6.828 2014 L1: O/S overview =

Overview

- 6.828 goals
    - Understand operating systems in detail by designing and implementing a small O/S
    - Hands-on experience with building systems ("Applying 6.033")
- What do applications want from an O/S?
    - Abstract the hardware for convenience and portability
    - Multiplex the hardware among multiple applications
    - Isolate applications to contain bugs
    - Allow sharing among applications
- What is an OS?
    - e.g. OSX, Windows, Linux
    - the small view: a h/w management library
    - the big view: physical machine -> abstract one w/ better properties
- Organization: layered picture h/w: CPU, mem, disk kernel: [various services] user: applications, e.g. vi and gcc
    - we care a lot about the interfaces and internel kernel structure
- What services does an O/S kernel typically provide?
    - processes
    - memory
    - file contents
    - directories and file names
    - security
    - many others: users, IPC, network, time, terminals
- What does an O/S abstraction look like?
    - Applications only see them via system calls
    - Examples, from UNIX / Linux:

```
fd = open("out", 1);
write(fd, "hello\n", 6);
pid = fork();
```

- Why is O/S design/implementation hard/interesting?
    - the environment is unforgiving: weird h/w, no debugger
    - it must be efficient (thus low-level?) ...but abstract/portable (thus high-level?)
    - powerful (thus many features?) ...but simple (thus a few composable building blocks?)
    - features interact: `fd = open(); ...; fork()`
    - behaviors interact: CPU priority vs memory allocator.
    - open problems: security, multi-core
- You'll be glad you learned about operating systems if you...
    - want to work on the above problems
    - care about what's going on under the hood

    - have to build high-performance systems
    - need to diagnose bugs or security problems

Class structure

- See web site: http://pdos.lcs.mit.edu/6.828

- Lectures

    - basic O/S ideas
    - extended inspection of xv6, a traditional O/S
    - several more recent topics
    - xv6 programming to re-inforce xv6 understanding
- Lab: JOS, a small O/S for x86 in an exokernel style

    - you build it, 5 labs + final lab of your choice
    - kernel interface: expose hardware, but protect -- no abstractions!
    - unprivileged library: fork, exec, pipe, ...
    - applications: file system, shell, ..
    - development environment: gcc, qemu
    - lab 1 is out
    - make grade
- Code review

- Two quizzes: one in class hours, one in final's week

Shell and system calls

- 6.828 is largely about design and implementation of system call interface. let's start by looking at how programs use that interface. example: the Unix shell.

- the shell is the Unix command UI

    - typically handles login session, runs other processes
    - you saw it in 6.033: http://web.mit.edu/6.033/www/assignments/handson-unix.html
    - the shell is also a programming/scripting language
    - look at some simple examples of shell operations, how they use different O/S abstractions, and how those abstractions fit together. See Unix paper if you are unfamiliar with the shell.
- Simplified xv6 sh.c

    - See chapter 0 of xv6 book
    - Basic organization: parsing and executing commands (e.g., ls, ls | wc, ls > out)
    - Shell implemented using system calls (e.g., read, write, fork, exec, wait) conventions: -1 return value signals error, error code stored in `errno`, `perror` prints out a descriptive error message based on `errno`.
    - Many systems calls are encapsulated in libc calls (e.g., fgets vs read)

- Trace system calls $ ls
    - On OSX: sudo dtruss ./a.out (where a.out is the compiled sh.c)
    - On Linux: strace ./a.out

- what does fork() do? copies user memory copies process kernel state (e.g. user id) child gets a different PID child state contains parent PID returns twice, with different values
- parent and child may run concurrently (e.g., on different processors).
- what does wait() do? waits for any child to exit what if child exits before parent calls wait?

- what are file descriptors? (0, 1, 2, etc. in read/write) echo.c

- what is i/o redirection? echo.c How would you implement ">" in sh.c

- what are pipes? (ls | wc) pipe1.c pipe2.c How would you implement them in sh.c?

- Homework assignment for shell Edit By MaHua