What is a process?

- Process represent a program/application (both words used interchangeably) in execution.
- Process can be thought of a container (or a sandbox) that holds the application and its related data. All resources accessed by the application are also tracked within a process. e.g. Instruction pointer, stack, memory used, files opened etc.

Why the need for such an abstraction?

- The abstraction allows to differentiate among applications and allows multiple applications co-exist in a single system
- Without such abstraction, it would be hard (or even impossible) to run multiple applications concurrently

How does process help in achieving virtualization?

- Process tracks the application execution and so, the process could be paused and resumed at any point in time. This allows the OS to switch among multiple processes on a fewer number of CPUs. Note that this deals with CPU virtualization only (We will talk about memory virtualization later).

What are the interfaces needed in OS to support process?

- Create a new process (fork())
- Destroy the process (kill())
- Wait (wait())
- Miscellaneous: Pause/Resume (Read about signals)
- Current status of a process

What happens during Process Creation?

- Load application code
- Initialize stack, heap
- Open files if needed
- Transfer control to main()

What is a process context?

- Volatile contents (cpu registers like esp, eip, ...) that are lost during a context switch.

Why the need for various process states?

- Applications are not going to execute instruction forever. Sometimes, they are going to read contents from disk or over the network or even wait for a child process. All these operations are really slow most times. To improve CPU utilization, OS swaps such process and allows a different process to run on CPU.
- Process states are used to track the status of each process. (a) Is it ready to run (b) Is it waiting for IO?

What are the different process states?
- Running, Ready (or runnable), blocked
- There could be many more in mainstream OS like Linux, Windows or Mac OS

Context switch
- Switching a new process to run on a cpu and the currently running process is moved out of execution for now (Mechanism)
- Choosing which process to run next is decided by a scheduler - more on this later (Policy)

How does OS maintain the list of processes running in the system?
- Check xv6 code and see how xv6 maintains process list

What is the difference between mechanism and policy and why separating them is beneficial?

Questions
1. Why need for a separate stack and heap within a process?
2. What is the mechanism needed to support cpu virtualization?
3. Imagine if there is no process state (OS does not know if a process is ready or running or waiting). What would happen?
4. What happens when you don't save a process context during a context switch?
5. Understand the fields in struct proc { } in fig 4.3 in chapter 4
6. The homework has some interesting questions - I would recommend to try them.