Process Management

- **Process**
  - Is a running program
  - Is an active entity
  - Has context and state
  - Is sequentially executed
    - A single instruction is executed on behalf of a process at any time
  - Also called
    - Job on batch systems
    - Task on time-sharing systems
Process State

- **Process state**
  - Running: process' instructions are being executed
  - Waiting: process is waiting for some event (e.g. I/O completion)
  - Ready: process is ready for execution, but must waiting for a processor to become available
Process Context

- Process context
  - Information that allows the OS to resume the execution of a process
  - Process Control Block (PCB)
    - State
    - CPU registers
    - Scheduling info
    - Memory info
    - I/O info
    - Accounting info
  - Process stack
Context Switch

P0 | CPU Scheduler | P1
---|---------------|---
running | interrupt | ready
ready | save state P0 | running
running | select P1 | ready
ready | restore state P1 | running
running | interrupt | ready
ready | save state P1 | running
running | select P0 | ready
ready | restore state P0 | running
Process Address Space

- `code`: program instructions
- `data`: program global data
- `heap`: program dynamically allocated data
- `stak`: program local data
- `OS`: system kernel
Process Creation

- A process is created when another process invokes the corresponding syscall (*e.g.* `fork`)
  - Creator = *parent* process
  - Created = *child* process
  - Child resources can be
    - Inherited from parent
    - Allocated from OS
  - Who creates the first process?
    - Forged by OS initialization procedure

- Process destruction
  - Natural: when a process terminates and calls `exit`
  - Forced
    - By the OS when a process misbehaves (*abort*)
    - By another process (parent) on convenience (*kill*)
Concurrent Processes

- Concurrent processes
  - Resource sharing (concurrence)
  - Speedup with multiple processing elements

- Independent process
  - A sequential program under execution
  - Private context
  - Output depends exclusively from input

- Cooperating processes
  - A parallel program under execution
  - Shared context
  - Output depends also on the relative execution order
Threads

- Threads
  - Also called lightweight process
    - Low creation overhead
  - Execution flow on a task
  - Share task's code, data and resources
  - Has its own stack
  - Traditional process = task + 1 thread