Todays Agenda [1/15]

- Course Introduction - syllabus
- Lab Project 1
- Examine the 3 main pieces of an OS
    - Virtualization
    - Concurrency
    - Persistence
- Start a 20,000 foot discussion of an OS and it’s components
    - Overview of the tasks of a kernel
    - Introductory discussion of a Process and how it relates to the various OS components
- Introduce Events and Interrupts

Reading Roadmap [Operating Systems; Three Easy Pieces]

Read now (and again over the next 2 weeks):

- Chapter 1 – Introduction (& Dialogue)
- Virtualization Chapters 3 & 4 – Processes

In class, for the next several lectures, I will cover the contents of these chapters as well as a broad overview of topics we will cover in detail as the course continues.

We will cover content of chapters 1 - 40 (a few chapters will be skipped)
I will tell you when we are moving to a new topic – it will also be recognizable by the lecture and slide content.

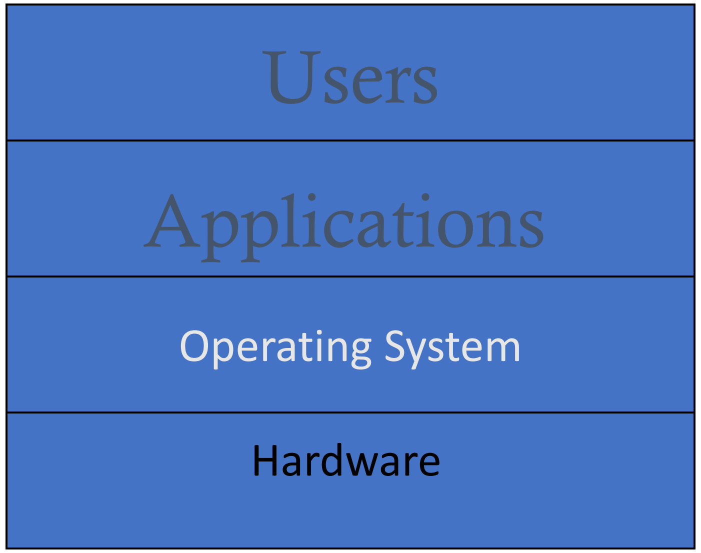
You should read the entire chapter as we start a topic (for example Processes or Threads or Memory Management). Then re-read as we move through the chapter.
Try the questions at the end of each chapter. – Don’t just answer in a brief way; use the question to prompt you to review the concepts introduced in the question.

- Additional Reading:
  - Stallings: Operating Systems Internals – PDF available in Canvas
    - I take material from this as well.
  - Gray: Interprocess Communication in Linux – PDF available in Canvas
    - Excellent programming reference
  - Kerrisk: The Linux Programming Interface – PDF available in Canvas
    - Excellent programming reference
- Class Participation
  - Attend all lectures
    - Interact and ask questions
  - For each lecture I will assign 8-10 people as ‘class assistants’
    - Responsible for material from previous lecture and upcoming lecture
    - Provide answers to questions I ask
    - Bring questions for me and the class.

- I will post my lecture slides and other information to Canvas.
  - I will also post a narrative from each topic/lecture: “Modules”
- Use the slides and Modules as
  - A review of critical ideas and concepts
  - Don’t just read them – use them as a guide to return to the book to fill in details
- If you are not proactive in keeping up, you will not pass this course
- The labs are a critical part of the course (25%); if you cannot program, you will most likely fail the course – even if you do well in exams.
  - Start YESTERDAY to practice/learn programming in C or C++

What is an Operating System?

Not easy to define precisely…



Operating System (OS):

Software that converts hardware into a useful form for applications

Some Introductory Perspective

- The relationship between programs, the CPU and other devices is critical to our effective use of a computer system
- Program – static instructions that we develop to control the CPU
- CPU: Fetch, Decide, Execute
  - Moves through instructions and memory driven by results of execution
- Computer system has devices other than the CPU: Memory, Disks, other I/O, network, etc.

- Early OS was really a “control program” – library of functions that a programmer could use to access/control non-CPU activities.
- Batch invocation – another s/w component of the control program for managing CPU
- What about other resources allocation?
- Recognition that device control is special and that users could interfere with the way devices worked – lead to ‘modes of operation’ of the CPU
- Opened up an entire area of change in how a computer and its control should be organized.

What DOES OS Provide?

Role #1: Abstraction - Provide standard library for resources

What is a resource?
- Anything valuable (e.g., CPU, memory, disk)

What abstraction does modern OS typically provide for each resource?
- CPU:
  - process and/or thread
- Memory:
  - address space
- Disk:
  - files

Advantages of OS providing abstraction?
- Allow applications to reuse common facilities
- Make different devices look the same
- Provide higher-level or more useful functionality

Challenges
- What are the correct abstractions?
- How much of hardware should be exposed?

What DOES OS PROVIDE?

Role #2: Resource management – Share resources well

Advantages of OS providing resource management?

- Protect applications from one another
- Provide efficient access to resources (cost, time, energy)
- Provide fair access to resources

Challenges

- What are the correct mechanisms?
- What are the correct policies?

OS Organization

- How to cover all the topics relevant to operating systems?

Three PIECES: FIRST

- **Virtualization**
  - **Make each application believe it has each resource to itself**

- **Demo**
  - **Virtualize CPU and memory**
    - **process**
    - **Address space**

CPU Virtualization

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

    while (1) {
        printf("%s\n", str);
        Spin(1);   //delay 1 second
    }
    return 0;
}
```

```
double
GetTime()
{
    struct timeval t;
    int rc = gettimeofday(&t, NULL);
    assert(rc == 0);
    return (double)t.tv_sec + (double)t.tv_usec/1e6;
}

void
Spin(int howlong)
{
    double t = GetTime();
    while ((GetTime() - t) < (double)howlong)
                ; // do nothing in loop
}
```

Execute multiple versions of the process simultaneously (concurrently) and see the interleaved output. In our time frame they appear to have their own CPU. But in reality the CPU is switched among the processes.

Memory Virtualization

```
int
main(int argc, char *argv[])
{
    if (argc != 2) {
            fprintf(stderr, "usage: mem <value>\n");
            exit(1);
    }
    int p;
    p = atoi(argv[1]);
    printf("(pid:%d) addr of p:   %llx\n", (int) getpid(),
            (unsigned long long) &p);
    while (1) {
            Spin(1);  //delay 1 sec
            p = p + 1;
            printf("(pid:%d) value of p: %d\n", getpid(), p);
    }

    return 0;
}
```

Execute multiple versions simultaneously and see the address of ‘p’ remain the same relative location in each process. Thus the ’virtual’ or process specific address space is the same although in different physical memories.