An Implementation of Weenix OS

Liam Elberty

Spring 2014

Weenix OS[1] is a Unix-like operating system developed by the teaching staff of Brown University's OS course. As a class project, most of Weenix's internals are stripped out and left for the students to rebuild. I implemented these internals as my capstone project. This involved creating processes, scheduling, disk/TTY drivers, a virtual file system, the S5FS[2] file system, and virtual memory. Single-threaded processes are managed by a simple, non-preemptive, FIFO-based scheduler. Disk and TTY device drivers give Weenix the ability to perform DMA I/O to a virtual disk and basic terminal emulation, respectively. The VFS layer provides an API for most Unix system calls, and S5FS is a very simple file system that implements this API. S5FS attempts no optimizations, and is thus slow but correct. The virtual memory layer features shadow objects with copy-on-write semantics, anonymous objects, and implementations of the fork, mmap, and munmap system calls. My completed version of Weenix is fully-functional, and is able to run simple userland binaries such as the text editor Ed[3].

[1] http://weenix.cs.brown.edu/mediawiki/index.php/Weenix
[2] http://weenix.cs.brown.edu/mediawiki/index.php/S5FS
[3] http://www.gnu.org/software/ed/manual/ed_manual.html