Processes:

- Process = job / unit of work
- A process is an instance of a program running
- Each process has own view of machine -
    - Its own address space -
    - Its own open files -
    - Its own virtual CPU (through preemptive multitasking)
- The process defines the address space and general process attributes (everything but threads of execution)
- Processes are the containers in which threads execute Processes become static, threads are the dynamic entities
- Process can be in one of several states -
    - new & terminated at beginning & end of life - running – currently executing (or will execute on kernel return) -
    - ready – can run, but kernel has chosen different process to run
    - waiting – needs async event (e.g., disk operation) to proceed
- Changing running process is called a context switch
- One of the ways to operating system keeps a track of processes is called a process table, which have the processid of the process as the key and a bunch of other values as the element such as a memory address along with
- Processes are expensive compared to threads
- WHen it comes to the scheduling, processes aren't being stored in the PQ or whatever scheduling algorithm is being used, what's being stored in the algorithm is are the threads which have the process id as a value.
- Finally the reason why processes can be good is that in modern computers which tend to be multi core, more cores can be used. Threads can't run on multiple because they don't have a memory address, but processes can. Hence, if 2 processes are on 2 cores being utilized, the execution can happen in parallel doing the exact same task. THerefore, you are breaking down exectution time of the task in half. THis is a big advantage of processes. (this might be a bit wrong)

References:

- http://www.scs.stanford.edu/15wi-cs140/notes/processes.pdf
- https://mathlab.utsc.utoronto.ca/courses/cscc69s15/lectures/week1_2015.pdf
- https://github.com/jay754/pos/blob/master/threads/thread.h