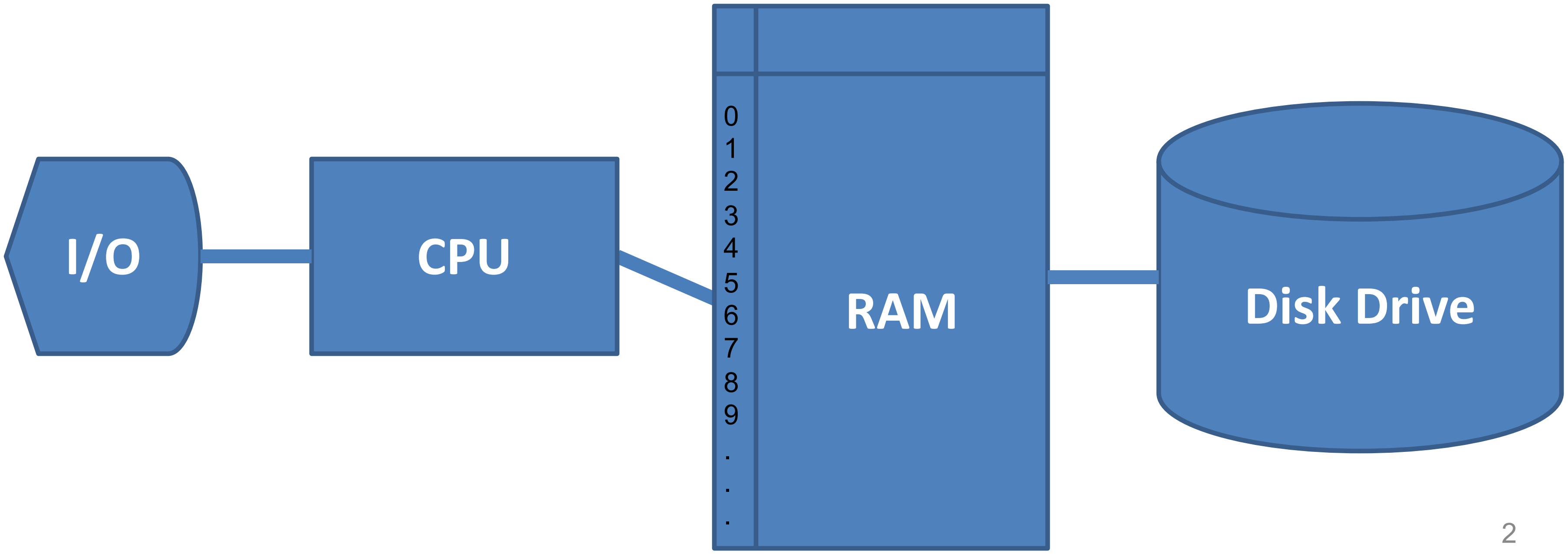
Chapter 3
Operating Systems Concepts

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 <u>manages the ways applications access the resources</u> in a computer, including its disk drives, CPU, main memory, input devices, output devices, and network interfaces.
  - An operating system <u>manages multiple users</u>.
  - An operating system <u>manages multiple programs</u>.

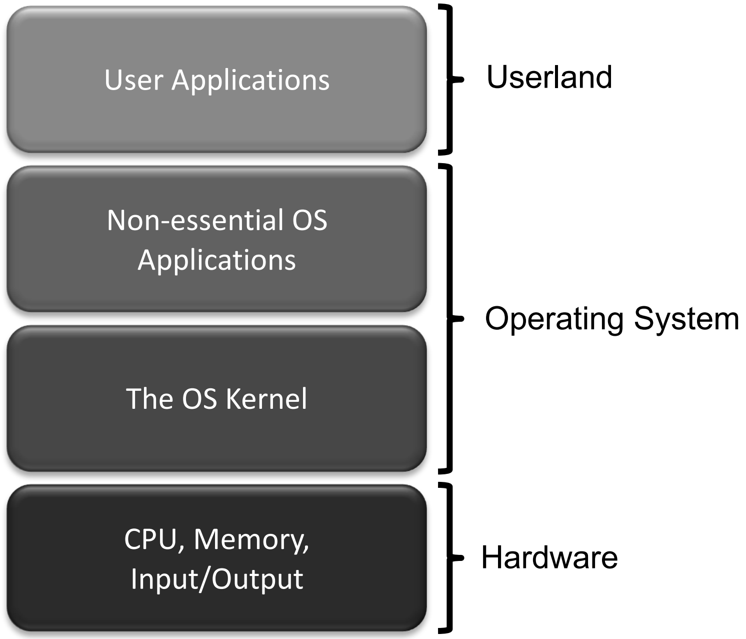
Multitasking

- 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

3.1.1 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

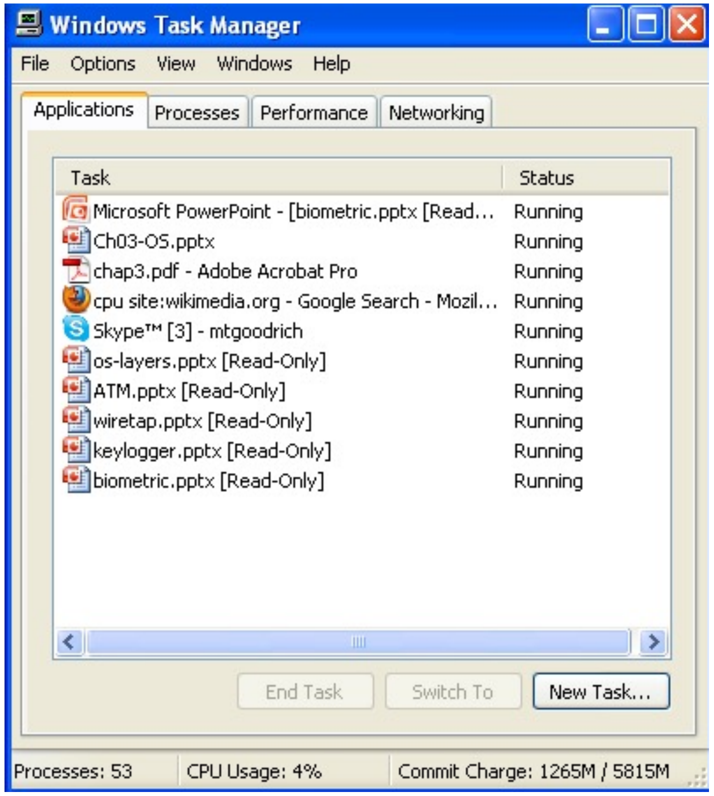
- The **input/output devices** of a computer include things like its keyboard, mouse, video display, and network card, as well as other more optional devices, like a scanner, Wi-Fi interface, video camera, USB ports, etc.
- Each such device is represented in an operating system using a **device driver,** 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).

3.1.2 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.



Task	Status
Microsoft PowerPoint - [biometric.pptx [Read...	Running
Ch03-OS.pptx	Running
chap3.pdf - Adobe Acrobat Pro	Running
cpu site:wikimedia.org - Google Search - Mozil...	Running
Skype™ [3] - mtgoodrich	Running
os-layers.pptx [Read-Only]	Running
ATM.pptx [Read-Only]	Running
wiretap.pptx [Read-Only]	Running
keylogger.pptx [Read-Only]	Running
biometric.pptx [Read-Only]	Running

Processes: 53 CPU Usage: 4% Commit Charge: 1265M / 5815M

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.

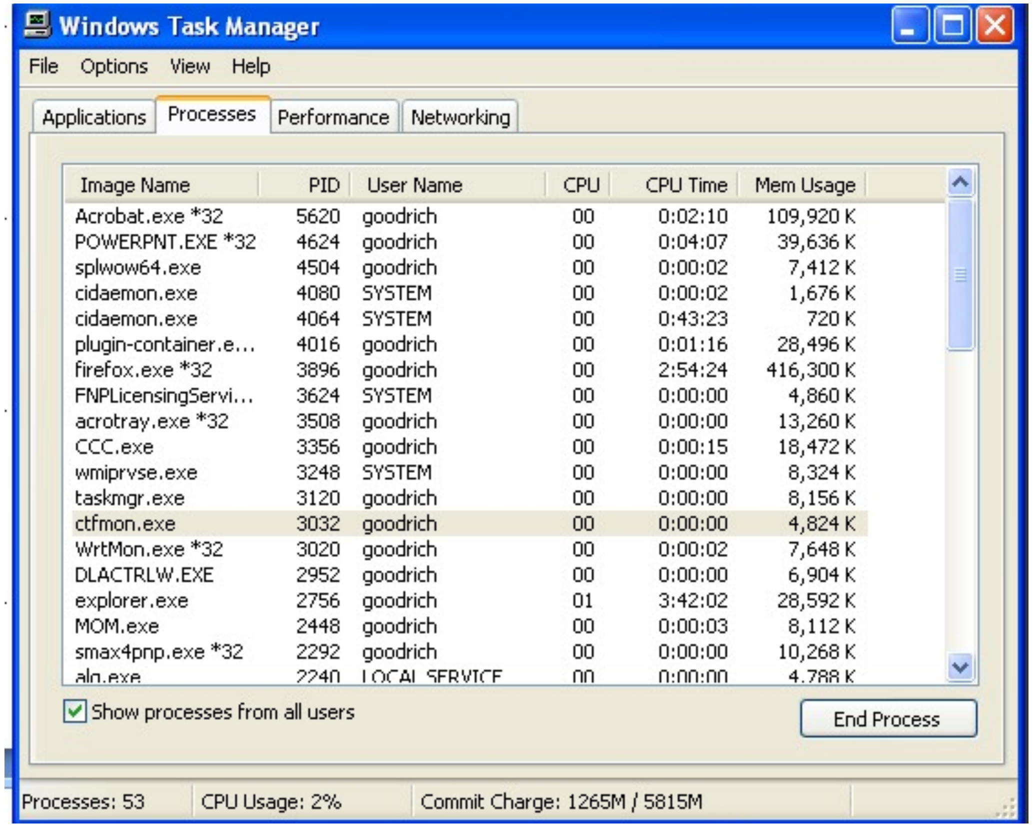


Image Name	PID	User Name	CPU	CPU Time	Mem Usage
Acrobat.exe *32	5620	goodrich	00	0:02:10	109,920 K
POWERPNT.EXE *32	4624	goodrich	00	0:04:07	39,636 K
splwow64.exe	4504	goodrich	00	0:00:02	7,412 K
cidaemon.exe	4080	SYSTEM	00	0:00:02	1,676 K
cidaemon.exe	4064	SYSTEM	00	0:43:23	720 K
plugin-container.e...	4016	goodrich	00	0:01:16	28,496 K
firefox.exe *32	3896	goodrich	00	2:54:24	416,300 K
FNPLicensingServi...	3624	SYSTEM	00	0:00:00	4,860 K
acrotray.exe *32	3508	goodrich	00	0:00:00	13,260 K
CCC.exe	3356	goodrich	00	0:00:15	18,472 K
wmiprvse.exe	3248	SYSTEM	00	0:00:00	8,324 K
taskmgr.exe	3120	goodrich	00	0:00:00	8,156 K
ctfmon.exe	3032	goodrich	00	0:00:00	4,824 K
WrtMon.exe *32	3020	goodrich	00	0:00:02	7,648 K
DLACTRLW.EXE	2952	goodrich	00	0:00:00	6,904 K
explorer.exe	2756	goodrich	01	3:42:02	28,592 K
MOM.exe	2448	goodrich	00	0:00:03	8,112 K
smax4pnp.exe *32	2292	goodrich	00	0:00:00	10,268 K
alg.exe	2240	LOCAL SERVICE	00	0:00:00	4,788 K

[✓] Show processes from all users

Processes: 53 CPU Usage: 2% Commit Charge: 1265M / 5815M

Users and the Process Tree

- If a user creates a new process by making a request to sun some program,
- The kernel sees this as an existing process asking to create a new process.
- The process is called forking.
- The existing process is the parent process and the one being forked is known as the child process.

Inter-process communication

1. Pass messages by reading and writing files
2. Share the same region of physical memory
3. Pipes and sockets: the sending and receiving processes to share the pipe or socket as an in-memory object.

Inter-process communication

- Signals: in Unix-based system, processes can send direct messages to each other asynchronously.
- Remote Procedure Calls: Windows rely on remote procedure calls (RPC) which allows a process to call a subroutine from another process's program.

Daemons and Services

- Computers run dozens of processes that run without any user intervention.
- In linux, these background processes are know as daemons.
- They are indistinguishable from any other process, are started by init process and operate at varying levels of permissions.
- They are forked before the user is authenticated, and able to run with higher permissions than any user, and survive the end of the login sessions.
- Examples are processes that control web servers, remote logins, and print servers.

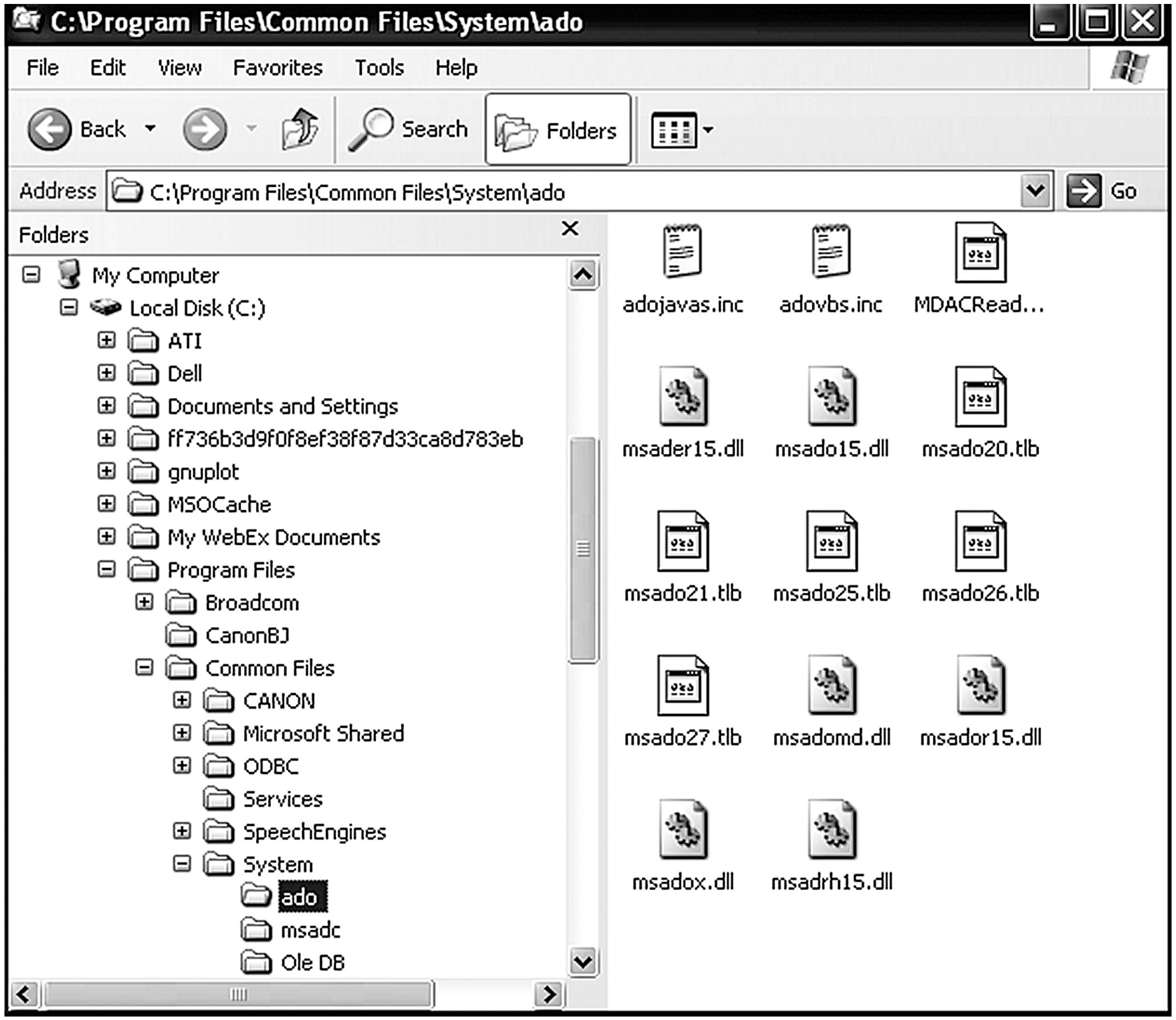
Daemons and Services

- Windows have an equivalent class of processes known as services.
- They are easily distinguishable from other processes,
- are differentiated in monitoring software such as Task Manager.

3.1.3 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

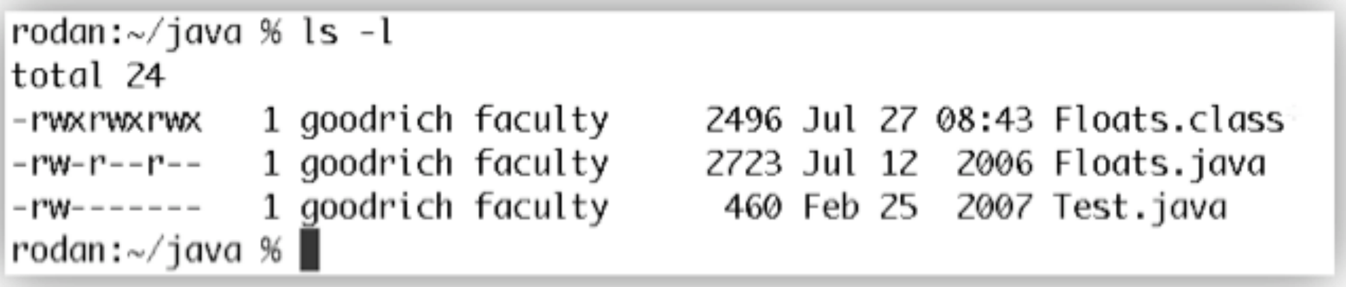

C:\Program Files\Common Files\System\ado
File
Edit
View
Favorites
Tools
Help
Back
Search
Folders
Address
C:\Program Files\Common Files\System\ado
Go
Folders

- My Computer
  - Local Disk (C:)
    - ATI
    - Dell
    - Documents and Settings
    - ff736b3d9f0f8ef38f87d33ca8d783eb
    - gnuplot
    - MSOCache
    - My WebEx Documents
    - Program Files
      - Broadcom
      - CanonBJ
      - Common Files
        - CANON
        - Microsoft Shared
        - ODBC
        - Services
        - SpeechEngines
        - System
          - ado
          - msadc
          - Ole DB

adojavas.inc, adovbs.inc, MDACRead..., msader15.dll, msado15.dll, msado20.tlb, msado21.tlb, msado25.tlb, msado26.tlb, msado27.tlb, msadomd.dll, msador15.dll, msadox.dll, msadrh15.dll

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 % █
```

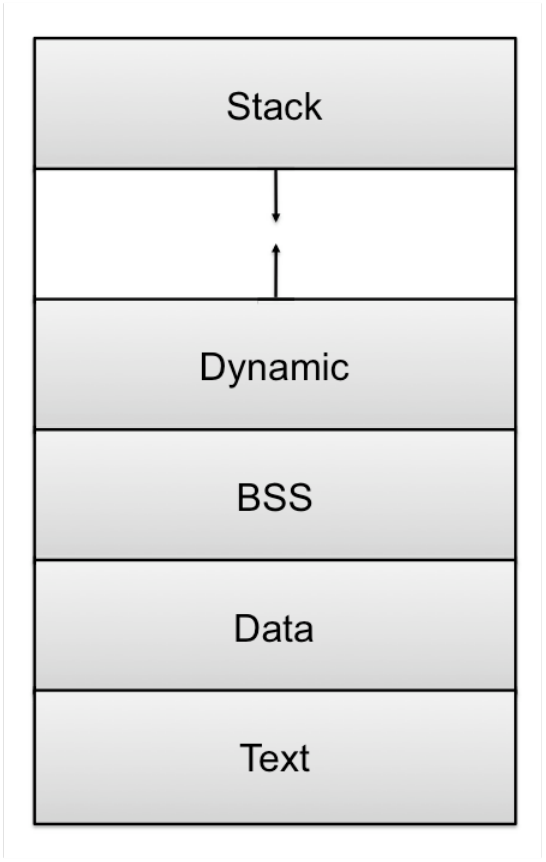
3.1.4 Memory Management

- 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

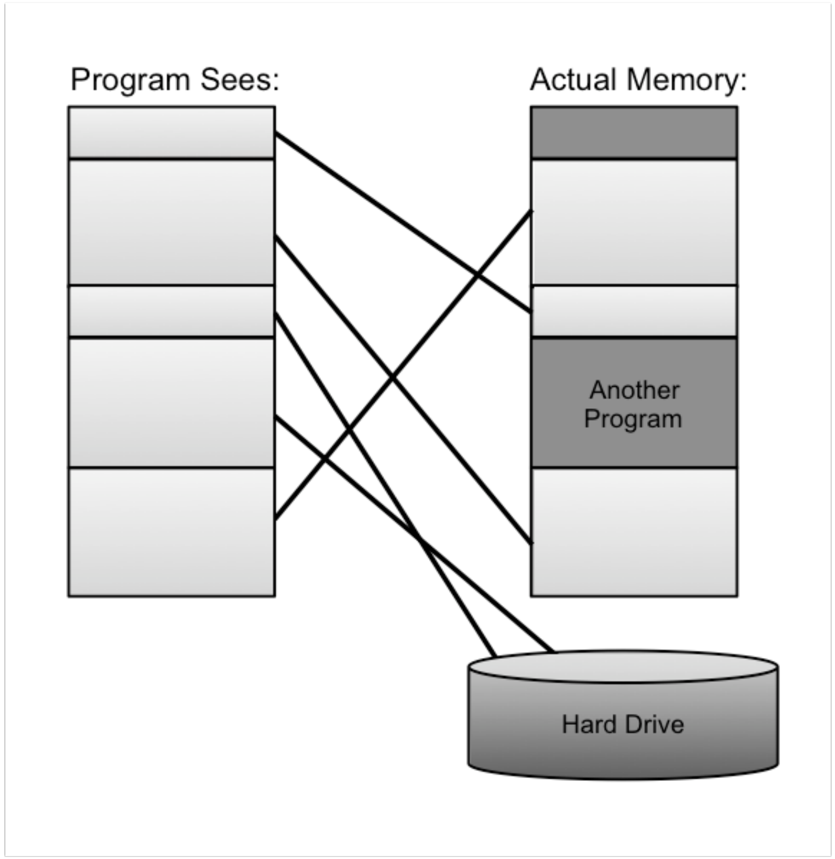
- **Text.** This segment contains the actual (binary) machine code of the program.
- **Data.** This segment contains static program variables that have been initialized in the program code.
- **BSS.** This segment, which is named for an antiquated acronym for block started by symbol, contains static 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

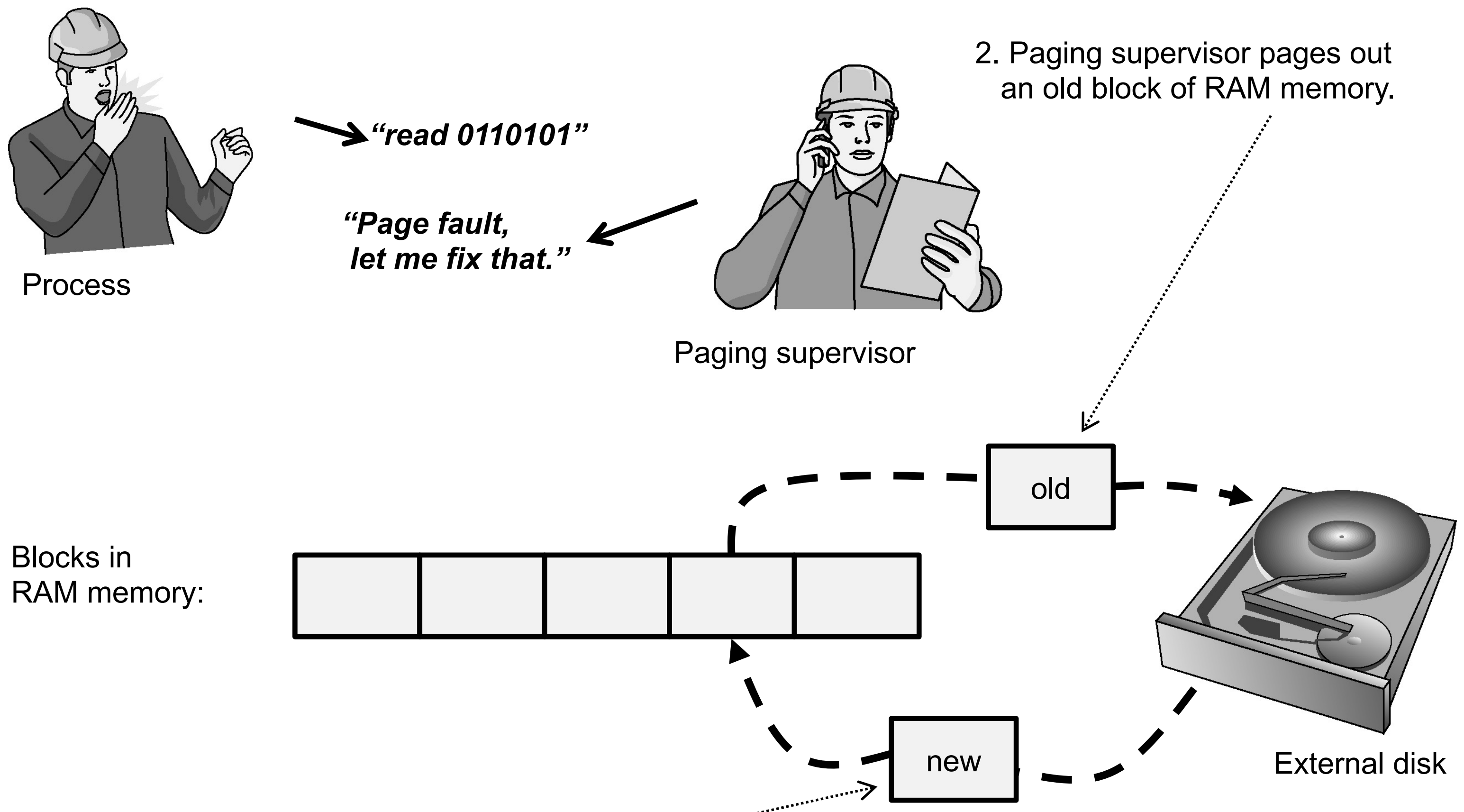
- There is generally not enough computer memory for the address spaces of all running processes.
- Nevertheless, the OS gives each running process the illusion that it has access to its complete (contiguous) address space.
- In reality, this view is **virtual**, in that the OS supports this view, but 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.

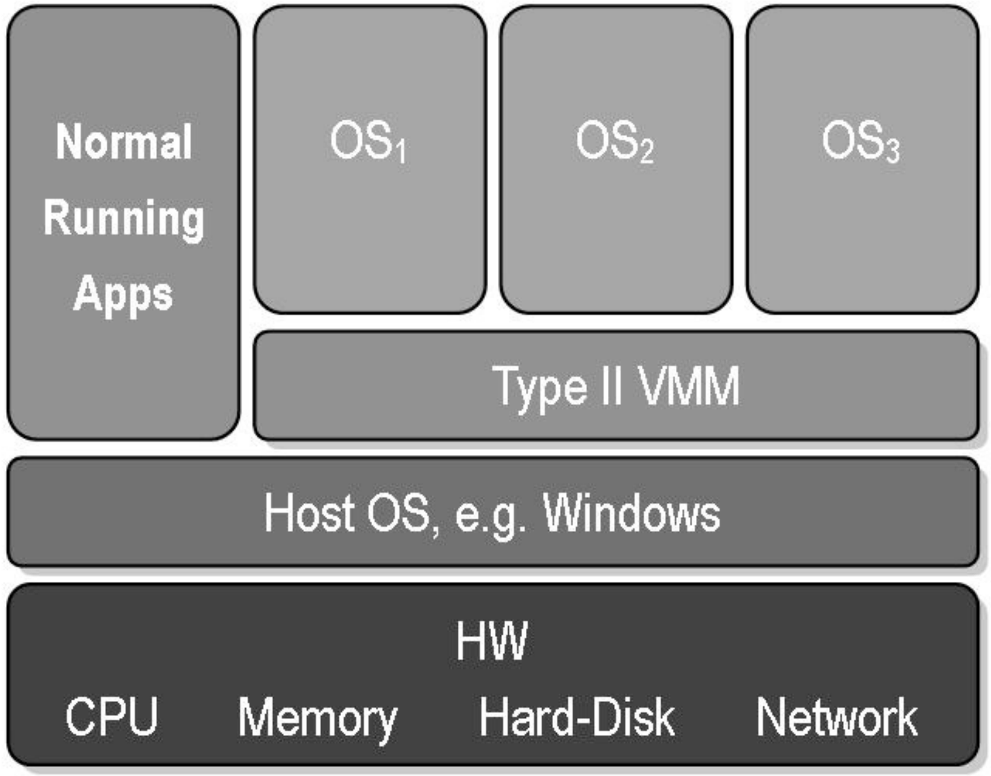
2. Paging supervisor pages out an old block of RAM memory.



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

3.1.5 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