CS 180 Notes – 2/29/16

- Threads (Overview)
  - If a program has only one thread, it's a single-threaded process
  - If a program has multiple-threads, then it is multi-threaded
  - Threads are scheduled
    - Threads in processes are scheduled
  - Threads are part of a process
    - We have "main" threads and worker threads
  - Unlike processes, threads can communicate with themselves out of the box with no overhead associated with it
    - However, this can lead to massive synchronization issues
  - The Kernel only sees processes, not threads (assuming they're user-level threads)
    - Modern OS's are now providing Kernel-level threads
    - Htop gets thread information
      - Htop's "running" actually means "running" **or** "ready" queue.
      - With a sufficient CPU, this shouldn't make a difference
  - Hardware development is well-ahead of software development as multi-threaded is not a trivial task.
  - Once a program is written as a single-threaded program, it is close to impossible to go back and rewrite it as multi-threaded.
    - Design programs as single or multi-threaded from the outset.
  - Examples of threads in use
    - If you were to modify a .JPG in sections, divide the work into non-dependent threads
    - Downloading a website using *wget*
    - Downloading a *.iso* from multiple mirrors (using a program called *axel*)
  - If you have global variable, you cannot reliably multi-thread (again, synchronization issues).