Introduction to Operating Systems

Core Problems Running programs execute instructions. This involves

```
- fetching the instructions from memory
- figuring out what the instruction is
- doing whatever the instruction specified This happens billions of times
  per second, on modern CPUs. This basic model of computation is called the
  _Von Neumann_ model. The goal of the operating system is to present an
  _easy-to-use abstraction_.
```

Virtualization This is done primarily through *virtualization* – physical

resources are abstracted into more understandable virtual forms. For example, this allows each program to run as through it's the only program, when there may actually be multiple processes being run and managed concurrently. We can also observe virtualization of memory: two programs can seem to allocate memory at the same address, when really the operating system maps these to different physical addresses.

Concurrency A second problem focus will be addressing the problems that

arise relating to *concurrency*. For example, if we have a process with threads that access shared memory in a loop that increments a value, because the steps are non-atomic (happening all at once) we can end up with unexpected results. This motivates operating system primitives to make writing concurrent programs easier.

Persistence A final theme that will be addressed is *persistence*. Being

able to store data in the face of system crashes or power loss is important to users. Persistence is implemented in both hardware and software, with the software implementation known as the *file system*. Unlike some of the virtualization-related abstractions, the file system mainly facilitates *sharing* of information between processes.

Design Goals

- Building easy-to-use abstractions
- Balance ease-of-use with performance

- Provide protection: malicious processes shouldn't impact others on the system
  - Provide isolation of processes
- Be reliable: if the operating system fails, everything stops
- Energy-efficiency, security, mobility, etc.