A process is an instance of a running program
- A thread is an execution context
- Process can have one or more threads
- Threads share address space (code, data, heap), open files
- Threads have their own stack and register state

UNIX Process APIs:
- fork() - Creates an exact copy of a process
- waitpid() - Waits for a child process to exit
- exit() - Exit the current process
- kill() - Kill a process
- execve() Loads a program on top of the current process
POSIX Thread APIs:
- `pthread_create()` - Creates a new thread
- `pthread_exit()` - Destroys current thread
- `pthread_join()` - Waits for thread to exit
OS/161 Naming Conventions

- `fork()`, `waitpid()`, `exit()` in assignment 2a
- `execv()` in assignment 2b
- `thread_fork()` creates a kernel thread
  - Five parameters: thread name, process, function pointer, two arguments
- Code deep dive