CS/SE3SH3 Operating Systems

Sanzheng Qiao

Department of Computing and Software

November, 2012

Course web page

My home page:

www.cas.mcmaster.ca/˜qiao

- announcements
- my address, office hours
- TA emails
- lecture notes, text book, reading materials
- assignments, project mechanics
- sample exams

About this course

Design and implementation principles of modern operating systems.

Evaluation scheme

- Three programming assignments 24%
- Midterm (closed book, 50 min) 20%
- Final (open book, 2 hrs) 56%
- A deferred midterm will be offered

About this course

What do you expect from the course (operating systems)?

About this course

What do you expect from the course (operating systems)?

Principles of modern operating systems.

- Basic concepts (threads, processes, synchronization). Compare existing and proposed solutions.
- Major components of an operating system (process management, memory management, input output, file system, network).
- Analyze a simple instructional OS (Nachos) in enormous detail to get a feel for implementations of the concepts in a real system.

Major components

- Process management: OS allows several users to be working at the same time, as if each had a private personal computer. Or, one user can be doing many things at the same time.
  Operations: create, delete, join, go-to-sleep
- Memory management: OS coordinates the uses of memory shared by several processes. It swaps information back and forth between disk and main memory so the system can run even the total memory needed by all processes is greater than the total size of main memory.
  Operations: allocate, deallocate, map processes to the memory.

Major components

- Input output: OS keeps the CPU busy while an I/O device is working.
  Operations: I/O interrupts, buffering, device drivers.
- File system: OS coordinates the usage of space for files so that all files can fit on the same disk.
  Operations: create/delete files and directories, open/close files, read/write files and directories, map files and directories to disk.
- Network: OS allows groups of workstations to communicate and work together.
  Operations: Send and receive messages to and from network.

What is an OS?

Two functions:

- A set of procedures that *coordinates* resources (CPU, memory, I/O devices, etc.) and enables a group of people to share a computer (computers) *efficiently*.
- A library of subroutines that provides a friendly *interface* between users and a computer (computers).

What is an OS?

Two functions:

- A set of procedures that *coordinates* resources (CPU, memory, I/O devices, etc.) and enables a group of people to share a computer (computers) *efficiently*.
- A library of subroutines that provides a friendly *interface* between users and a computer (computers).

Most of this course will deal with the coordination aspect.

A question

What happens in operating systems (in two machines):

after a mail is composed and the `send` button is clicked and before the mail is send to the unreliable physical network

after the mail arrives and before the mail is stored in a user mailbox

A question

What happens in operating systems (in two machines):

after a mail is composed and the `send` button is clicked and before the mail is send to the unreliable physical network

after the mail arrives and before the mail is stored in a user mailbox

The third assignment (requires the first two assignments).

Why is studying OS interesting?

- Combining many things such as languages, hardware, data structures, algorithms.
- Creating illusions like a wizard making computer appear to be more than it really is.
    - Memory as large as disk
    - Single processor running multiple programs concurrently

Why is studying OS interesting?

- Managing large, complex systems (including users and programs), often in conflict needs (interactive vs. batch, friendly vs. secure, small jobs vs. big jobs).
- Protecting users from each other; security issues pose tough problems especially in networking environment.
- Presenting general techniques. So you can learn and apply them elsewhere.

Discipline and Craftsmanship

How to build large software systems in a reasonable amount of time, and make them work reliably?

Programming is a craft: something like a high-tech form of silversmithing. To make big systems work, it takes a tremendous amount of discipline and structure. Unfortunately, many of us aren't too good at this stuff. Fortunately, if you are, you'll have a tremendous advantage.

Discipline and Craftsmanship

Rule 0: simplicity. It’s easy to make things complicated, harder to make them simple. Don’t accept complexity.

Discipline and Craftsmanship

Rule 0: simplicity. It's easy to make things complicated, harder to make them simple. Don't accept complexity.

Rule 1: adopt a consistent style and use it everywhere. Decide on file organization, procedure structuring, naming conventions, location of curly braces, everything. By doing things consistently, they don't get omitted.

Discipline and Craftsmanship

Rule 0: simplicity. It's easy to make things complicated, harder to make them simple. Don't accept complexity.

Rule 1: adopt a consistent style and use it everywhere. Decide on file organization, procedure structuring, naming conventions, location of curly braces, everything. By doing things consistently, they don't get omitted.

Rule 2: don't litter. Temptation is to make fast fixes that dirty things up. It's crucial to take the time to fix things right at the beginning. You'll never have time to come back to it later!

Discipline and Craftsmanship

Rule 3: document carefully as you go. Don’t put this off! Most important things are interfaces: procedure headers and data structures and other things that tie together the parts of the system.

Discipline and Craftsmanship

Rule 3: document carefully as you go. Don’t put this off! Most important things are interfaces: procedure headers and data structures and other things that tie together the parts of the system.

Rule 4: documentation should describe things at a higher level than code.

Discipline and Craftsmanship

Rule 5: Put documentation near the code: otherwise you forget to change the documentation when you change the code.

Discipline and Craftsmanship

Rule 5: Put documentation near the code: otherwise you forget to change the documentation when you change the code.

Rule 6: Quality, not quantity.

Discipline and Craftsmanship

Rule 5: Put documentation near the code: otherwise you forget to change the documentation when you change the code.

Rule 6: Quality, not quantity.

Summary: discipline, craft. Take time up front to save time later.

Suggestions

- Attend lectures and tutorials (skipping lectures can be costly, instead of saving you time)
- Review lectures regularly, at least once a week (easier when it is still fresh, for the same reason, deferred exam is harder for you)
- Work independently (helps you understand the details)
- Get help whenever it is needed, office hours, tutorials, appointments (more efficient)

Maximizing your time!

Let’s work together.

Let’s work together.

Good luck!