CSE 30341: Operating Systems

- Instructor: Surendar Chandra (surendar@nd.edu)
  Room: 381 Fitz (631-8975)
  Office Hours: Tues 1:00-3:00, Wed 2:00-3:00
  (other times, by email appt)
  Email/iChat/AIM is the best way to reach me
  I am usually on AIM (surendar),
  Yahoo(surendarcsdukeedu), Skype (c.surendar)
- TAs: Nathan Regola and Santanu Chatterjee
- Course Web: cse.nd.edu/courses/cse30341/www
- Mailing list: cse30341-01-sp08@listserv.nd.edu

Outline for today

- High level introduction to Operating System principles
- Course policies:
  - Course goals, organization and expectation
  - Grading policy, late policy, reevaluation policy
  - Academic honesty

Computer System Structure

- Computer system can be divided into
  - Hardware – provides basic computing resources
    - Processing (CPU, graphics controller), storage(disk,memory), I/O devices (keyboard, mouse, CD/Tape, printer)
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
- Users (people, machines) want to run applications on hardware
- Operating system controls and coordinates use of hardware among various applications and users

Awful C program to show role of OS

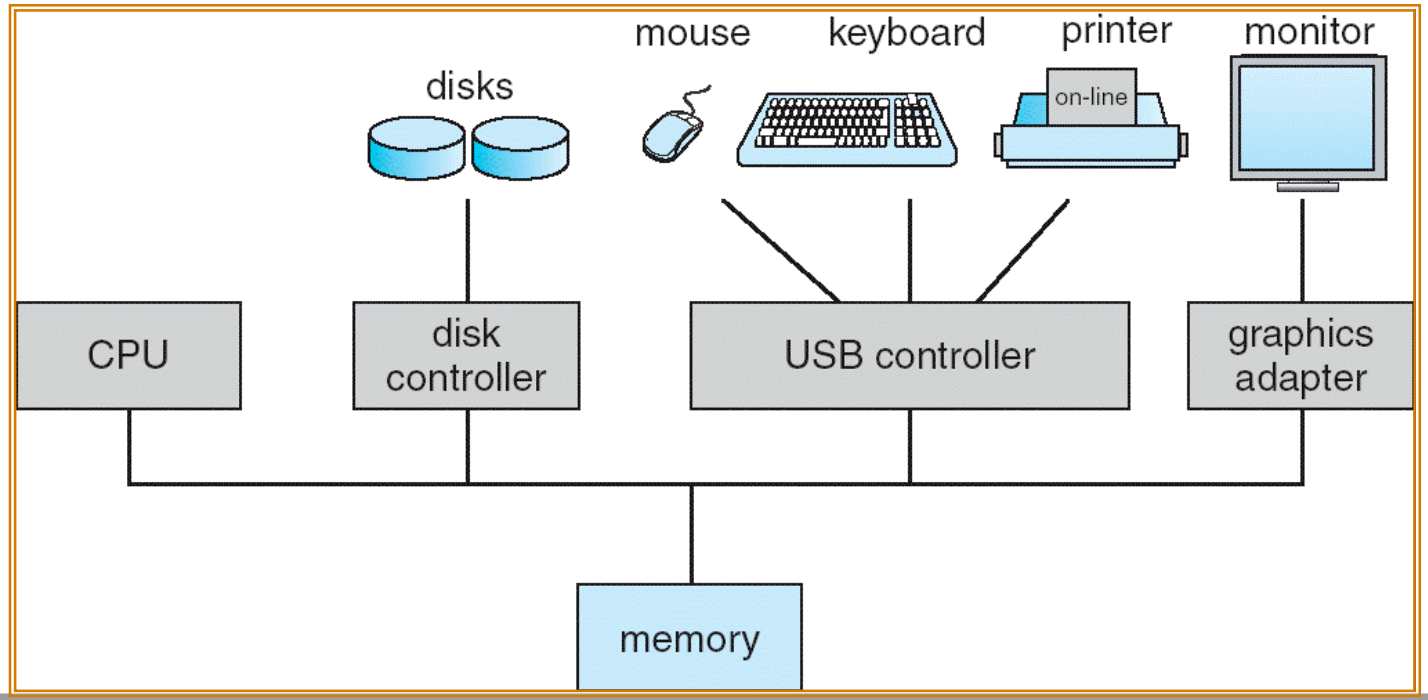
```
1)  int main(int argc, char *argv[], char *envp[]) {
2)    char buf[100];
3)    char *ptr = malloc(100);
4)    printf("Hello world\n");
5)    int fd = open("/dir1/file", O_WRONLY, 0666);
6)    *(ptr + 1000) = '\0';
7)    write(fd, ptr, 100);
8)    fsync(fd);
9)    close(fd);
10)   exit(0);
11)}
```

Computer Hardware

- Components:
  - Processing - (CPU) performs all computations
  - Input/Output - Mouse, Keyboard, printer
  - Storage - Memory, Hard disk, CD
- Technology limitations and cost leads to hierarchy of components
  - Storage
    - Hard disks are cheap, provide persistence and are slow
    - Main memory is expensive, not persistent but faster
    - Cache/Registers very expensive, not persistent but fastest
    - CD is persistent, very slow, cheap and removable
  - Processing
    - CPU is expensive and general purpose
    - Dedicated controllers (graphics processors, SCSI controllers)
- Limited resources
  - Need to share components

Computer System Organization

- Computer-system operation: concurrency for utilization and performance
  - One or more CPUs and device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices while competing for memory cycles



What is an Operating System?

- Acts as an intermediary between a user of a computer and the computer hardware
  - Challenge is to manage resources for competing uses: simultaneously playing interactive games and sending a print out to an laser printer
- OS is a **resource allocator**
  - Manages all resources (CPU, memory, disks etc.)
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

OS as a resource allocator:What if resources were infinite?

- As part of managing resources, OS hides system details from applications
  - Uses buffers to move data around for good performance. Even with unlimited resources, applications need this behavior
  - Applications can sometimes help. For example, threads are a way to tell the Operating System to assign different code segments to different processors. Similarly, posix sync() call to flush buffers to disk.
- Sometimes, multiplexing is not bad. Infinite monitors, keyboards, printers are not viable

Computer-System I/O Operation

- I/O devices and the CPU can execute concurrently
- Offload work to the device controller
  - Each device controller is in charge of a particular device type
  - Each device controller has a local buffer to store pending data
  - CPU moves data from/to main memory to/from local buffers
  - I/O is from the device to local buffer of controller
  - Device controller informs CPU that it has finished its operation by causing an *interrupt*

Operating System Structure

- Multiprogramming needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via job scheduling
  - When it has to wait (e.g, for I/O), OS switches to another job
  - Need to be careful for IO. For example, printer cannot be directly shared between two jobs

Operating System Structure

- Timesharing (multitasking) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing
  - Response time should be small (< 1 second)
  - Each user has programs executing in memory ⇨process abstraction
  - If several jobs ready to run at the same time ⇨ process management
  - If processes don't fit in memory, swapping moves them in and out to run ⇨ memory and storage management
    - Virtual memory allows execution of processes not completely in memory

Protection structure

- OS should protect the users/processes from each other as well as protect itself from users
- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user

Popular operating Systems

- Desktop and server OS: Microsoft Windows, UNIX and variants (Linux, Solaris, FreeBSD, Mac OSX)
  - Main focus of this course. We will use Linux for source code examples
  - Resource utilization is important (even though servers care about different things than desktops, e.g. throughput vs interactive computing)
- Embedded and realtime systems: runs your cars, TV, washing machines, nuclear plants etc. (e.g. QNX, Vxworks)
  - Many real time applications require timing guarantees
  - There are more embedded systems than desktops - not the focus of this course

Important “facts”

- Operating Systems cannot make hardware go faster. However, OS can make h/w appear faster
  - Caching previously read contents can make subsequent reads “appear” faster
- Tradeoff depend on specific operating scenario
  - Consider scheduling traffic through a four way intersection without the use of traffic lights
    - Distributed solution - stop signs: each driver makes independent decisions, on tie right (left) driver wins
    - Distributed solution - stop signs with more lanes to increase throughput along a certain artery
    - Centralized solution - traffic cop
    - Centralized solution - mechanical device that allows traffic from each side in a round robin fashion
  - In exams, write your solution and your assumptions

Why you should care about OS

- If you are interested in Architecture: arch and OS complement each other. Without OS, cannot use new arch. features. Without h/w support, OS cannot do its magic
- If you are interested in computing:
  - Question: Are these two code segments equivalent?

1)
```
for (int j=0; j<1024; j++)
        for (int i=0; i<1024; i++)
                array[i][j] = 1;
```

2)
```
for (int i=0; i<1024; i++)
        for (int j=0; j<1024; j++)
                array[i][j] = 1;
```

- Your friend asks: should I buy a laptop with 3.2 GHz processor or a 2.4 dual core processor? Should I buy a 500 GB 7200 rpm drive or 120 GB 15000 rpm drive?

Course Goals

- Cover core technologies in depth & introduce current operating system technologies
  - Need to read course topics before class. Quizzes to verify
- Goal is to cover as much breadth rather than depth
  - I want you to understand why we do something
- As much hands on experience as possible
- Home work projects should help

Course Organization

We will follow the course text for the most part

I will augment the discussions with topical research topics

I encourage open discussion about the technologies

Grade distribution

- Modules:
  - Process Management
  - Process Synchronization
  - Memory management
  - Storage management (CSE 70481)
  - Protection, Security (CSE 40567), Distributed (CSE 40771) and Real time (CSE 40463)
- Exams - 50%
  - 5 module exams (5 * 6%)
  - 5 Quizzes (5 * 2%)
  - Final (10%)
- Take home assignments – 25%
  - 5 modules (5 * 5%)
- Programming projects - 25%
  - 5 projects (5 * 5%)

Module assignments, exams and Final

- Exams:
  - Open book, open notes, in class exams
    - Module exams: 30 minutes
- Module take home assignments
  - assigned at the beginning of each module
  - designed to help you prepare for the final exams and module exams
  - You may have to perform experiments to answer some of the questions

Homework projects

- **Projects are group (ideally two) efforts**.
- Each project should be electronically turned in with a succinct report on your implementation strategy and what you learned.
- Projects should compile without any modifications. C is the preferred language. Use the Linux cluster in Cushing 208 for the projects. If you need a specific OS, you should make arrangements beforehand
- I may randomly select submissions for an one-on-one oral interview

Reevaluation policy

- Arithmetic errors, missed grading will be reevaluated promptly
- I encourage you to discuss concerns with your solution with me
- I discourage re-evaluation of partial credits (partial credits are based on the complexity of your solution and the overall class performance):
  - Football penalty policy:

    If you think you deserve a better partial grade, write down the reason why you think that you deserve a better grade and how many extra points you think you deserve. If I agree, you could get up to this many extra points. If I disagree, you will lose this much points. You can increase your odds by performing experiments to prove your answer

Late policy

- None – Projects/homework are due at 10:40 am (right before the beginning of class). **I do not accept late submissions** (not even a second)

- Please contact me regarding unforeseen emergencies

Academic Honesty

- Freedom of information rule:
  - Collaboration is acceptable (even for individual efforts such as take home assignments as long as you follow the rules of this course)
  - To assure that all collaboration is on the level, **you must always write the name(s) of your collaborators on your assignment.** Failure to adequately acknowledge your contributors is at best a lapse of professional etiquette, and at worst it is plagiarism. Plagiarism is a form of cheating.

Academic Honesty – Gilligans Island Rule

- This rule says that you are free to meet with fellow students(s) and discuss assignments with them. Writing on a board or shared piece of paper is acceptable during the meeting; however, you may **not take any written (electronic or otherwise) record away from the meeting**. This applies when the assignment is supposed to be an individual effort. After the meeting, engage in half hour of mind-numbing activity (like watching an episode of Gilligan's Island), before starting to work on the assignment. This will assure that **you are able to reconstruct what you learned from the meeting, by yourself, using your own brain**.