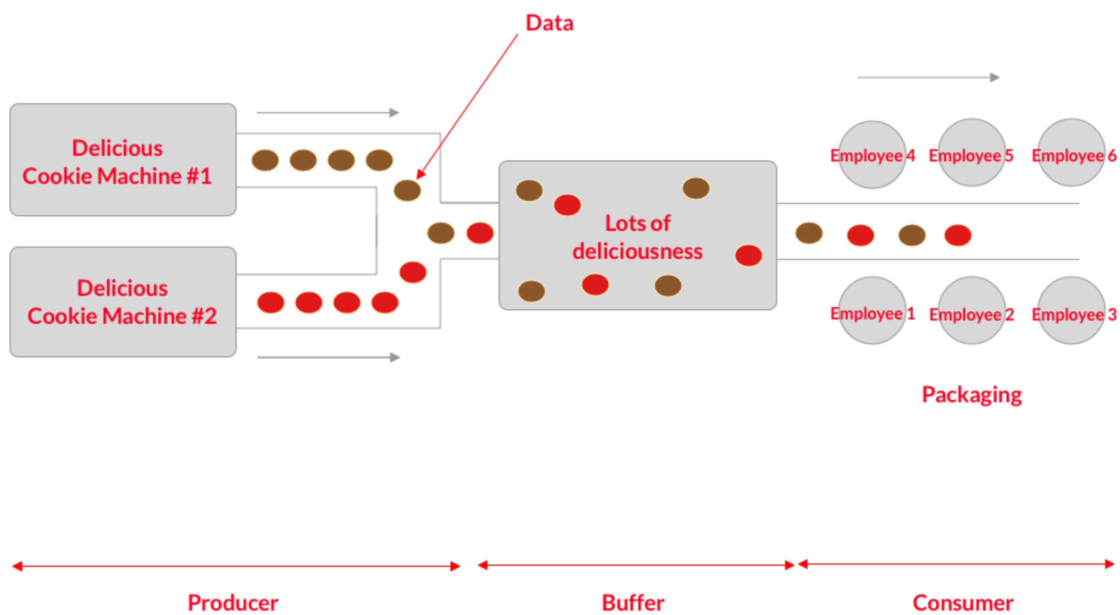
CSC369 Week 3 Notes

Hyungmo Gu

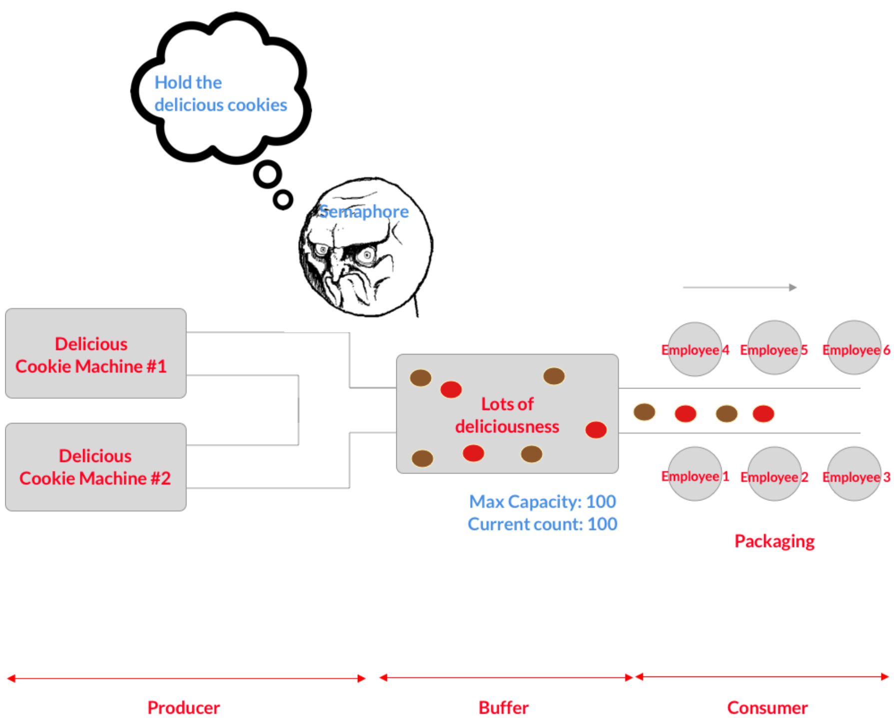
May 24, 2020

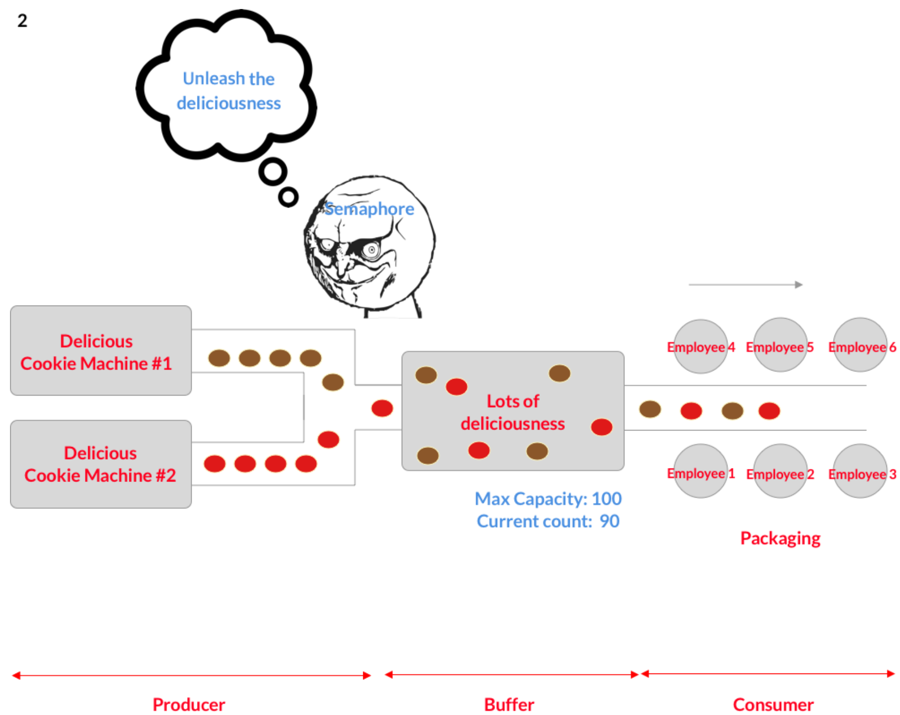
1 Synchronization

- Producer and Consumer Problem
  - Is also known as **bound-and-buffer** problem
  - Achieves synchronization
  - Has two types of processes
    1. **Producer**
       * Produces data
       * Puts data into buffer
    2. **Consumer**
       * Consumes data
       * Removes data from buffer, one piece at a time
  - It's like kimchi factory, or delicious cookie factory :)

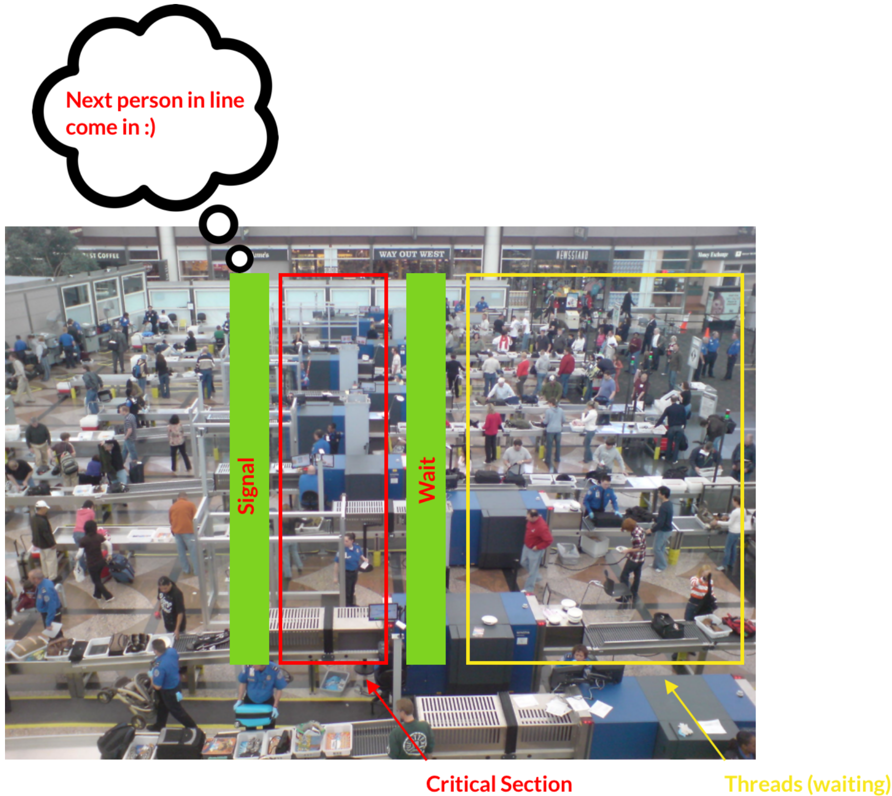


- Semaphore
  - Developed by Dijkstra in 1962.
  - is a signal
    * Uses a non-negative integer variable that is shared between threads [*Note: Need to come back later*]
    * Has two “**atomic**” operations
      1. **Wait** (Also called P, or decrement)
      2. **Signal** (Also called V, or increment)
  - Is easy to understand
  - Is difficult to use
    * One tiny mistake and everything comes to a halt
- Types of Semaphores
  1. Counting Semaphore
     - $count = N \Rightarrow$ Max number of resources
     - $count \uparrow$ when resource added
     - $count \downarrow$ when resource used
     - $count = 0 \Rightarrow$ No resources available $\Rightarrow$ **Wait** until $count > 0$





2. Binary Semaphore
   - *count* = *1* $\Rightarrow$ Unlocked / Available
     * A thread can go in
   - *count* = *0* $\Rightarrow$ Locked / Unavailable $\Rightarrow$ **Wait** until *count* $>$*0*
     * Other threads must wait
   - It's like the security at airport, or the portable bathroom from week 1 notes



- Monitors
  - Is solution to semaphore
    - Is easier to implement
    - Creates less bugs
  - Still not a good solution
    - Usable only in few programming languages
      - C not supported
      - Java supported
    - Is designed for single or multiple CPUS with access to a common memory
    - Not designed for the age of internet
      - Fails with distributed system with each having private memory connected via internet
      - Can't exchange information between machines

2 Intro to Scheduling

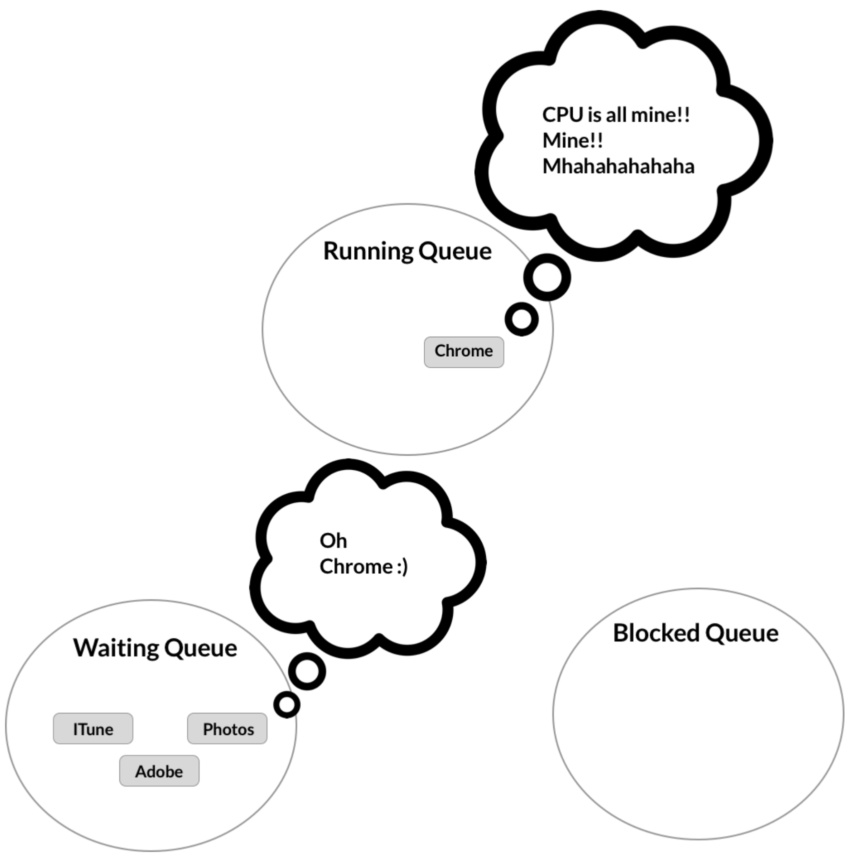
- What is Proceess Scheduling

- Is a process at which allows one process to use the CPU while another is on hold, to make full use of CPU [1]
- This is key to multi-programming

References

1) Study Tonight: What is CPU Scheduling?, link
2) University of Illinois: CPU Scheduling, link

- Types of Scheduling
  - Non-preemtive Scheduling
    * Once the the CPU has been allocated to a process, the CPU keeps the process until it releases either by terminating or by switching to the waiting state [1]



    * e.g Windows 3.1
    * Suitable for batch scheduling
  - Preemptive Scheduling
    * Usually assigns tasks with prioirities [1]
    * Can interrupt for higher priority task [1]
    * Resumes existing task once priority task completes execution [1]

    * Needed in interactive or real time systems
    * Feels like juggling

References

1) Study Tonight: What is CPU Scheduling?, link