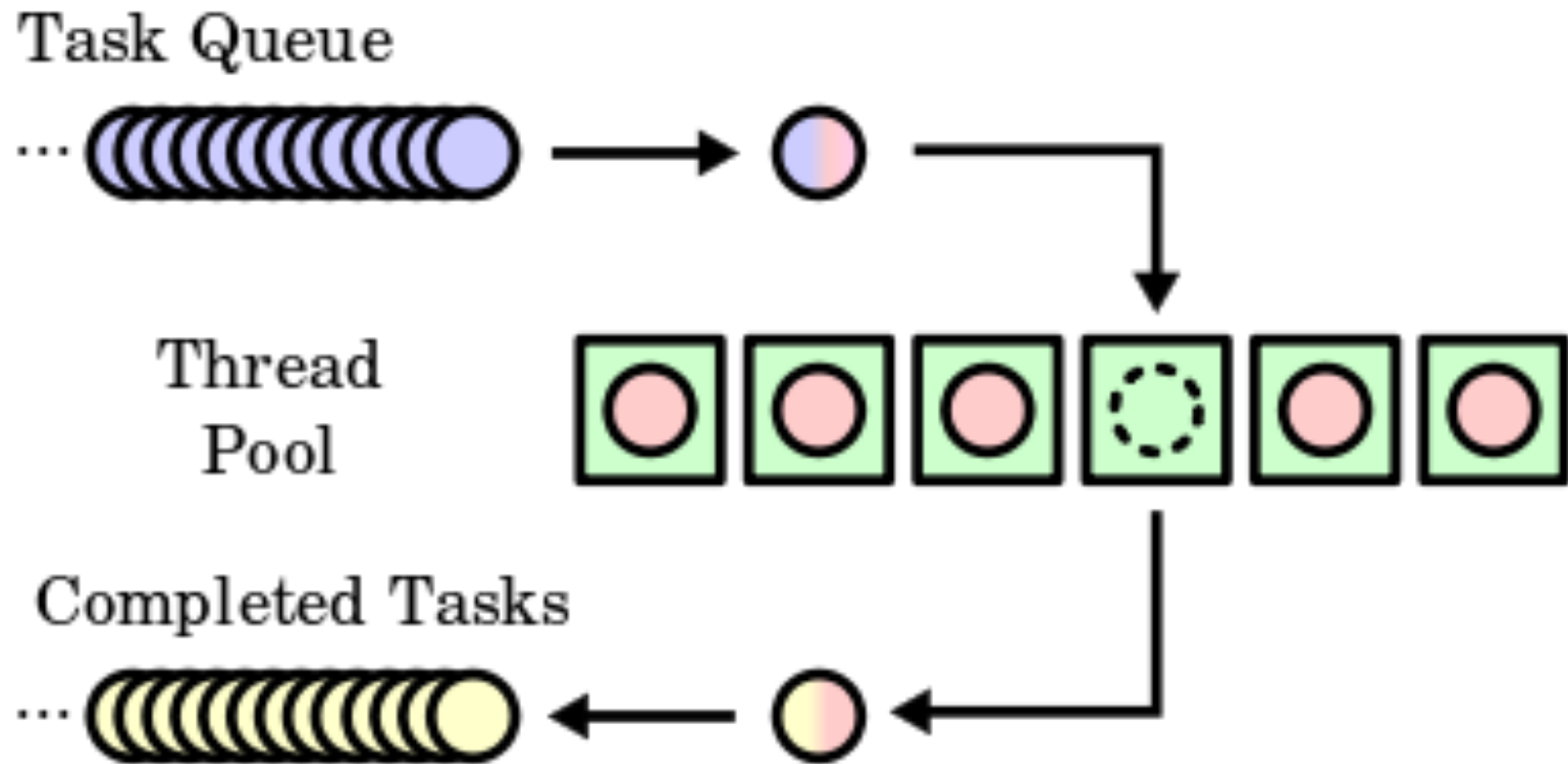
-The programs referenced are represented in systems code by a unit called a thread

-Deciding when certain threads sleep and what priority they run in is done by a Scheduler

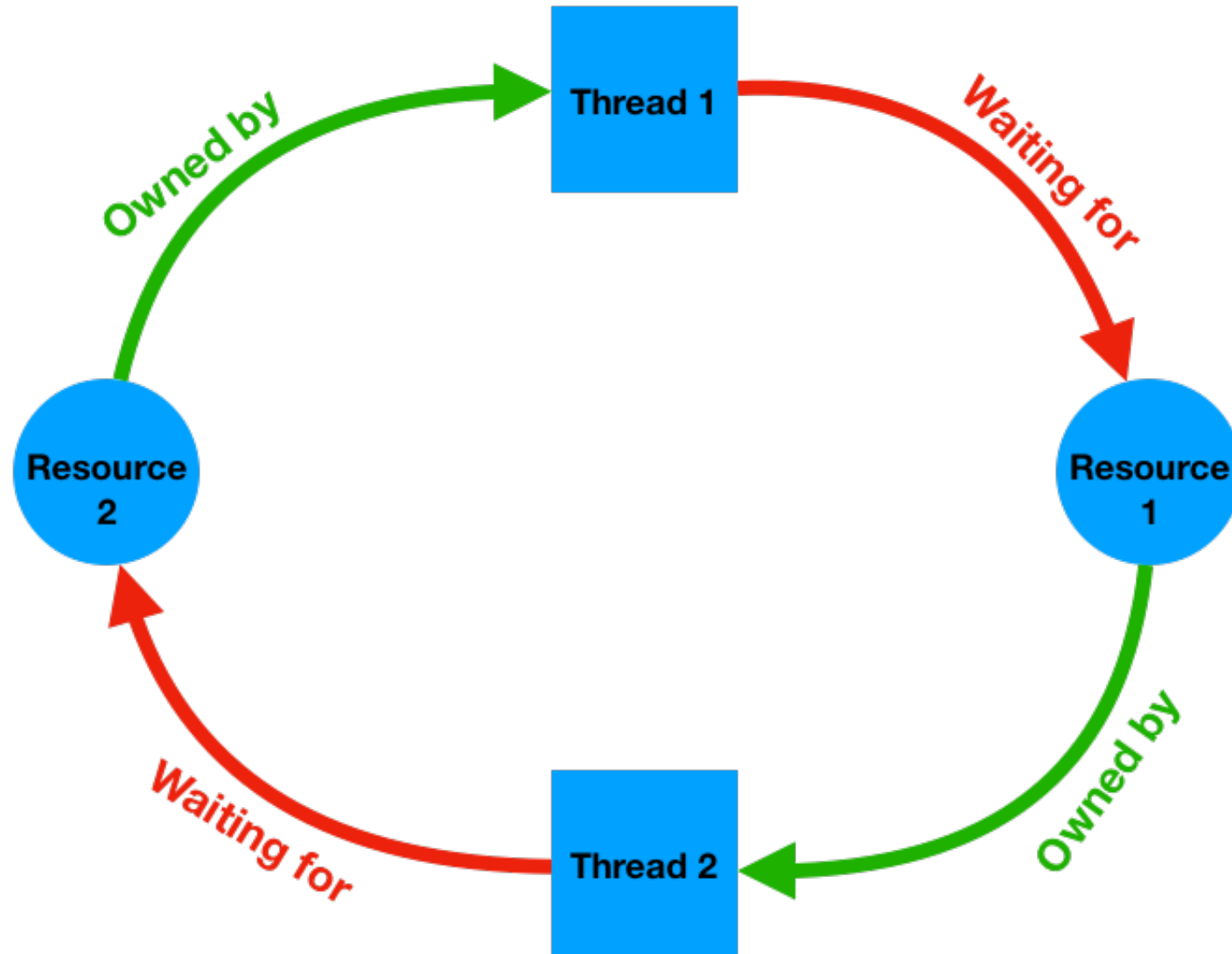
-When one writes code for systems courses, the code itself deals with threads and determines how and when they run along with how they should interreact with different threads.



A scheduler often makes use of a thread pool. A thread pool is the collection of threads which are currently running or are sleeping and waiting to be run again.

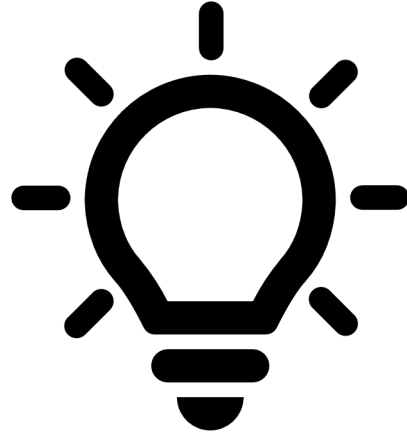


Often the threads will share resources

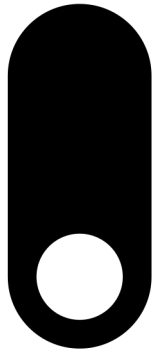




# Sharing Resources



ON



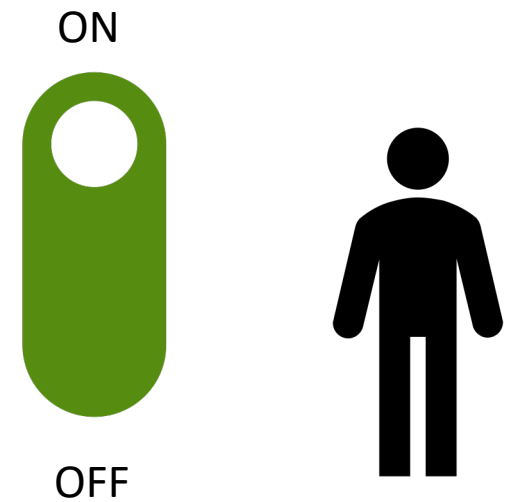
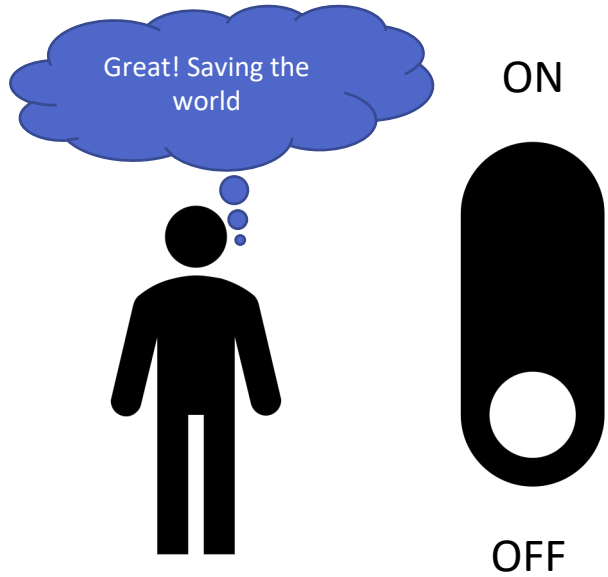
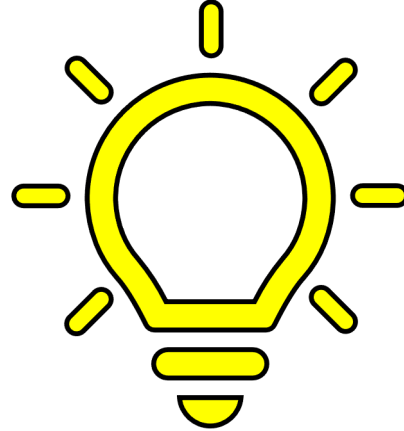
OFF

ON

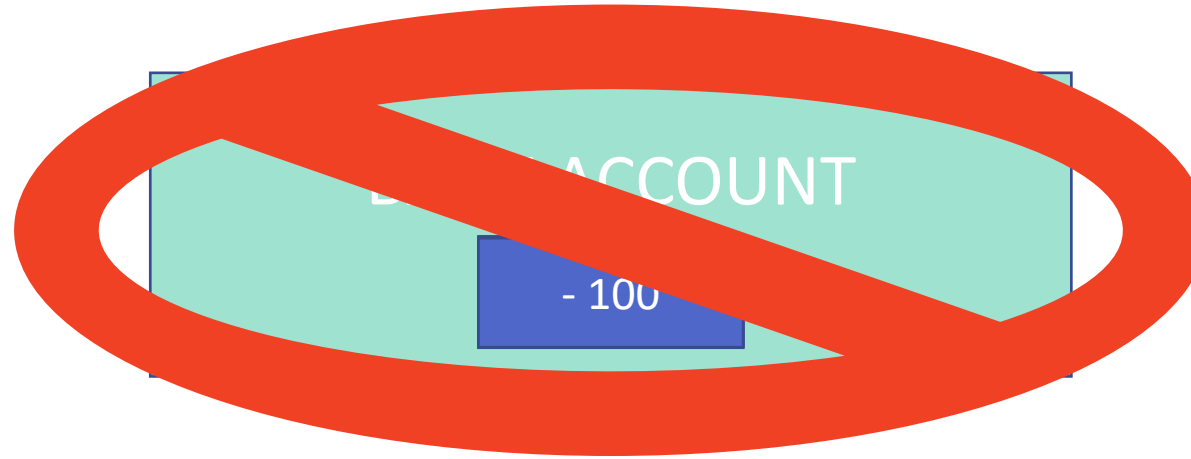


OFF

# Issues with Sharing Resources



# Issues with Sharing Resources



THREAD A

+ 100

Can I take  
out 200?



- 200

THREAD B

+ 200

Can I take  
out 200?



- 200

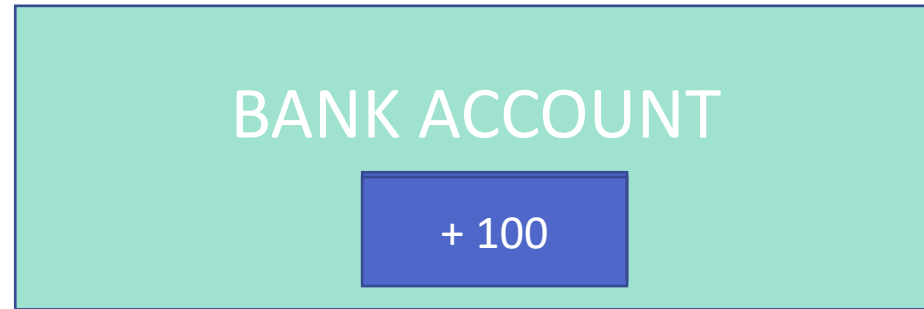
# Challenge:

- How would you change the OS so that this overdraft does not happen?

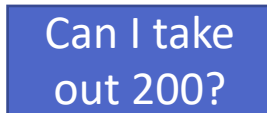
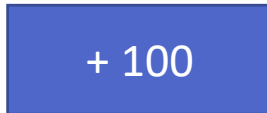
# Solution: LOCK

- Only allow one thread to edit the bank at a time

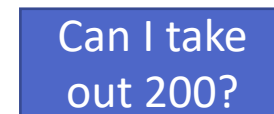
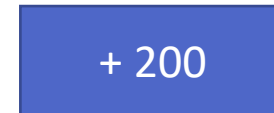
# Solution: LOCK



Sorry! You need to wait for A to be done



DONE



# LOCKS

- Locks can help mitigate issues surrounding shared resources, and are an important aspect of systems programming
- Learning how and when to effectively use locks is an important part of being OS engineer



# Recap:

- History of early computers
- Where the idea of an operating system came from
- Innovations in operating systems such as multitasking, threads, and locks
- Other key concepts in systems include compilers, networks, and databases

# How do these innovation affect our life?





# Q & A

If you enjoyed this lecture, Stanford offers courses such as CS 110 and CS 140 to undergraduate and graduate students, which dive into these topics.