CS466 Class Notes #1
Topics: Introduction to OS and Virtualization
Reference: OSTEP Ch. 1-2

What is an Operating System?

1. **Main purpose:**
   - Scheduler
   - Abstraction
   - Manager
   - Virtualizer

2. What is virtualization?
   - Taking physical resource and sharing it (creating a virtual version)

3. How do users interact with the OS?
   - API
   - Open a file
   - Close
   - Read

4. An OS is also a

History of OS

Early:
- Library

Middle:
- Time sharing
- Mini computers

Modern:
- Personal computers
The OS is responsible for making it easy to run programs (even allowing you to seemingly run many at the same time), allowing programs to share memory, enabling programs to interact with devices, and other fun stuff like that.

Stuck in an infinite loop:
The OS must run non-stop; when it fails, all applications running on the system fail as well.

Abstraction layers for onion

layer | use
------|------------------
hardware | actual computing engine
IO drivers | basic IO support
OS kernel | process control
OS Functions | high level functions (application software callable)
shell | user's interface to the (real) OS (The point of assignment 1: OS != shell)
application | programs THEY write

1) abstraction example: disk HW: T S H, basic IO: virtual sectors, OS: files, shells: cwd

2) OS lets us share disk space, cpu time, ...

3) early system: card-reader, CPU, printer (expensive CPU idle much of the time)

4) buffers (buffer manager = early OS): card-readers' output to tape, tape to CPU, ...

5) multi tasking:
   a. fixed memory partitioning
   b. variable memory partitioning
   c. paging - more later

6) interrupts: clock (time to switch), IO done, key hit, ...

7) Mention OS structure: old monolithic, new as collection of processes.
Design Goals in Operating Systems

1. Use abstractions to make the system convenient and easy to use.
2. Provide high performance (minimize overheads)
3. Provide protection between applications, and between the OS and applications
4. Provide a high degree of OS reliability
5. Be energy efficient
6. Have security against malicious applications
7. Have mobility to work on smaller devices

Myth of Multi-Tasking: https://www.youtube.com/watch?v=xO_oEGHWSMU

Abstraction Layers
What does the above code output if we run it as "./thread 20"?

What about with "./thread 1000000"?

Wrong answer

Why?

no locks
need for atomic operations
Deadlock 😰

Persistence

disk

Finally, two notes from your predecessors

1) I learned about the powers of procrastination.

2) I think my classmates and I are struggling with the length of these assignments because it's relatively new to us. In every other project throughout the curriculum, it's been easy to imagine the first step and get a sense of the overall process. (In part, this is because, in other classes, the class IS the project--when you sit down in CS302, you learn about the structure of a binary tree. I feel like the lecture portions and assignments are more disjointed than that in OS.) I think it's necessary to go beyond that and attempt what the OS course has--that is to say, it's necessary to push students to design structures that, initially, they don't understand at all.
Three easy pieces: Virtualization, Concurrency, and Persistence

V1. Virtualizing the CPU: How do we run, for instance, 4 programs on one CPU?

V2. Virtualizing Memory: Can different programs access each other’s memory?

Concurrency:

```
#include <stdio.h>
#include <stdlib.h>
#include "common.h"

volatile int counter = 0;
int loops;

void *worker(void *arg)
{
   int i;
   for (i = 0; i < loops; i++)
   {
      counter++;
   }
   pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
   if (argc != 2)
   {
      fprintf(stderr, "usage: threads <loops>\n");
      exit(1);
   }

   loops = atoi(argv[1]);
   pthread_t p1, p2;
   printf("Initial value : %d\n", counter);
   pthread_create(&p1, NULL, worker, NULL);
   pthread_create(&p2, NULL, worker, NULL);
   pthread_join(p1, NULL);
   pthread_join(p2, NULL);
   printf("Final value : %d\n", counter);
   return 0;
}
```