

# CS 547: Foundation of Computer Security

S. Tripathy  
IIT Patna

# Previous Class

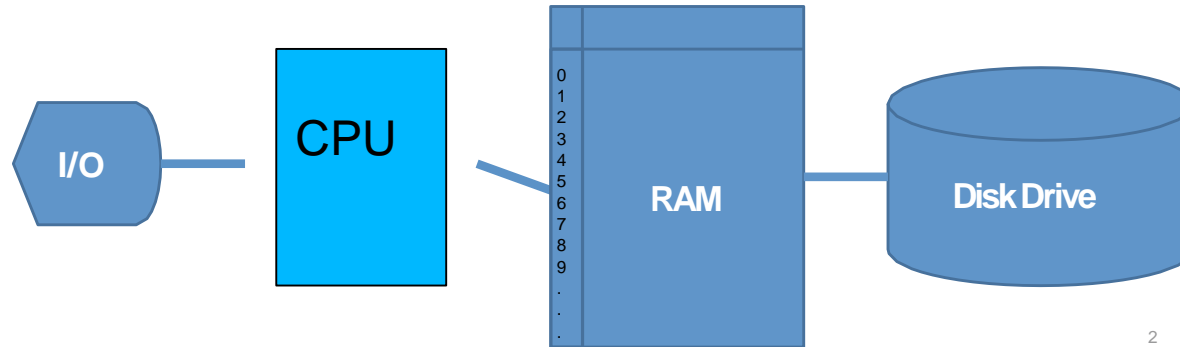
- Program Security
  - Unintentional program flaws
  - B.O, TOCTTOU, Incomplete mediation, Format string attack
  - Malicious code: Malware
    - Viruses
    - Worms
    - Trojan Horse, Logic Bomb, Back Door, Rootkit

# *Present Class*

- Operating System Security
- Protection in General-Purpose Operating Systems
  - Segmentation and Paging
  - Linux File System
  - Access Control Matrix
    - Access Control Lists vs. Capabilities

# Operating System

- Computer System



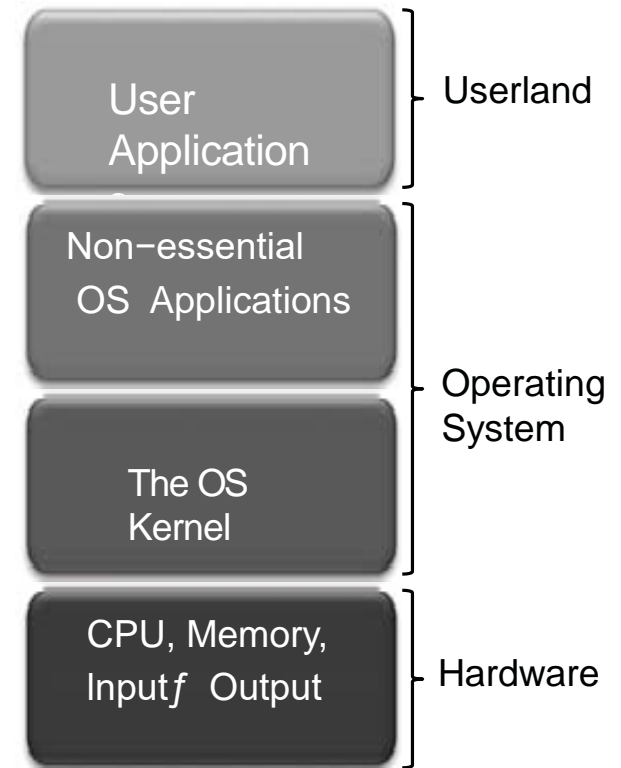
- An operating system has to deal with those Projected objects: CPU
- Memory
- I/O devices (printers, keyboards,...)
- Disks
- Programs
- Data
- Networks

# 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.

# Kernel

- Kernel: the core component of the operating system.
  - handles the management of low-level hardware resources, including memory, processors, I/O devices,
- Most operating systems define the tasks associated with the kernel in terms of a **layer** metaphor, with the hardware components on the bottom, and users and applications being on the top.



# Input/Output

- Input/Output

- It includes 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 device uses a **device driver**, which encapsulates the details of how interaction with that device should be done.
- The application programmer interface (**API**), allows the application programs to interact with those devices at a fairly **high level**, while the **operating system** performs the **low-level** interactions

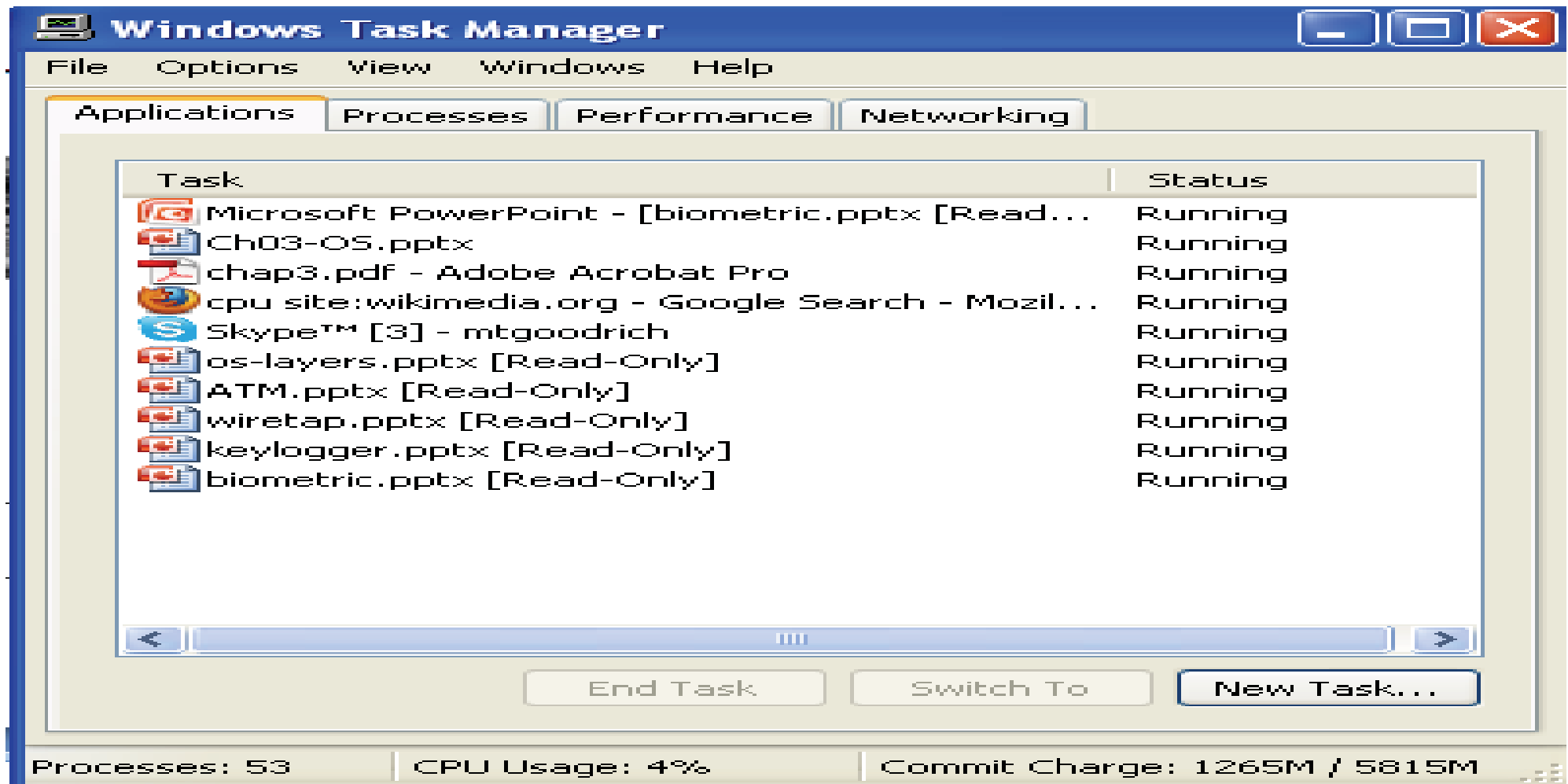
# System calls

- User applications don't communicate directly with low-level hardware components, and instead **delegate such tasks to the kernel via system calls.**
- System calls:
  - usually contained in a collection of programs, that is, a library such as the C library (libc),
  - 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: performing file I/ O (open, close, read, write) and running application programs (exec).



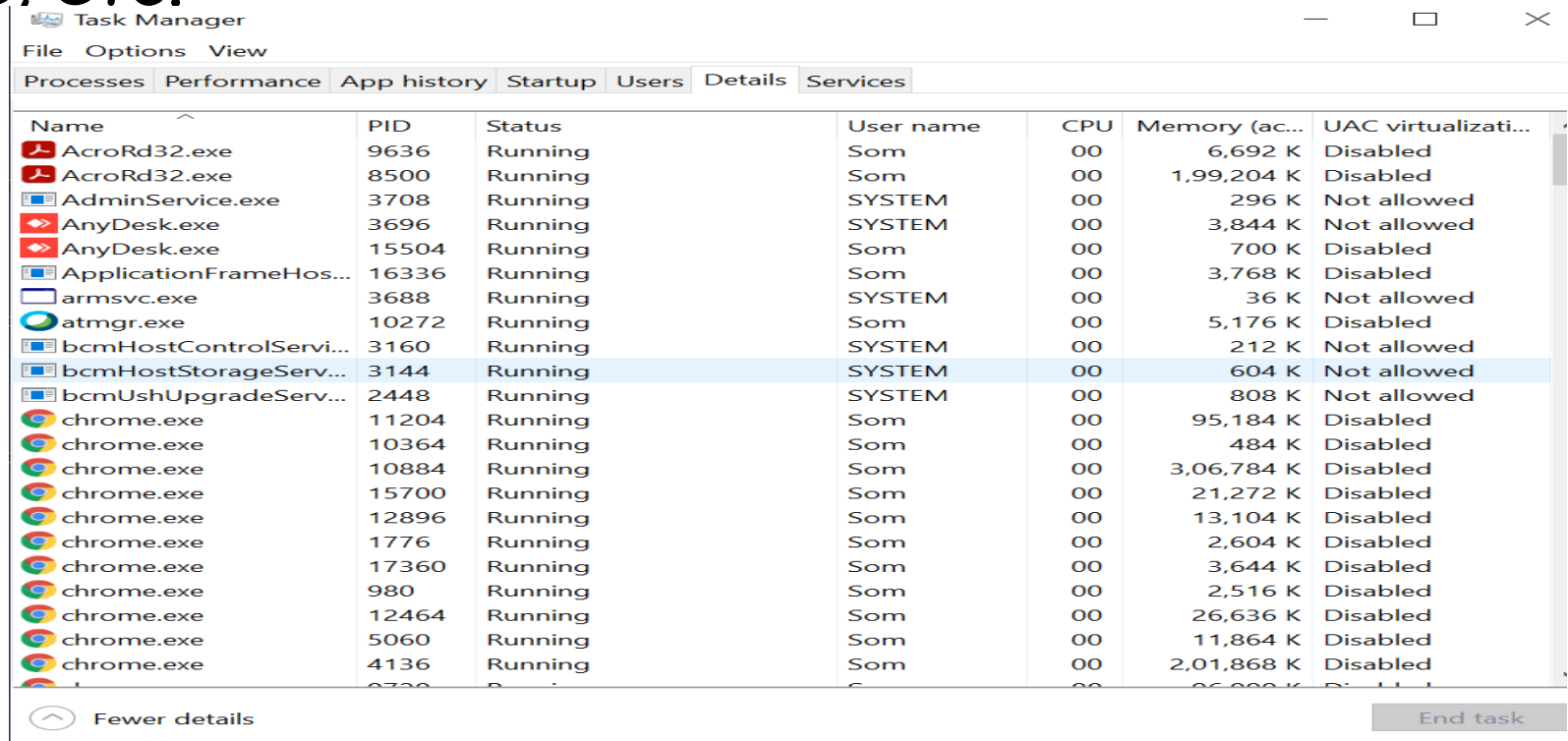
# Process

- Process:



# Process id

- 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.



The screenshot shows the Windows Task Manager application with the 'Details' tab selected. The table lists various running processes, including AcroRd32.exe, AdminService.exe, AnyDesk.exe, ApplicationFrameHos..., armsvc.exe, atmgr.exe, bcmHostControlServi..., bcmHostStorageServ..., bcmUshUpgradeServ..., and multiple instances of chrome.exe. The columns displayed are Name, PID, Status, User name, CPU, Memory (ac..., and UAC virtualizati... (partially visible).

Name	PID	Status	User name	CPU	Memory (ac...	UAC virtualizati...
AcroRd32.exe	9636	Running	Som	00	6,692 K	Disabled
AcroRd32.exe	8500	Running	Som	00	1,99,204 K	Disabled
AdminService.exe	3708	Running	SYSTEM	00	296 K	Not allowed
AnyDesk.exe	3696	Running	SYSTEM	00	3,844 K	Not allowed
AnyDesk.exe	15504	Running	Som	00	700 K	Disabled
ApplicationFrameHos...	16336	Running	Som	00	3,768 K	Disabled
armsvc.exe	3688	Running	SYSTEM	00	36 K	Not allowed
atmgr.exe	10272	Running	Som	00	5,176 K	Disabled
bcmHostControlServi...	3160	Running	SYSTEM	00	212 K	Not allowed
bcmHostStorageServ...	3144	Running	SYSTEM	00	604 K	Not allowed
bcmUshUpgradeServ...	2448	Running	SYSTEM	00	808 K	Not allowed
chrome.exe	11204	Running	Som	00	95,184 K	Disabled
chrome.exe	10364	Running	Som	00	484 K	Disabled
chrome.exe	10884	Running	Som	00	3,06,784 K	Disabled
chrome.exe	15700	Running	Som	00	21,272 K	Disabled
chrome.exe	12896	Running	Som	00	13,104 K	Disabled
chrome.exe	1776	Running	Som	00	2,604 K	Disabled
chrome.exe	17360	Running	Som	00	3,644 K	Disabled
chrome.exe	980	Running	Som	00	2,516 K	Disabled
chrome.exe	12464	Running	Som	00	26,636 K	Disabled
chrome.exe	5060	Running	Som	00	11,864 K	Disabled
chrome.exe	4136	Running	Som	00	2,01,868 K	Disabled

# File Systems

- File system:
  - an abstraction of how the external, nonvolatile memory of the computer is organized.
- OS typically organizes files hierarchically into folders, called directories.



# File Permissions

- File Permissions

- 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.

```
/home/fac/som$ls -l
total 160
drwxr-xr-x 7 som fac 4096 Oct 23  2017 15_nov
drwxr-xr-x 2 som fac 4096 Feb 13  2017 2017-JCST-Rcache
drwxr-xr-x 5 som fac 4096 Oct 25  2017 2017-P2P
-rwxr-xr-x 1 som fac 6759 Aug 10 15:07 a.out
drwx--x--x 2 som fac 4096 Aug 24  2017 asmbly
drwxr-xr-x 3 som fac 4096 Aug 10  2013 ASSIGNMENT
drwxr-xr-x 2 som fac 4096 Aug 10  2016 compsec
drwx----- 5 som fac 4096 Feb 10  2010 CS222
drwx----- 43 som fac 4096 Jul 30  2016 cs222-asmbly
```

# Memory Management

- The (RAM) primary 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.
  - The basic security concerns require that we never mix up these different segments.

# Operating System Security

- An operating system allows different users to access different resources in a **shared way**
- The operating system needs to control this sharing and provide an interface to allow this access
- **Protection:**
  - Mechanism that prevent accident or intentional misuse of a system
- **Aspects of Protection**
  - Identification, authentication and authorization are required for the **access control**

# History

- Operating systems evolved as a way to allow multiple users use the same hardware
  - Sequentially (based on **executives**)
  - Interleaving (based on **monitors**)
- OS makes resources available to users if required by them and permitted by some policy
- OS also protects users from each other
  - Attacks, mistakes, resource overconsumption
- Even for a single-user OS, protecting a user from him/herself is a good thing
  - Mistakes, malware

# Separation

- Keep one user's objects separate from other users
- **Physical** separation
  - Use different physical resources for different users
  - Easy to implement, but expensive and inefficient
- **Temporal** separation
  - Execute different users' programs at different times
- **Logical** separation
  - User is given the impression that no other users exist
- **Cryptographic** separation
  - Encrypt data and make it unintelligible to outsiders
  - Complex

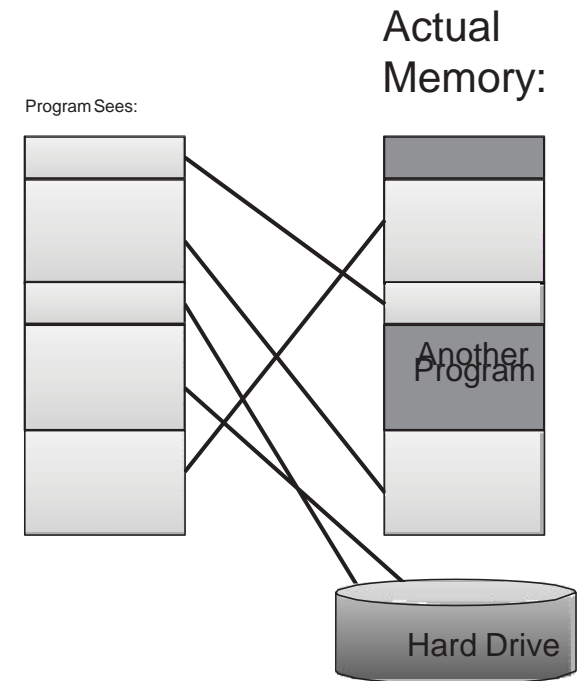


# Sharing

- Sometimes, users want to share resources
  - Library routines (e.g., libc)
  - Files or database records
- OS should allow **flexible sharing**, not “all or nothing”
  - Which files or records? Which part of a file/record?
  - Which other users?
  - Can other users share objects further?
  - What uses are permitted?
    - Read but not write, view but not print (Feasibility?)
    - Aggregate information only
  - For how long?

# Virtual memory

- Memory id 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 fault?



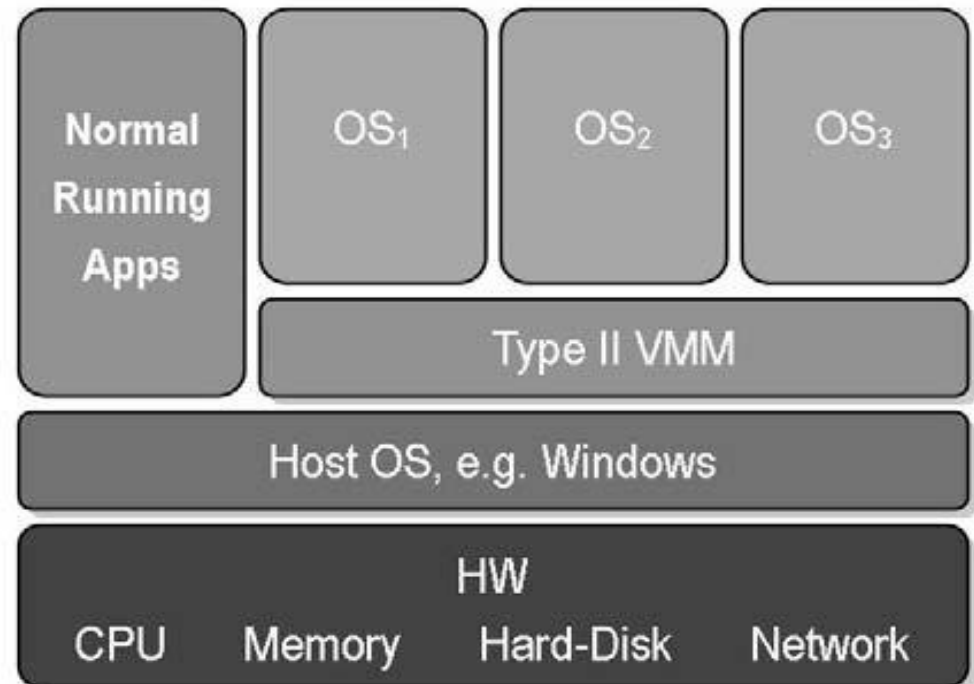
# Virtual machines

- Virtual Machines

- 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:

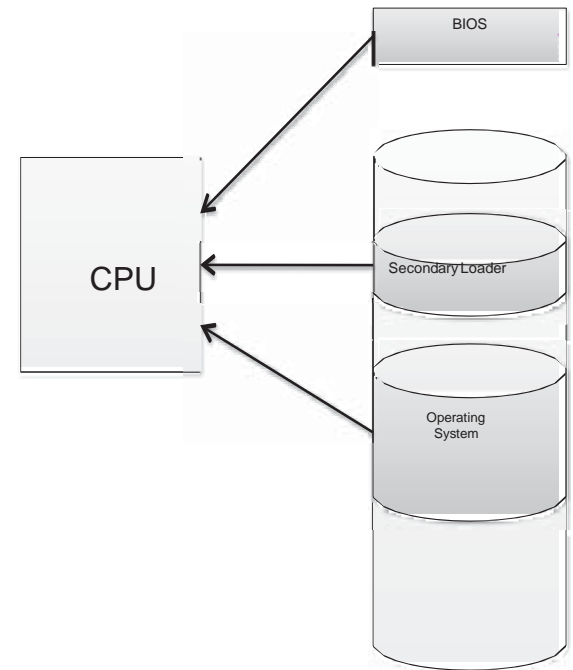
- Hardware Efficiency
- Portability
- Security
- Management



- Booting:

# Boot Sequence

- The action of loading OS into memory from a powered-off state
- When computer is turned on, it first executes code stored in a firmware component known as the BIOS (basic input/output system).
- On modern systems, the BIOS loads into memory the second-stage boot loader, which handles loading the rest of the operating system into memory and then passes control of execution to the operating system
- **An attacker attack from initiating the first stages of booting,**
  - So many computers feature a **BIOS password** that does not allow a second- stage boot loader to be executed without proper authentication



# Hibernation

- Hibernation:
  - Modern machines have the ability to go into a powered-off state known as **hibernation**.
    - OS stores the contents of machine's memory into a **hibernation file** (such as hiberfil.sys) on disk so the computer can be quickly restored later.
- But... without additional security precautions, hibernation exposes a machine to potentially invasive forensic investigation.

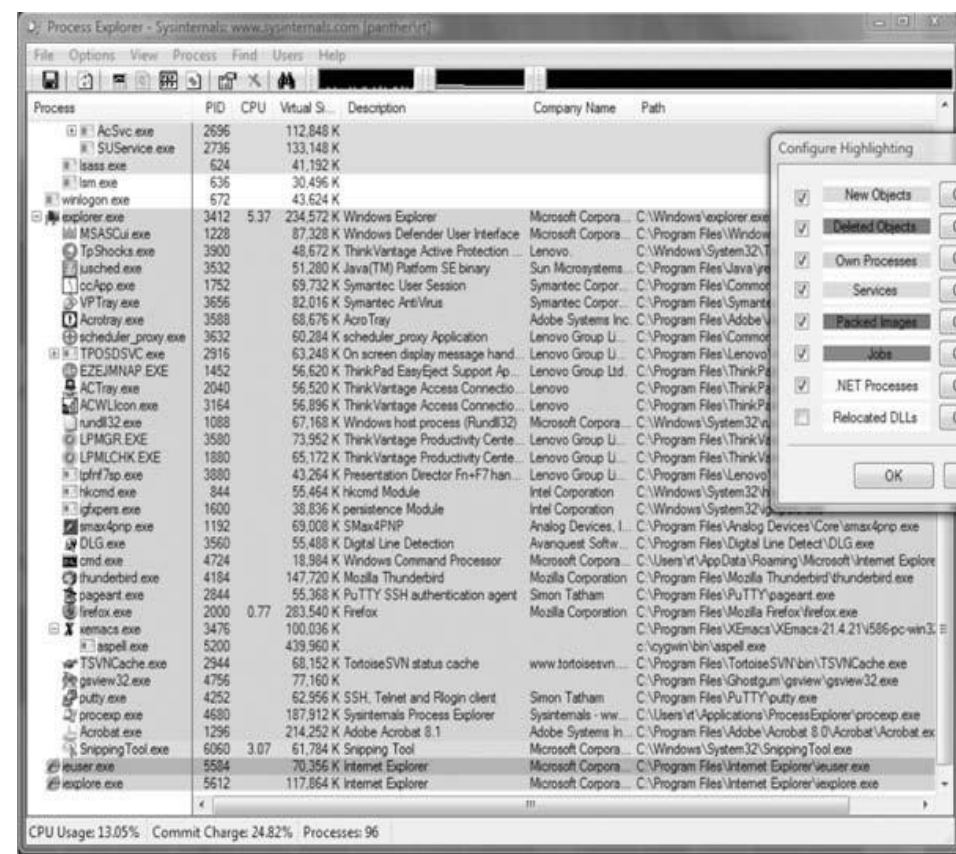


1. User closes a laptop computer, putting it into hibernation.



2. Attacker copies the hiberfil.sys file to discover any unencrypted passwords that were stored in memory when the computer was put into hibernation.

- Keeping track of what processes are running, what other machines have interacted with the system via the Internet, and if the operating system has experienced any unexpected or suspicious behavior can often leave important clues not only for troubleshooting ordinary problems, but also for determining the cause of a security breach.



# Memory and File-system Security

- The contents of a computer are encapsulated in its memory and file system.
- Thus, protection of a computer's content has to start with the protection of its memory and its file system.

# Security Methods of Operating Systems

- Want to be able to share resources without compromising security
  - Do not protect
  - Isolate different processes
  - Share all or nothing
  - Share via access limitation (**granularity**)
  - Share by capabilities
  - Limit use of an object



# Memory and Address Protection

- Prevent program from corrupting other programs or data, maybe operating system also
- Often, the OS can exploit hardware support for this protection, so it's cheap
  -
- Memory protection is part of translation from virtual to physical addresses
  - Memory management unit (MMU) generates exception if something is wrong with virtual address or associated request
  - OS maintains mapping tables used by MMU and deals with raised exceptions

# Memory Organizations

Simplest: *uniprogramming*

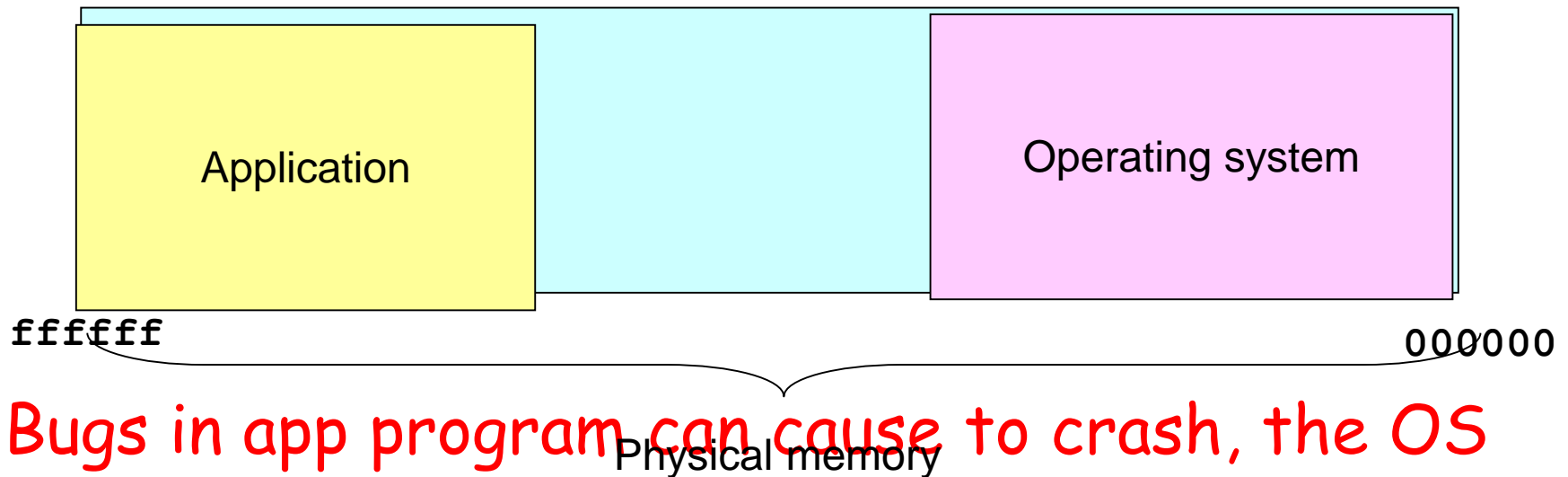
- Each application runs within a hardwired range of physical memory addresses

One application runs at a time

- Application can use the same physical addresses every time, across reboots

# Uniprogramming Without Memory Protection

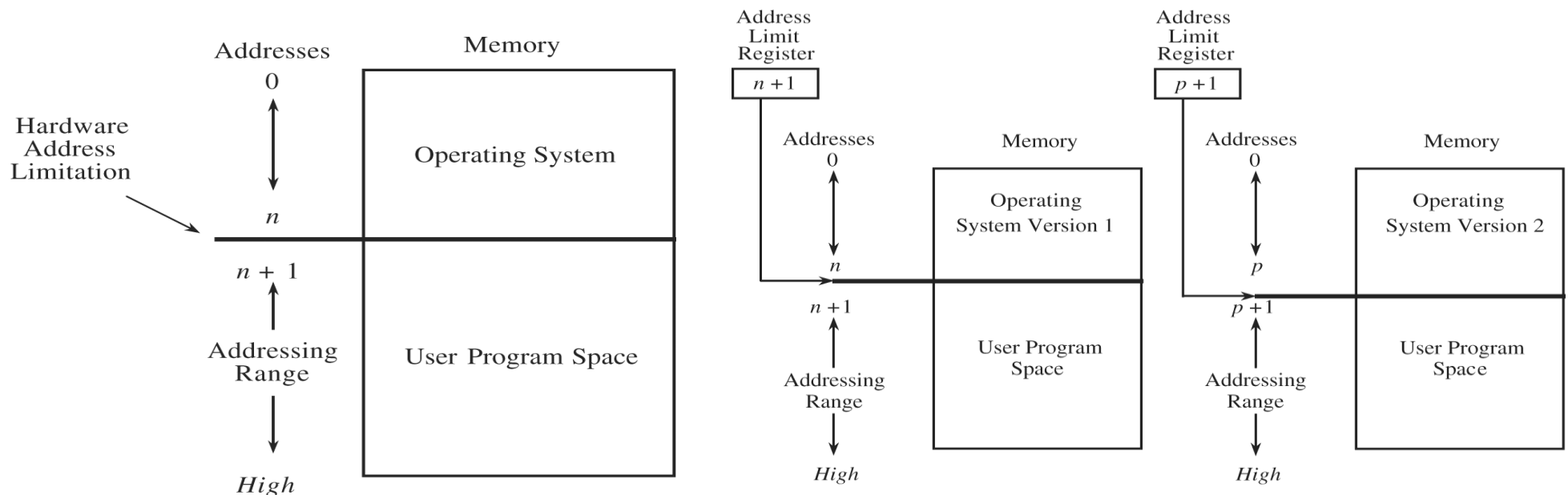
- Applications typically use the higher memory addresses
- An OS uses the lower memory addresses
- An application can address any physical memory location



- Bugs in app program can cause to crash, the OS

# Fence Register – Memory Protection

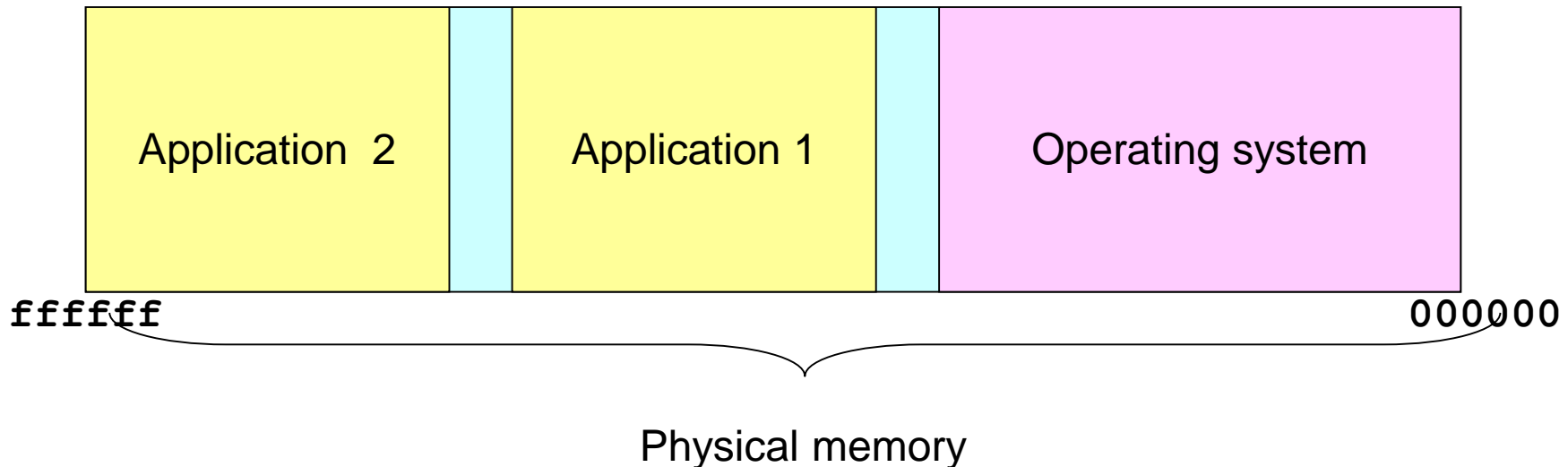
- Fence register
  - Simplest of all protection
  - Confine the user to one side of a boundary
  - Used to separate OS and Program (wasteful use of space)
  - Protects a user from an OS but not a user from another user
  - Single user only
  - Exception if memory access below address in fence register
  - Protects operating system from user programs



# Multiprogramming Without Memory Protection

When a program is copied into memory, a *linker-loader* alters the code of the program (e.g., loads, stores, and jumps)

- To use the address of where the program lands in memory



Bugs in any program can cause other programs to crash, even the OS

# Base/Bound Register

- Base/bounds register pair

- Created for a multiuser environment
- Exception if memory access below/above address in base/bounds register
- Different values for each user program
- Maintained by operating system during context switch
- Limited flexibility

