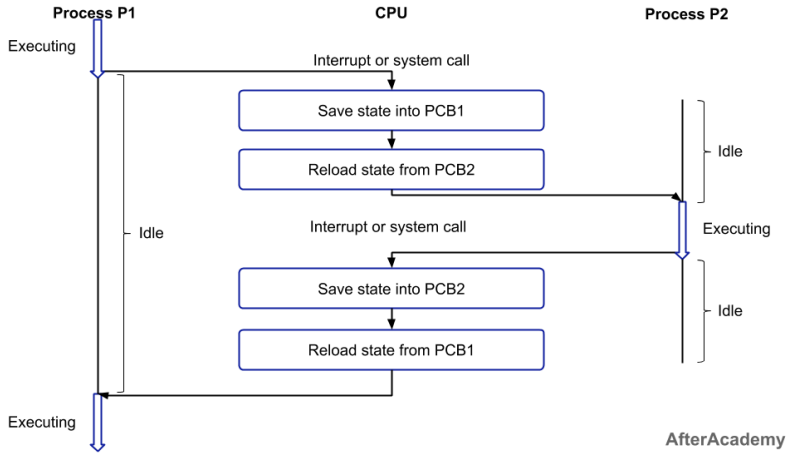
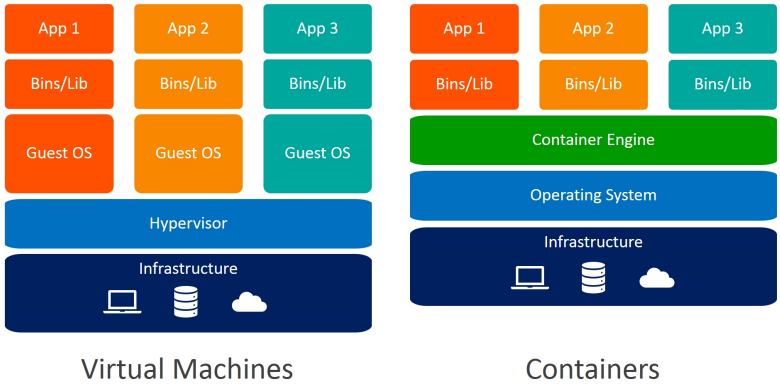
- What is OS?
    1. A special layer of SW provides applications access to HW.
    2. Abstraction of physical resources and provide easier access
    3. Manage and Protect, Isolation
- User, Kernel, Hardware mode
    1. User mode: App, libs, shells, ...
    2. Kernel mode (between user/hardware): System calls, Sigterm, I/O, Scheduling
    3. Hardware: Controller of device, Memory access
- Thread
    1. A single execution context
    2. Including **Program Counters, Registers, Stack Pointer**
    3. When executing: load on processor's register
    4. When suspended: not loaded
    5. Switch: Save PC, REG, SP to memory and load new thread
    6. Just like a light weighted process
    7. **No protection!**
    8. Multithread process: sharing same global variables, heap, code
    9. Private: stack, thread control block (SP, REG, Params)
- Process
    1. Execution environment with **restrictions**
    2. A **protected** address space
    3. A process can have many threads (they share the same address space under a process)
    4. **Isolation**
    5. Process Management
        (a) Fork, exit, exec, ...
        (b) Child process **shared the same state!** (Addr space, memory, ...)
- Multithread
    1. **Semaphore and Mutex:** Semaphore $> 0$ can acquire, mutex = binary semaphore
    2. Concurrency: Handling multiple things at once. Can be done with a single core
    3. Parallelism: Multiple tasks running at the same time
- Context Switching
    1. Thread: Init by CPU
    2. Process:
        (a) Init by OS scheduler
        (b) Flushed address space and load new states
        (c) Heavy cost!




- VM and Container

  1. VM: has guest OS, and a Hypervisor layer captures syscall
  2. Container: is just an application (with Container layer above host OS)




- RTOS

  1. RTOS is **preemptive** and **event-driven**
  2. Needs to monitor the priority == knowing the execution time
  3. Event-driven == switching executing task based on priority