Intro Lecture

I. Administrative Matters

Instructors: Eric Brewer and Joe Hellerstien

Eric Brewer

- PhD MIT, 1994
- Internet Systems, Mobile computing, Security, Parallel computing
- Founded Inktomi, Federal Search Foundation
- Contact: brewer@cs.berkeley.edu

Joe Hellerstien

- PhD, Wisconsin, 1995
- Databases, declarative networking
- Contact: hellerstein@cs.berkeley.edu

Course prerequisite: no **entrance exam** this year, but please review undergrad material

Traditional goals of the course:

- Introduce a variety of current OS research topics
- Teach you how to do OS research

New goals:

- Common foundation for OS and database research
- Motivation: OS and DB communities historically separate, despite much overlap in goals and ideas. We often use independent vocabulary and have largely separate "classic" papers that the other community typically hasn't read (especially OS people not reading DB papers, since a DB is "just an application")
- Part 1 of a year-long advanced systems course
- 262A satifies software breadth requirement by itself

Research = analysis & synthesis

- Analysis: understanding others' work - both what's good AND what's bad.
- Systems research isn't cut-and-dried: few "provably correct" answers.
- Synthesis: finding new ways to advance the state of the art.

Lecture 1

2 parts to course:

- literature survey
- term project

Will not cover basic material.

Analysis: literature survey: read, analyze, criticize papers.

- All research projects fail in some way.
- Successful projects: got some interesting results anyway.

In class: lecture format with discussion

Synthesis: do a small piece of real research.

- Suggested projects handed out in 3-4 weeks
- Teams of 2-3
- Poster session and "conference paper" at the end of the semester
- Usually best papers make it into a real conference (with extra work)

Class preparation:

- Reading papers is hard, esp. at first.
- **Read before class.**

Homework: *brief* summary/paper

- 1/2 page or less
- 2 most important things in paper
- 1 major flaw

Course topics: (much of systems design is about sharing)

- File systems
- Virtual memory
- Concurrency, scheduling, synchronization
- Communication
- Multiprocessors
- Distributed Systems
- Transactions, recovery, & fault-tolerance

Lecture 1

- Protection & security
- OS structure
- "Revealed truth" — overall principles

Grad class ⇒ material may be controversial (hopefully!)

Lecture format: cover 1-2 papers plus announcements

Grading: 50% project paper, 15% project demo, 25% midterm, 10% paper summaries

- can miss 3 summaries without penalty, no reason needed or wanted
- Summaries: < 1/2 page, at least one criticism, summary should be at most half

Reading list:

- No textbook
- Almost everything is online, otherwise I'll distribute copies in class.
- "Warm up" paper: "UNIX timesharing"
- Reading for next time: System R paper (in the Red Book)

II. The UNIX Time-Sharing System

"Warm-up" paper. Material should mostly be a review.

Classic system and paper: described almost entirely in 10 pages.

Key idea: elegant combination of a few concepts that fit together well.

System features:

- time-sharing system
- hierarchical file system
- device-independent I/O
- shell-based, tty user interface
- filter-based, record-less processing paradigm

Version 3 Unix:

- ran on PDP-11's
- < 50 KB
- 2 man-years to write
- written in C

Lecture 1

File System:

- ordinary files (uninterpreted)
- directories (protected ordinary files)
- special files (I/O)

Directories:

- root directory
- path names
- rooted tree
- current working directory
- back link to parent
- multiple links to ordinary files

Special files:

- uniform I/O model
- uniform naming and protection model

Removable file systems:

- tree-structured
- *mount*'ed on an ordinary file

Protection:

- user-world, RWX bits
- set-user-id bit
- super user is just special user id

Uniform I/O model:

- open, close, read, write, seek
- other system calls
- bytes, no records

File system implementation:

- table of i-nodes
- path name scanning

Lecture 1

- mount table
- buffered data
- write-behind

I-node table:

- short, unique name that points at file info.
- allows simple & efficient fsck
- can't handle accounting issues

Processes and images:

- text, data & stack segments
- process swapping
- pid = fork()
- pipes
- exec(file, arg1, ..., argn)
- pid = wait()
- exit(status)

The Shell:

- cmd arg1 ... argn
- stdio & I/O redirection
- filters & pipes
- multi-tasking from a single shell
- shell is just a program

Traps:

- hardware interrupts
- software signals
- trap to system routine