CS110: API Programming Against the UNIX Filesystem

- **Software layered over hardware, filesystem API calls**
  - First off, we'll take a first pass at understanding how the physical hardware of a disk drive can be made to look like software to store traditional files. I'll leave some details out, but will provide enough detail to be clear how regular files of wildly different sizes can be stored on disk and retreived via the sessions with those files managed by data types like **`FILE *`**, **`ifstream`**, and **`ofstream`**.
  - We'll learn how programmers can interact (either directly, or indirectly though the **`FILE *`** and **`[io]stream`** implementations) with the file system via **system calls**, which are a collection of kernel-resident functions that user programs must go through in order to access and manipulate system resources. Requests to open a file, read from a file, extend the heap, etc, all eventually go through system calls, which are the only functions that can be trusted to touch the system.
  - Today's lecture examples reside in **`/usr/class/cs110/lecture-examples/spring-2017/filesystems`**.
  - The **`/usr/class/cs110/lecture-examples/spring-2017`** directory is a mercurial repository that will be updated with additional examples as the quarter progresses.
    - To get started, type **`hg clone /usr/class/cs110/lecture-examples/spring-2017 cs110-lecture-examples`** at the command prompt to create a local copy of the master.
    - Each time I mention there are new examples, navigate into your local copy and type **`hg pull && hg update`**.

- More importantly, read Sections 1 through 5 of the Saltzer & Kaashoek online textbook, paying special attention to the details in Section 5, which will help you with your first assignment (which goes out on Friday).