Programming with Concurrent Processes

Robert M. Dondero, Ph.D.
Princeton University
Objectives

- You will learn or review...
  - What a process is
  - Concurrency via processes
    - How to fork processes
    - How to wait for processes
    - How processes can communicate via pipes
  - Via examples in C
    - Possible in Python (see class Process)
    - Possible in Java (see class ProcessBuilder)
      - But unusual
Processes

- **Program**
  - Executable code

- **Process**
  - An instance of a program in execution
  - A process has its own distinct *context*
Process Context

- **Context** consists of:
  - Process id
  - Address space: TEXT, RODATA, DATA, BSS, HEAP, STACK
  - Processor state: EIP, EFLAGS, EAX, EBX, etc. registers
Unix Process Creation

- At boot-up...
  - Several processes are created automatically
- When you login...
  - Existing process **forks** a process running your login shell (e.g. bash)
- When you execute a command...
  - Bash **forks** a process and **execs** the specified program
- Specified program can **fork** additional processes as desired
Forking Processes

- So, how to fork concurrent processes?
- See forking.c
  - Parent process forks "blue" child process and "red" child
  - Three processes execute concurrently
  - Parent process may exit before child processes
    - Malformed!!!
    - A parent should wait for its children to exit
    - A parent should "reap" its children
Waiting for Processes

- See `waiting.c`
  - Parent waits for children
  - Parent process is "blocked" until both children exit
  - Proper pattern
Inter-Process Comm

- See `parenttochild.c`
  - Parent process writes data to a pipe
  - Child process reads data from a pipe
- Generalization:
  - A Unix pipe is a queue for inter-process comm
- Where have you seen this pattern?
Inter-Process Comm

- See `childtochild.c`
  - Parent process creates "producer" and "consumer" child processes
  - Producer child process writes data to pipe
  - Consumer child process reads data from pipe

- Where have you seen this pattern?
Inter-Process Comm

- See `sortmore.c`
  - Parent process forks two child processes
  - First child executes "sort somefile"
    - With stdout redirected to pipe
  - Second child executes "more"
    - With stdin redirected to pipe
Inter-Process Comm

- Generalization
  - Unix users routinely command the shell to fork communicating concurrent processes
    - E.g. "sort somefile | more"
  - sortmore.c is a hardcoded shell
Summary

- We have covered...
  - What a *process* is
  - *Concurrency* via processes
    - How to *fork* processes
    - How to *wait* for processes
    - How processes can communicate via *pipes*
  - Via examples in C