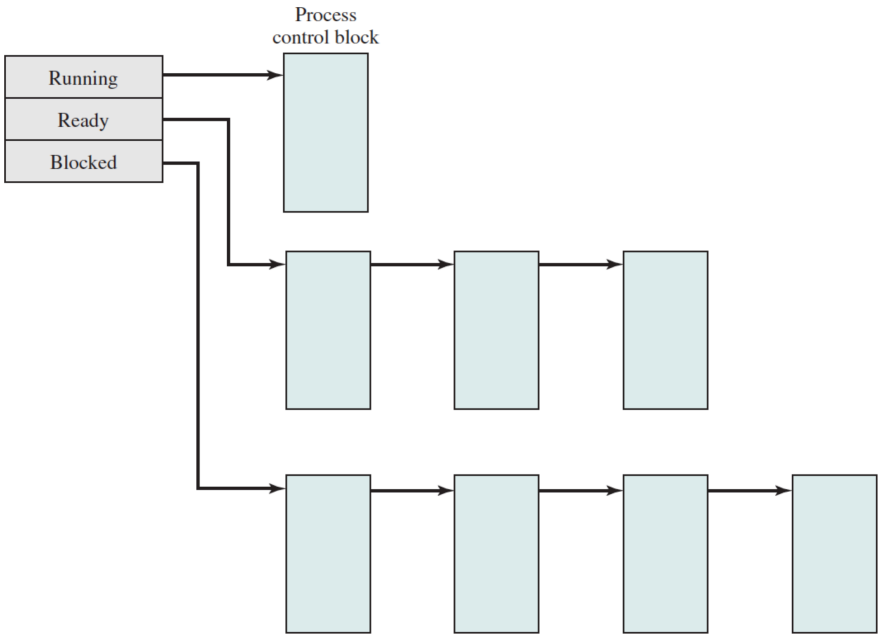
Chapter 3

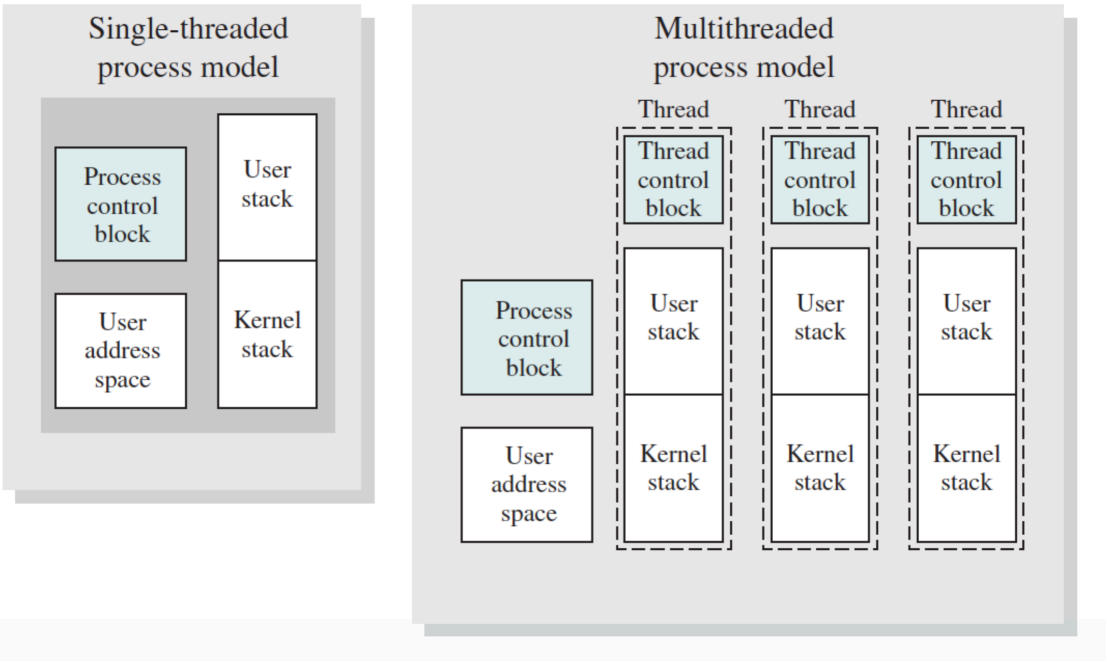
- **Process**
  - Can be:
    - A program in execution; or
    - An entity that can be assigned to and executed on a processor; or
    - A unit of activity with thread of execution, a state, and an associated set of resources
- Process control block:
  - Id
  - State information
    - User registers, control/status registers
    - Program counter
    - Stack pointer
  - Control information
    - State
    - Priority
- Suspended processes:
  - When all processes are blocked, swap some processes for suspended processes stored in secondary memory in order to free main memory
  - Suspended/blocked & suspended/ready states
  - Characteristics:
    - Process is not ready
    - Blocking condition is independent of suspend condition
    - Process was suspended by an agent
    - Process cannot be removed until agent orders it to
- Process image = PCB + user data + user program + stack



- **Process creation**
  - Assign unique id
  - Allocate memory space
  - Initialize PCB
  - Set linkages (e.g. put into ready queue)
  - Create other data structures (e.g. accounting file)
- Execution of OS
  - Non-process kernel
    - Kernel has own region of memory; always operates in privileged mode
  - OS functions within user processes
    - Only mode switch is required for interrupts
  - OS functions as separate process
    - Modular; useful in multiprocessor environment

Chapter 4

- **Threads**
  - Thread = unit of dispatching/execution (wrt. scheduling)
  - Process = unit of resource ownership
  - Multithreaded, single-process – all share the same resources, address space & data



  - Benefits of threads:
    - Takes less time to spawn than a process
    - Less time to terminate than a process
    - Less time to switch between threads within the same process
    - More efficient communication between execution entities
    - Threads within the same process share memory

    - Examples of multithreading uses:
        - Foreground + background work
        - Asynchronous processing
        - Speed of execution
        - Modular program structure
- User-level threads vs. kernel-level threads**:**
    - Advantages of ULT:
        - Switching threads does not require a mode switch to kernel
        - Scheduling can be application specific
        - Can run on any OS
    - Disadvantages of ULT:
        - ULT making a system call blocks all threads within the process
            - In KLT, kernel can schedule another thread of the same process
        - Cannot take advantage of multiprocessing (only application-level multiprogramming within one process)
            - In KLT, threads of the same process can be run on multiple processors
- Amdahl's Law:
    - If f = fraction of program that is inherently parallel (without overhead) and $(1 - f)$ is inherently serial, the speedup due to using N processors is

$$\frac{1}{(1-f)+\frac{f}{N}}$$

Chapter 5

-