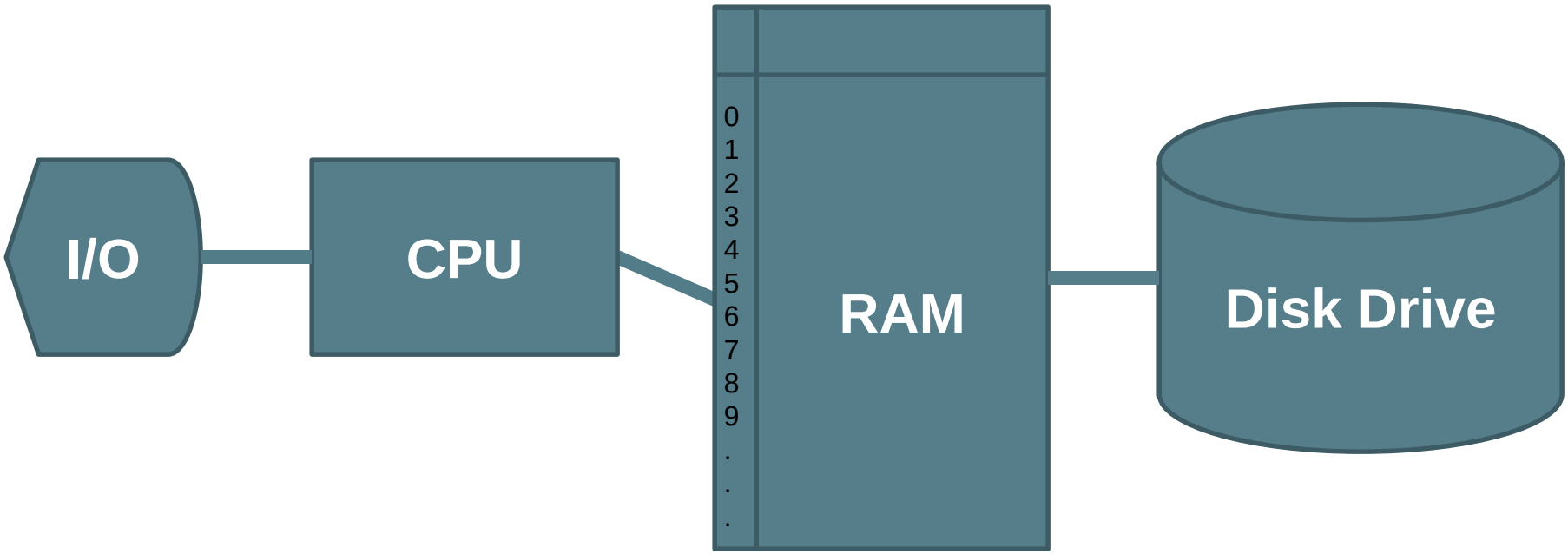
CS 497: Cybersecurity

Galin Zhelezov
Department of Engineering and Computer Science
York College of Pennsylvania

Operating Systems Concepts

A Computer Model

- **An operating system has to deal with the fact that a computer is made up of a CPU, random access memory (RAM), input/output (I/O) devices, and long-term storage.**



OS Concepts

- **An operating system (OS) provides the interface between the users of a computer and that computer's hardware.**
  - An operating system manages the ways applications access the resources in a computer, including its disk drives, CPU, main memory, input devices, output devices, and network interfaces.
  - An operating system manages multiple users.
  - An operating system manages multiple programs.

Multitasking

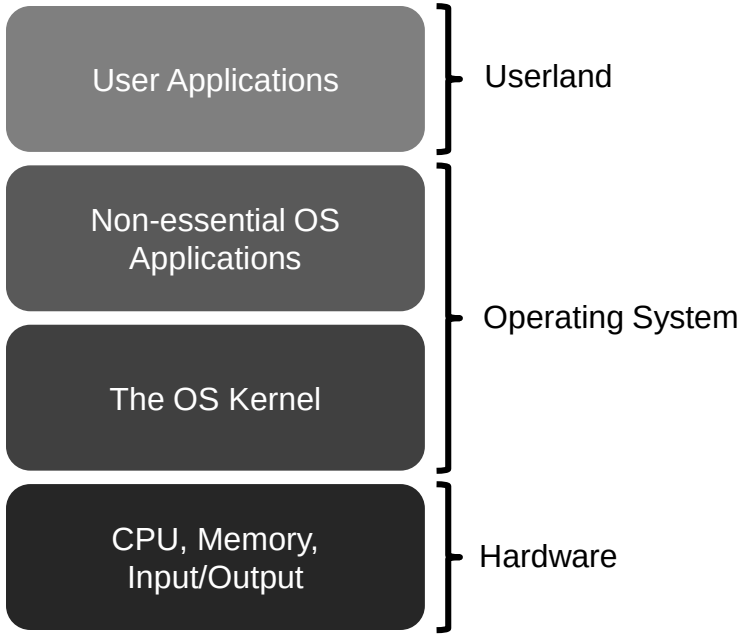
- **Give each running program a “slice” of the CPU’s time.**
- **The CPU is running so fast that to any user it appears that the computer is running all the programs simultaneously.**

Public domain image from http://commons.wikimedia.org/wiki/File:Chapters_meeting_2009_Liam_juggling.JPG

The Kernel

- **The kernel is the core component of the operating system. It handles the management of low-level hardware resources, including memory, processors, and input/output (I/O) devices, such as a keyboard, mouse, or video display.**

- **Most operating systems define the tasks associated with the kernel in terms of a layer metaphor, with the hardware components, such as the CPU, memory, and input/output devices being on the bottom, and users and applications being on the top.**



Input/Output

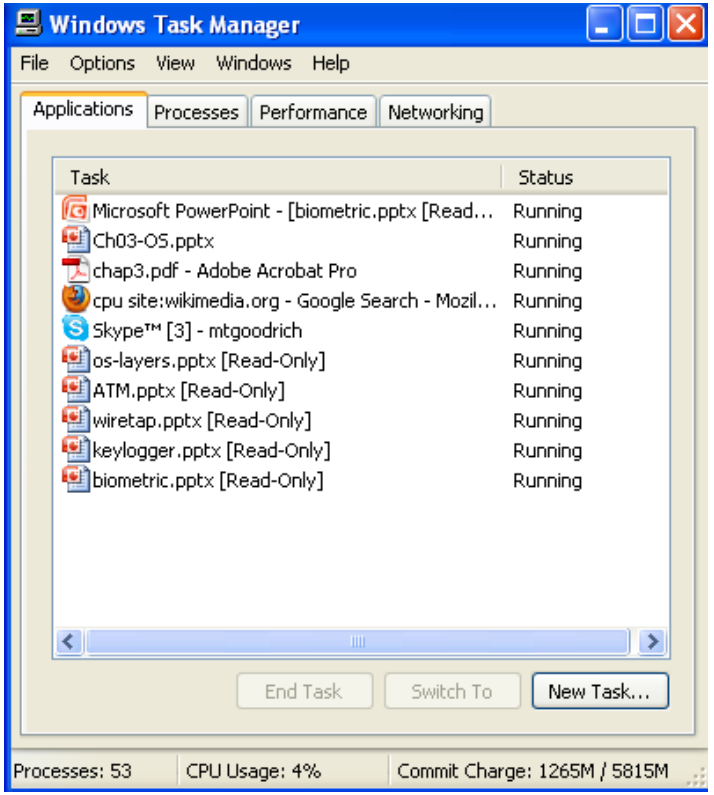
- **The input/output devices of a computer include things like its keyboard, mouse, video display, and network card, as well as other more optional devices, like a scanner, Wi-Fi interface, video camera, USB ports, etc.**
- **Each such device is represented in an operating system using a device driver, which encapsulates the details of how interaction with that device should be done.**
  - The **application programmer interface** (**API),** which the device drivers present to application programs, allows those programs to interact with those devices at a fairly high level, while the operating system does the “heavy lifting” of performing the low-level interactions that make such devices actually work.

System Calls

- **User applications don't communicate directly with low-level hardware components, and instead delegate such tasks to the kernel via system calls.**
- **System calls are usually contained in a collection of programs, that is, a library such as the C library (libc), and they provide an interface that allows applications to use a predefined series of APIs that define the functions for communicating with the kernel.**
  - Examples of system calls include those for performing file I/O (open, close, read, write) and running application programs (exec).

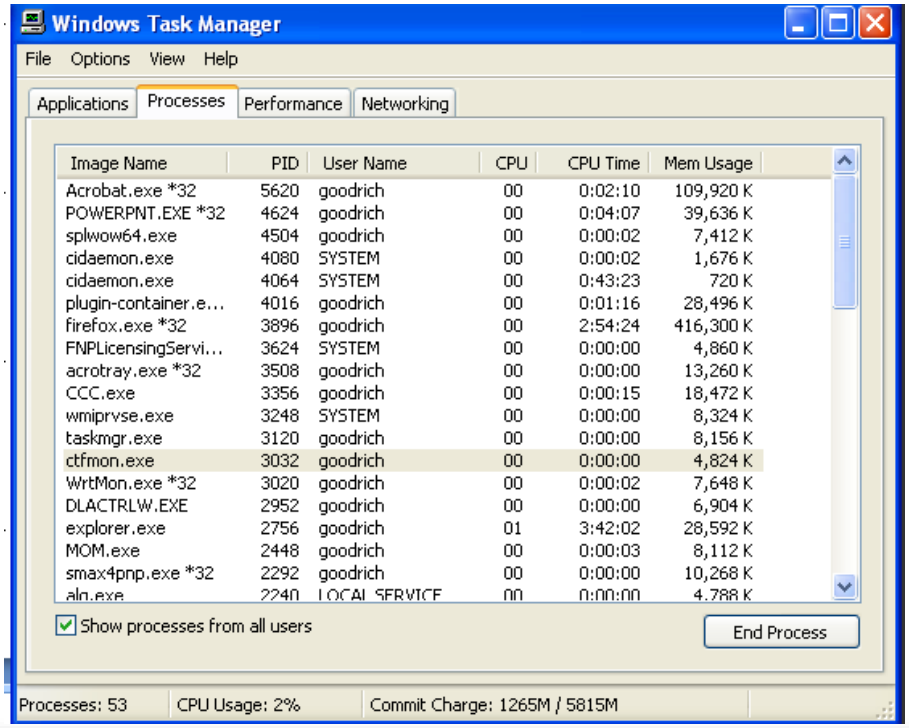
Processes

- **A process is an instance of a program that is currently executing.**
- **The actual contents of all programs are initially stored in persistent storage, such as a hard drive.**
- **In order to be executed, a program must be loaded into random-access memory (RAM) and uniquely identified as a process.**
- **In this way, multiple copies of the same program can be run as different processes.**
  - For example, we can have multiple copies of MS Powerpoint open at the same time.



Process IDs

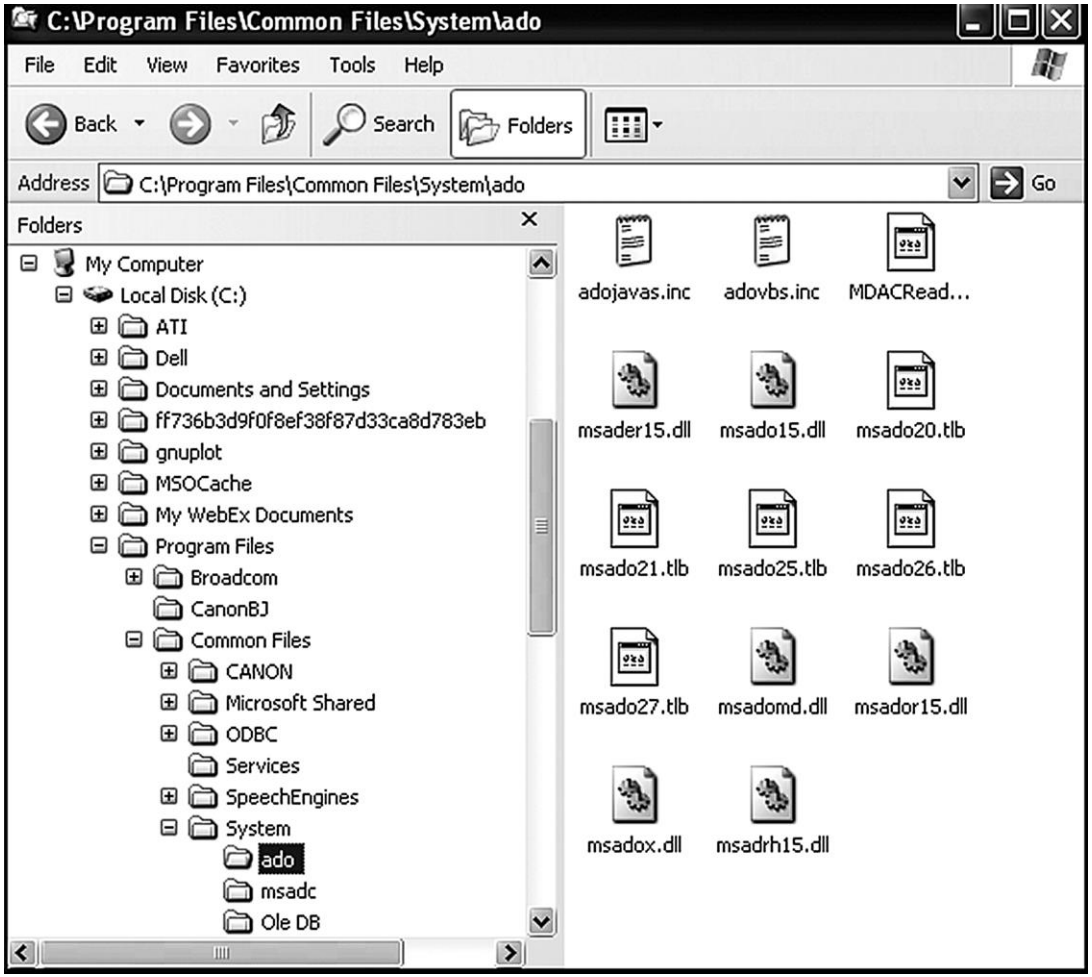
- **Each process running on a given computer is identified by a unique nonnegative integer, called the process ID (PID).**
- **Given the PID for a process, we can then associate its CPU time, memory usage, user ID (UID), program name, etc.**



File Systems

- **A filesystem is an abstraction of how the external, nonvolatile memory of the computer is organized.**
- **Operating systems typically organize files hierarchically into folders, also called directories.**
- **Each folder may contain files and/or subfolders.**
- **Thus, a volume, or drive, consists of a collection of nested folders that form a tree.**
- **The topmost folder is the root of this tree and is also called the root folder.**

File System Example



File Permissions

- **File permissions are checked by the operating system to determine if a file is readable, writable, or executable by a user or group of users.**

- **In Unix-like OS's, a file permission matrix shows who is allowed to do what to the file.**

  - Files have **owner permissions**, which show what the owner can do, and **group permissions**, which show what some group id can do, and **world permissions**, which give default access rights.

```
rodan:~/java % ls -l
total 24
-rwxrwxrwx   1 goodrich faculty     2496 Jul 27 08:43 Floats.class
-rw-r--r--   1 goodrich faculty     2723 Jul 12  2006 Floats.java
-rw-------   1 goodrich faculty      460 Feb 25  2007 Test.java
rodan:~/java %
```

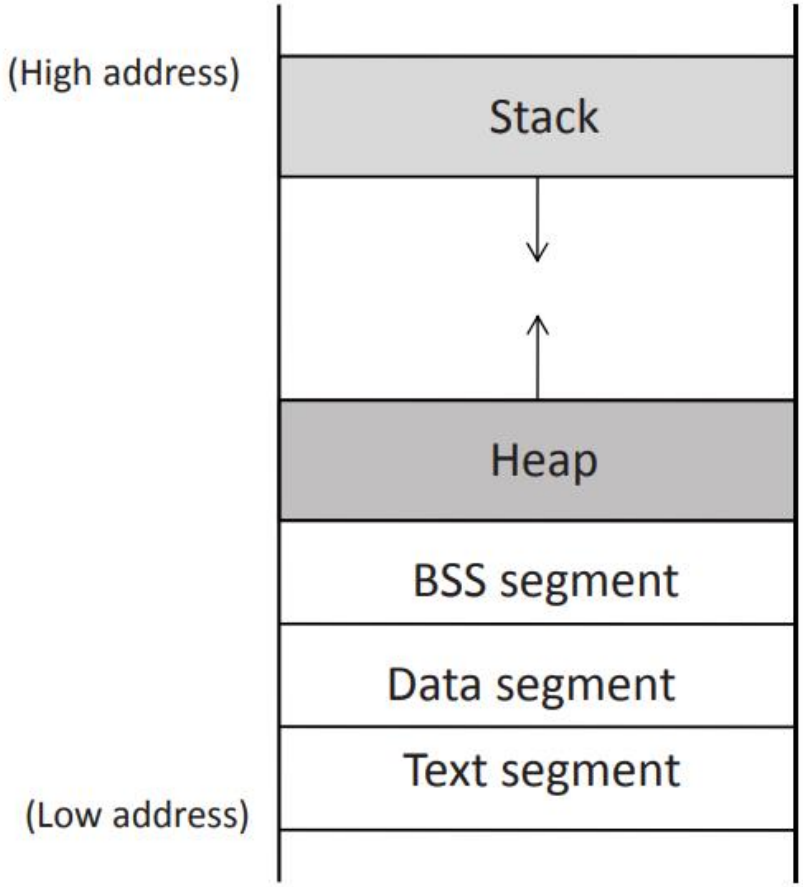
Memory Management

- **The RAM memory of a computer is its address space.**
- **It contains both the code for the running program, its input data, and its working memory.**
- **For any running process, it is organized into different segments, which keep the different parts of the address space separate.**
- **As we will discuss, security concerns require that we never mix up these different segments.**

Memory Organization

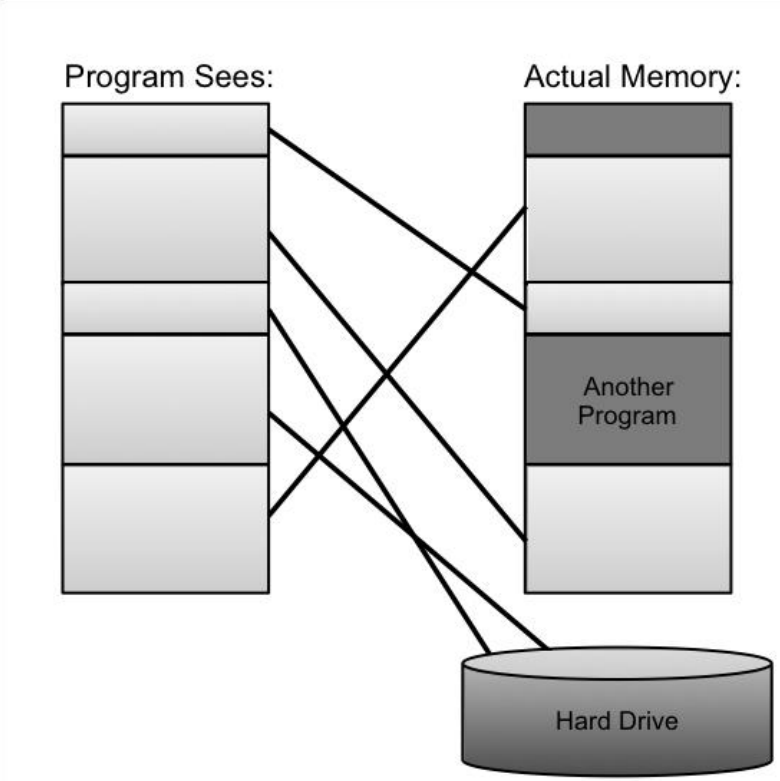
- **Text. This segment contains the actual (binary) machine code of the program.**
- **Data. This segment contains static program variables that have been initialized in the program code.**
- **BSS. This segment, which is named for an antiquated acronym for block started by symbol, contains static variables that are uninitialized.**
- **Heap. This segment, which is also known as the dynamic segment, stores data generated during the execution of a process.**
- **Stack. This segment houses a stack data structure that grows downwards and is used for keeping track of the call structure of subroutines (e.g., methods in Java and functions in C) and their arguments.**

Memory Layout



Virtual Memory

- **There is generally not enough computer memory for the address spaces of all running processes.**
- **Nevertheless, the OS gives each running process the illusion that it has access to its complete (contiguous) address space.**
- **In reality, this view is virtual, in that the OS supports this view, but it is not really how the memory is organized.**
- **Instead, memory is divided into pages, and the OS keeps track of which ones are in memory and which ones are stored out to disk.**



Page Faults

1. Process requests virtual address not in memory, causing a page fault.

Process

"read 0110101"

"Page fault, let me fix that."

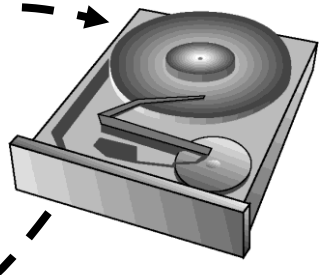
Paging supervisor

2. Paging supervisor pages out an old block of RAM memory.

Blocks in RAM memory:

old

new

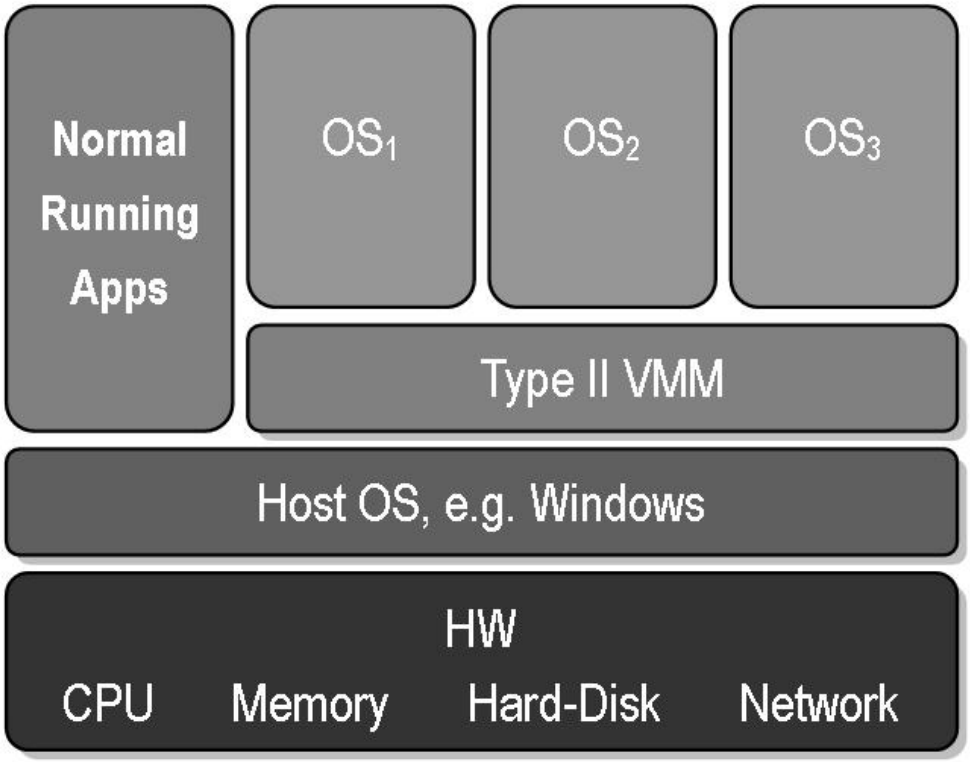


External disk

3. Paging supervisor locates requested block on the disk and brings it into RAM memory.

Virtual Machines

- **Virtual machine: A view that an OS presents that a process is running on a specific architecture and OS, when really it is something else. E.g., a windows emulator on a Mac.**
- **Benefits:**
  - **Hardware Efficiency**
  - **Portability**
  - **Security**
  - **Management**



Public domain image from http://commons.wikimedia.org/wiki/File:VMM-Type2.JPG