Welcome to graduate operating systems! This course will cover an exciting range of materials from the broad field of operating systems, including basic operating system structure, communication, memory management, reliability, file systems and storage, virtual machines, security, and manageability. We will examine influential historical systems and important current efforts, extracting lessons both on how to build systems as well as how to evaluate them.

Introductions

Poll questions:

- Who is a CS student? ECE? Something else?
- Who is first year? Second year? Something else?
- Who sees systems as their primary interest? DB? Architecture? Networking?

Long view of operating systems:

1. What are they?
2. What do they do?
3. How are they built?
4. Why do we have Linux and Windows?
5. What services do they provide?
6. What are core problems?
7. What are the general techniques for solving these problems?
8. How do you evaluate them?

Class Organization

Focus on major function of operating systems:

- structure/abstractions
    - Protection
    - Extensibility
- Subsystems:
    - Processors: scheduling, threads, communication
    - Memory
    - Storage
    - I/O
- Goals
    - Security
    - Reliability
    - Manageability

- Power

Question: are there particular topics people want to spend more time?

Question: my nature is to look back at the original papers on a topic - more meat. We can sacrifice some breadth of knowledge to read a modern paper if people would like.

Project

In this course, you will be completing two projects: a warm-up project and a main project. More information will be available on the project page. The main project will be done in groups of 3 or 4, with the same grade for all participants.

Exams

There will be two midterm exams, each one covering one half the semester and of equal grading weight. You should be familiar with the papers and be able answer the questions from the reading section about those papers. Exams will be open note, open book, closed internet.

The midterm will be in the evenint

Honesty

There are ample opportunities to cheat:

- Old reviews are available on line – I reuse papers and wouldn't recognize it if you reused one
- You can copy code for the mini project from your partners
- You can download code off the internet (and text) for your project

This class is for you – if you cheat, you won't learn as much. In grad school, grades don't matter that much, so it isn't worth cheating. If you feel stressed about your grades and are tempted to cheat, please talk to me instead

Grading

- 25% Reading and class participation
- 15% First exam
- 15% Second exam
- 10% Warm-up Project
- 35% Project

Late assignments will be docked 10% per day late. If you know you will be gone for a conference or interview, contact me early. In general, I give enough time for assignments that they can be done early.

Discussion Board

We will be using the Piazza discussion system for this class. If you post there, other classmates will see your post and can answer. I will also monitor it. This is a good way

to contact the rest of the class or get questions answered that everybody may be interested in. I've enrolled all of you already

Reading

The reading schedule for this course will be intense. The readings are grouped into four major categories: complexity, performance, reliability, and security. Within each category, we will read papers that address that issue in the context of different portions of an operating system.

You should form a discussion group for talking about the papers. You should have three to five people in your group, and discuss each paper sometime before class meets. You can use the Piazza discussion board to help find a group member, or make a group after class today or Thursday.

It is up to you if you want to meet just once a week or twice a week before each class, but you should discuss each paper. When you have formed a group, please send me an email with a list of group members.

When multiple papers are due, follow directions on web site as to which one to review

When discussing each paper, you are encouraged to consider the following questions: As you read, here are some questions you should consider:

1. What problem are the authors trying to solve?
    - Why was the problem important?
    - Why was the problem not solved by earlier work?
2. What is the authors solution to the problem?
    - How does their approach solve the problem?
    - How is the solution unique and innovative?
    - What are the details of their solution?
3. How do the authors evaluate their solution?
    - What specific questions do they answer?
    - What simplifying assumptions do they make?
    - What is their methodology?
    - What are the strengths and weaknesses of their solution?
    - What is left unknown?
4. What do you think?
    - Is the problem still important?
    - Did the authors solve the stated problem?
    - Did the authors adequately demonstrate that they solved the problem?
5. What future work does this research point to?

You should be prepared to discuss these questions in class.

1. Good approach:
    i. Skim paper for major ideas
    ii. Read in depth to understand:
        1. What was the motivation of the paper?
        2. What are they key problems they are solving?
        3. What is their solution? How does it really work?
        4. How do they evaluate their solution?
    iii. Things to think about:
        1. What are the assumptions? Are they valid?
        2. Many papers are historic; where did they go wrong? Why?

Responsibilities

For every lecture, you will be reading one or two research papers. For each paper, you have four responsibilities:

1. Read the paper
2. Discuss the paper with your group
3. Submit a short writeup about the paper

Summarizing

I will ask each group to prepare a short summary of the paper. If you are the group, this means reading not just the paper, but other materials about the paper – learn as much as you can about the paper and what people think of it and why it is significant. You can read other students' reviews, to see what they think

In class, you will talk for 10 minutes about the significance, content of the paper.

- What do we learn from the paper
- What is wrong about the paper
- What are the contributions of the paper?

Writeup

Before 9:00 am on day of class, please post your review to the blog at:

Your posting should contain:

1. A one or two sentence summary of the paper
2. A one description of the problem they were trying to solve
3. A summary of the contributions of the paper
4. The one or two largest flaws in the paper

5. Additional question based on the portion of the course (listed in discussion blog)

The writeup should not be more than a page in length. Late writeups will receive a zero grade.

Writeup Grading

What I'm looking for:

- Does the review include all sections (summary, problem, contributions, flaws, relevance)
- Are all assertions backed up (e.g. "X is a bad idea" is not acceptable, but "X is a bad idea because Y") is acceptable
- Is the review concise? The summary should be a few sentences and give the essence of the design in the paper, not the problem. (E.g., "This paper is about how to build a multiprocessor operating system" is not acceptable, but "This paper is about building a multiprocessor operating system by layering abstractions that mask the existence of multiple processors" is acceptable)
- Did the student understand the material? Are there factual flaws in the review? For example, if the paper defines a term, does the student use it appropriately? As another example, if students state that a paper is relevant because modern operating systems do things the same way, is that true?

Intro to OS:

1. What is an OS?

1. What is an OS?
    1. (ASK CLASS)
    2. Wikipedia: "In computing, an operating system (OS) is the system software responsible for the direct control and management of hardware and basic system operations. Additionally, it provides a foundation upon which to run application software such as word processing programs and web browsers."
    3. Dictionary.com: "<operating system> (OS) The low-level software which handles the interface to peripheral hardware, schedules tasks, allocates storage, and presents a default interface to the user when no application program is running."
    4. Computerhope.com: "An Operating System, or OS, is a software program that enables the computer hardware to communicate and operate with the computer software. Without a computer Operating System, a computer would be useless."
    5. "The collection of functions which make available computing services to people who require them"
    6. "The purpose of an OS is to provide an environment where a user can execute programs. The primary goal of an OS is thus to make the computer system convenient to use. A secondary goal is to use the computer hardware in an efficient manner."
    7. Alan Creak (prof, Auckland University) "The OS is the collection of functions, mostly implemented in software, that people want to use but which are not

implemented anywhere else. In effect, the OS does everything that everyone wants, but no one else is prepared to do."

8. Bill Wulf (Hydra): "An OS provides a 'virtual machine' which is more hospitable than the base hardware for two reasons: (a) it make available certain 'virtual resources' such as files, directories, virtual memory, etc. absent from the base hardware. (b) it masks certain unpleasant hardware features, such as interrupts, from the user and maps them into more acceptable ones, such as P-V synchronization primitives
9. Bill Wulf (Hydra): an OS manages the physical resources of the computer, suchas primary memory, processor, channels, etc. so as to improve their utilization
10. Peter Denning: "A computer system may be defined in terms of the various supervisory and control functions it provides for the processes created by its users: 1) creating and removing processes, 2) controlling progress of processes, 3) acting on exception conditions arising during the execution of a process, 4) allocating hardware resources among processes, 5) providing access to software resources, e.g. files, editors, compilers, assemblers …, 6) providing protection, access control, and security for information, and 7) providing interprocess communication where required. These functions must be provided by the system because they cannot be handled adequately by the processes themselves. The software that assists the hardware in implementing these functions is the OS.
11. Major abstractions
    i. Resource manager:
        1. Efficiently multiplex cpu, memory, devices
    ii. Protection
        1. Isolate and safely share between programs
    iii. Hardware services
        1. Simplify access to cpu, memory, devices
            1. E.g. automatic swapping vs. overlays
            2. Stream/message based device access vs. dma/programmed i/o
    iv. Higher-level services
        1. Communication
            1. Ordered, reliable message delivery vs. unreliable packets (TCP/IP)
        2. Storage
            1. Hierarchical, named storage vs. blocks (file systems
        3. Protection
            1. Safe sharing / communication
        4. Users / programs use abstract virtual machine instead of real hardware -> masks hardware differences
    v. OS provides interfaces to:
        1. Hardware (e.g. device drivers)
        2. Applications (e.g. system calls)
        3. User (e.g. cmd line, gui)
        4. System manager (e.g. registry, scripts. .conf/.ini files)
        5.

2. Why are OS's hard to build?
    1. Managing competing demands under finite resources
    2. Managing unknowable / unpredictable demands

3. Widely differing usages (e.g. Linux used on wristwatch, in google cluster, on ascii Q supercomputer, on desktop)
4. Balance performance and programmability (for app and kernel devs)
    i. Abstract away complexity of hardware
    ii. Maintain performance of hardware despite abstraction

3. Key themes in Operating system development
    1. Complexity: simplifying, adding structure, or removing complexity from system to make it easier to understand / adapt
    2. Performance: making existing applications run faster by removing unnecessary code/ providing better access hardware / better compensation for HW problems
    3. Reliability: making it work under failure conditions
    4. Security: preventing bad things from happening
    5. Manageability: removing human element / reducing human element in operations
        1.