News

- tshlab due next Tuesday
  - Will go over today
- The cool stuff starts now!
Today

- Processes
- Signals
- tshlab
Processes

- What is a program?
  - A specification
  - No state, just instructions
  - Passive
What is a **process**?

- An instance of a program in execution
- A Great Idea in Computer Science
  - Ubiquitous on multitasking systems
- A fundamental abstraction provided by the OS
  - Single thread of execution (control flow)
  - Full, private memory space and registers
  - Various other state (files, etc.)
Processes

- Four basic process control functions
  - fork()
  - exec()
  - exit()
  - wait()
- Standard on all Unix systems
Processes

- fork()
  - Creates a process
  - Parent and child are exactly alike
    - Except for the return value
    - Equal but *separate*
      - Execution (%eip)
      - Registers
      - Memory
      - File *descriptors*
        - Files are shared
Processes

- `exec()`
  - Replaces process context
  - How programs are run
    - Replace memory image with new program
    - Set up stack with arguments
    - Start execution at the entry point
  - Actually a family of functions
    - `man 3 exec`
Processes

- `exit()`
  - Terminates a process
  - OS frees resources used by the process
  - Tiny leftover data
    - Exit status for the parent
    - Must be freed
Processes

- wait()
  - Waits for a child to change state
  - If a child terminates, the parent “reaps” the child, freeing all resources and getting the exit status
  - Lots of details
    - man 2 wait
Processes

- States
  - Running
    - Executing instructions on the CPU
    - Number bounded by number of CPU cores
  - Runnable
    - Waiting to be running
  - Blocked
    - Waiting for an event
    - Not runnable
  - Zombie
    - Terminated, not yet reaped
Today

- Processes
- Signals
- tshlab
Signals

- Primitive form of interprocess communication
- Notify a process of an event
- **Asynchronous** with normal execution
- Come in several types
  - man 7 signal
- Sent in various ways
  - ^C, ^Z, ^\`
  - man kill
  - man 2 kill
Signals

- **Disposition**
  - Ignore
  - Catch and run signal handler
  - Terminate
  - `man sigaction`

- **Blocking**
  - `man sigprocmask`

- Can’t modify behavior of SIGKILL and SIGSTOP
- Pending signals don’t queue
Signal handlers

- Can be installed to run when a signal is received
- Type is void (*sa_handler)(int)
- Separate control flow in the same process
- Resumes normal control flow on return
- Signal handlers can be called anytime
Today

- Processes
- Signals
- tshlab
First real programming lab

Many ways to do it

- Pick a good way!
- Style will be taken seriously (10 points)

A few tricky parts

- Start soon!

Read the spec!

- Especially the section with all the hints

Read the code!
Hazards

- Race conditions
- Reaping zombies
- Waiting for a foreground job
Thanks!