CSCD 303
Essential Computer Security
Fall 2017

Lecture 7 - Operating System Prinicples and Design

Overview

- Learning Objectives
  - Understand the functions of an Operating System (OS)
  - Learn about different kernel designs
  - Security Principles in OS design

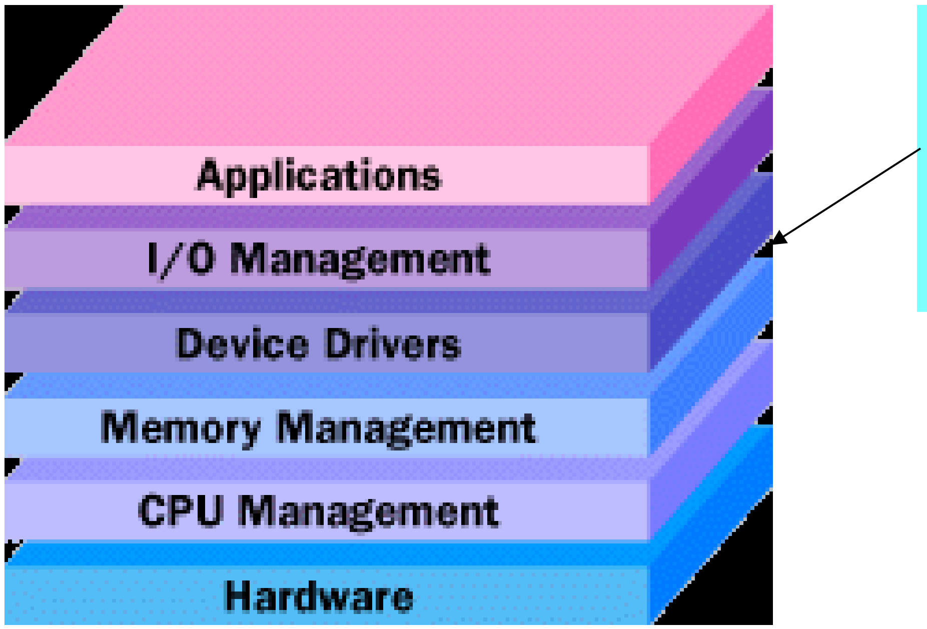
Operating System Design

- Operating systems and their design critical to overall security
- Foundation of security
- Poor choices allow greater access to important resources and make it much more difficult to secure
- Also, are universal security principles should be considered when designing any system

Operating Systems (OS) Functions

- What do OS's do?
  - Operating system's tasks, in most general sense, fall into six categories:
  - Processor management
  - Memory management
  - Device management
  - Storage management
  - Application interface
  - User interface

Operating System Functions

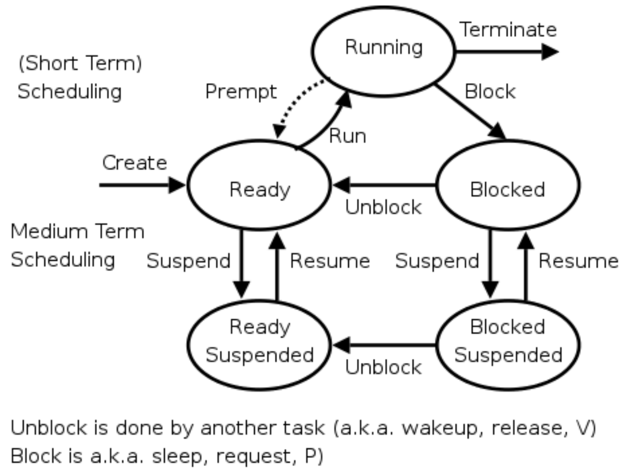


Picture from HowStuffWorks.com

OS controls every task of your computer plus access to all the computer resources

OS Functions

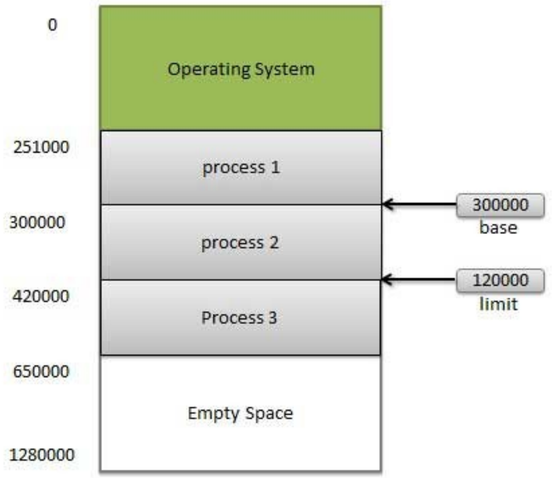
Scheduler Tasks



- Processor Management
  - Allows multiple processes to share resources of processor fairly
  - Does this by scheduling processes to get execution time
    - User Processes
    - Microsoft Word, Foxfire or Skype
    - System Processes:
    - Print spooler, network connections, security accounts manager ... plus many others

Memory Storage and Management

When an operating system manages computer's memory, there are two broad tasks to be accomplished:



1. Each process must have enough memory in which to execute, and respect memory boundaries
2. Different types of memory must be used properly so that each process can run most effectively such as
   - Cache, RAM and Virtual Memory

- First task requires operating system to set up memory boundaries for types of software and for individual applications

Device Management

- Path between operating system and virtually all hardware not on computer's motherboard goes through special program called a Device Driver
  - Driver's function by translating between hardware subsystems and high-level programming languages of operating system and application programs

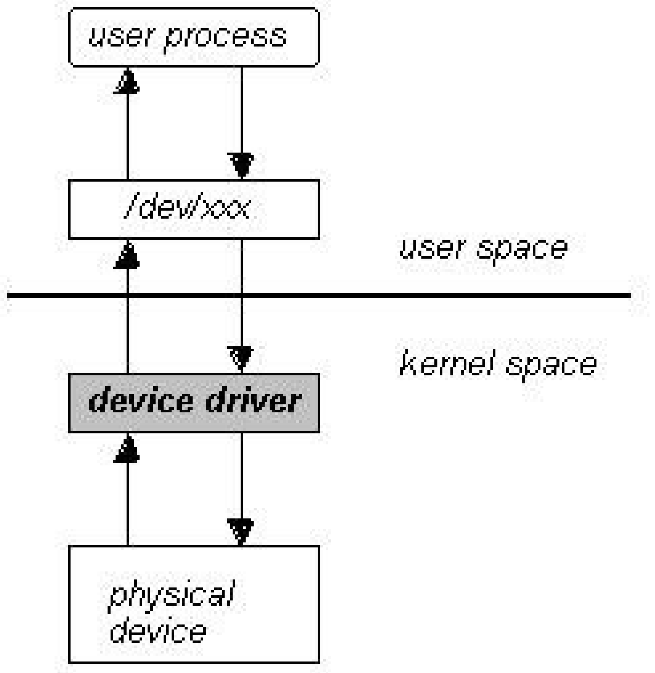
Device Management

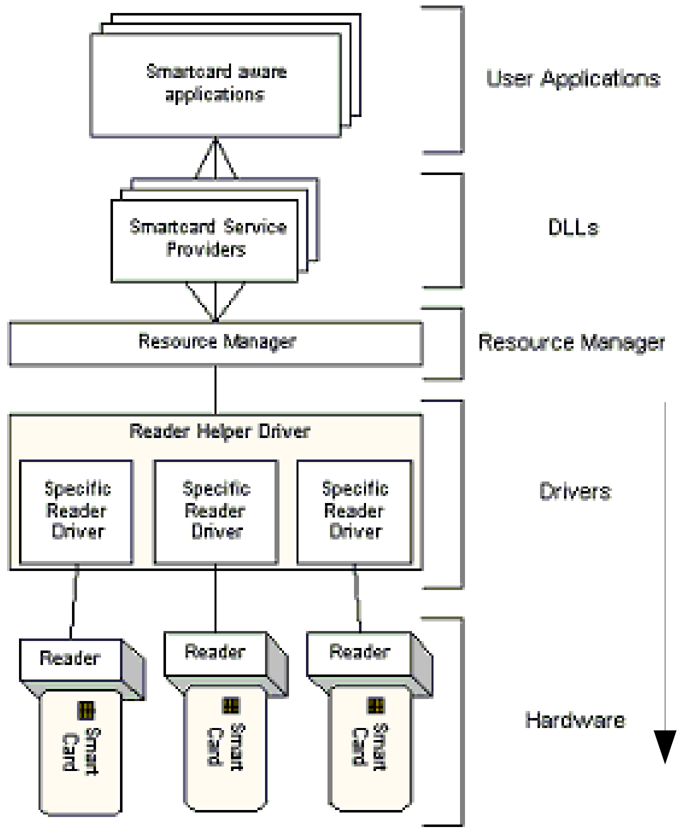
- Drivers separate from operating system so that new functions can be added to drivers
- Plus, new drivers added
- Without requiring operating system itself to be modified, recompiled and redistributed

Device Drivers

Linux

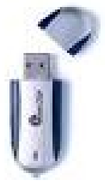
Windows Smart Card





Kernel space

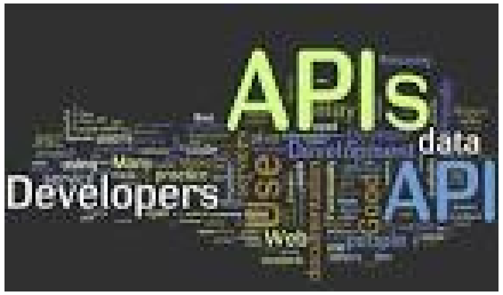
Storage Management



- Manages and Organizes disk resources
  - Includes temporary devices
  - CD and DVD drives, thumb drives, external drives

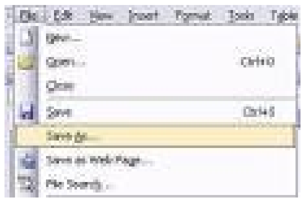
- Creates file systems for storing both OS types of files and user files
- Must somehow keep track of who can access these files – Access Control

Application Programming Interface (API)



- Drivers provide for applications to make use of subsystems without having to know every detail of internal operations
  - Application Program Interfaces (APIs) lets application programmers use OS functions without having to directly keep track of details in CPU's operation
  - Hides details of processor and other resources from program

Application Programming Interface (API)



- For Example
  - Microsoft Word or Open Office Word Processor
  - You click, “Save file”
  - If didn't have an API
    - Word or Open Office would have to know all details of file system and ultimately call disk controller to create file on disk
    - Instead, language program is written in, C or Java has a function that is mapped to operating system API for creating file

User Interface

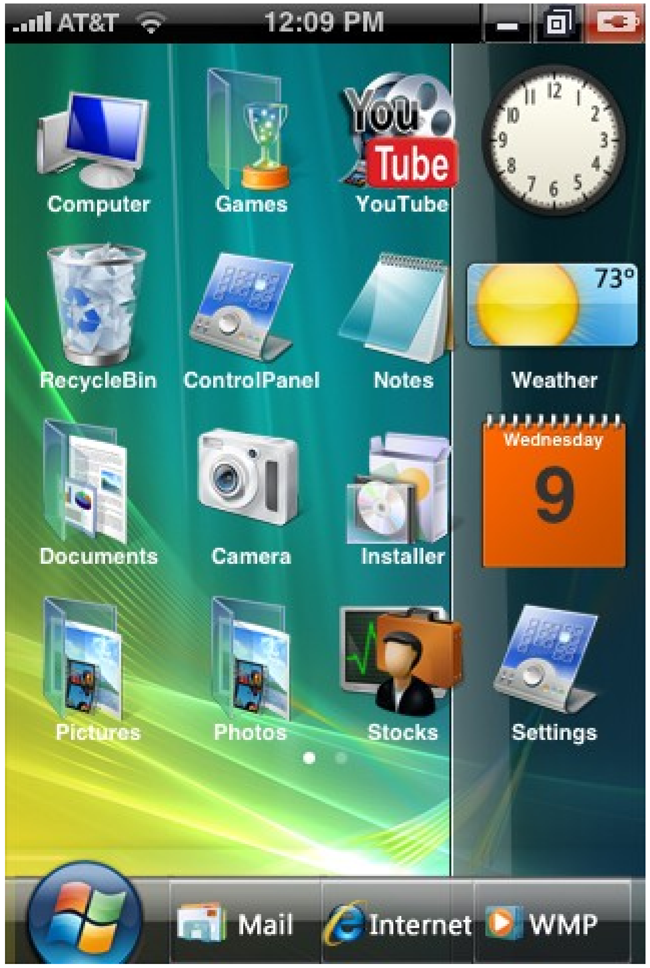
MAC
OS X

- User Interface (UI) brings structure to interaction between user and computer
  - In last decade, almost all development in user interfaces has been in area of graphical user interface (GUI),
  - Provides the “look and feel” of the computer

User Interface

Max OS X

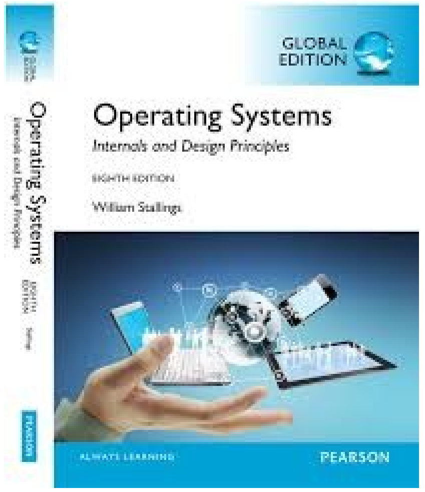
Windows Vista



Linux Ubuntu

Further Study Operating Systems

- Please see CSCD340 taught by
  Stu Steiner
- For the BCS ... do not have to take this course
- I highly recommend you take this course
- It is core Computer Science
- Can't really know about developing applications without knowing about the Operating System ....



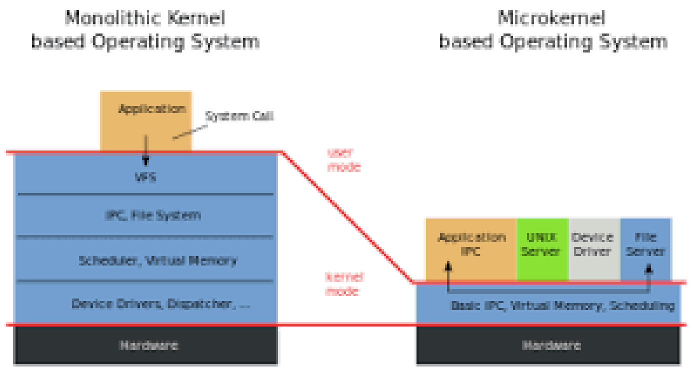
OS Design Decisions

vs

- The kernel is the heart of the OS and manages most of the functionality of the OS including access to device hardware
- Have been competing designs for the kernel
  - Monolithic vs Micro Kernel vs Hybrid Kernel Designs
  - These design decisions affect ultimate security of OS

Nice resource for Operating System Concepts

http://www.brokenthorn.com/Resources/OSDevIndex.html

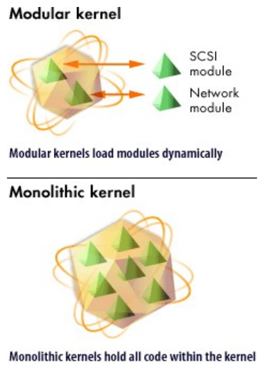


Operating System Design

OS Design Decisions

- Monolithic Design
  - Integrate a great deal of functionality into OS core
  - Services are interdependent
  - Core system is larger
  - Every flaw in system is exposed through services that depend on that system

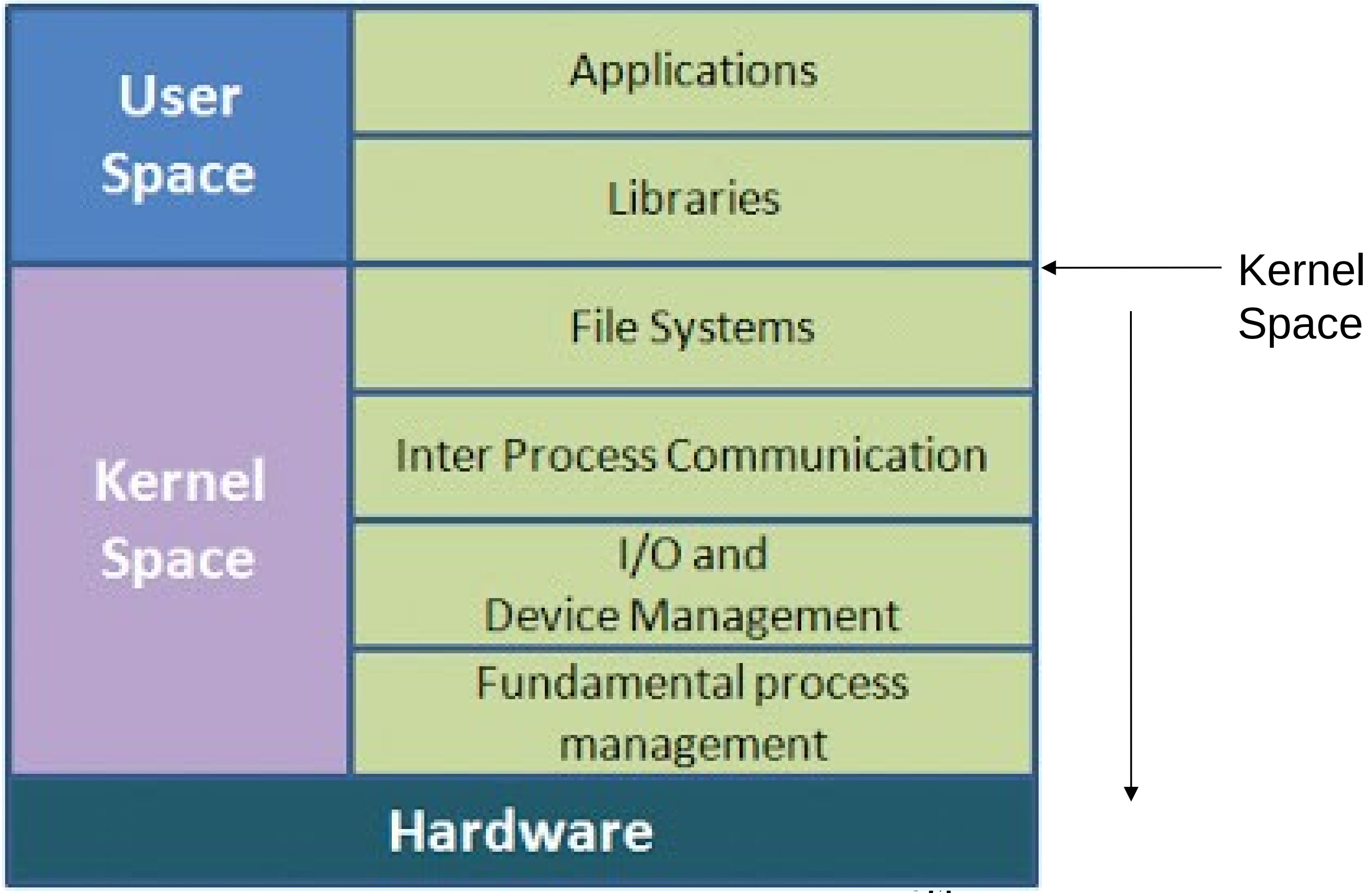
Monolithic Kernel


Modular kernel
SCSI module
Network module
Modular kernels load modules dynamically
Monolithic kernel
Monolithic kernels hold all code within the kernel

- Monolithic kernel where all services
- File system, device drivers as well as core functionality scheduling, memory allocation are a tight knit group sharing same space
- Do not confuse term modular kernel to be anything but monolithic
- Some monolithic kernels can be compiled to be modular
- What matters is that module is inserted to and run from same space that handles core functionality
- Examples of Monolithic Kernel?

  Linux, BSDs (FreeBSD, OpenBSD, NetBSD), Solaris, OS-9, AIX, HP-UX, DOS, Microsoft Windows (95,98,Me)

Monolithic Kernel Diagram


User Space
Applications
Libraries
Kernel Space
File Systems
Inter Process Communication
I/O and Device Management
Fundamental process management
Hardware
Kernel Space

Monolithic Kernel Pros and Cons

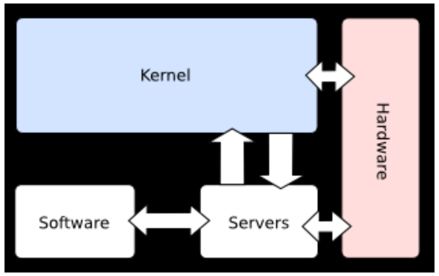
- Pros

  * More direct access to hardware for programs
  * Easier for processes to communicate between each other
  * If your device is supported, it should work with no additional installations
  * Processes react faster because there is more direct access to CPU

- Cons

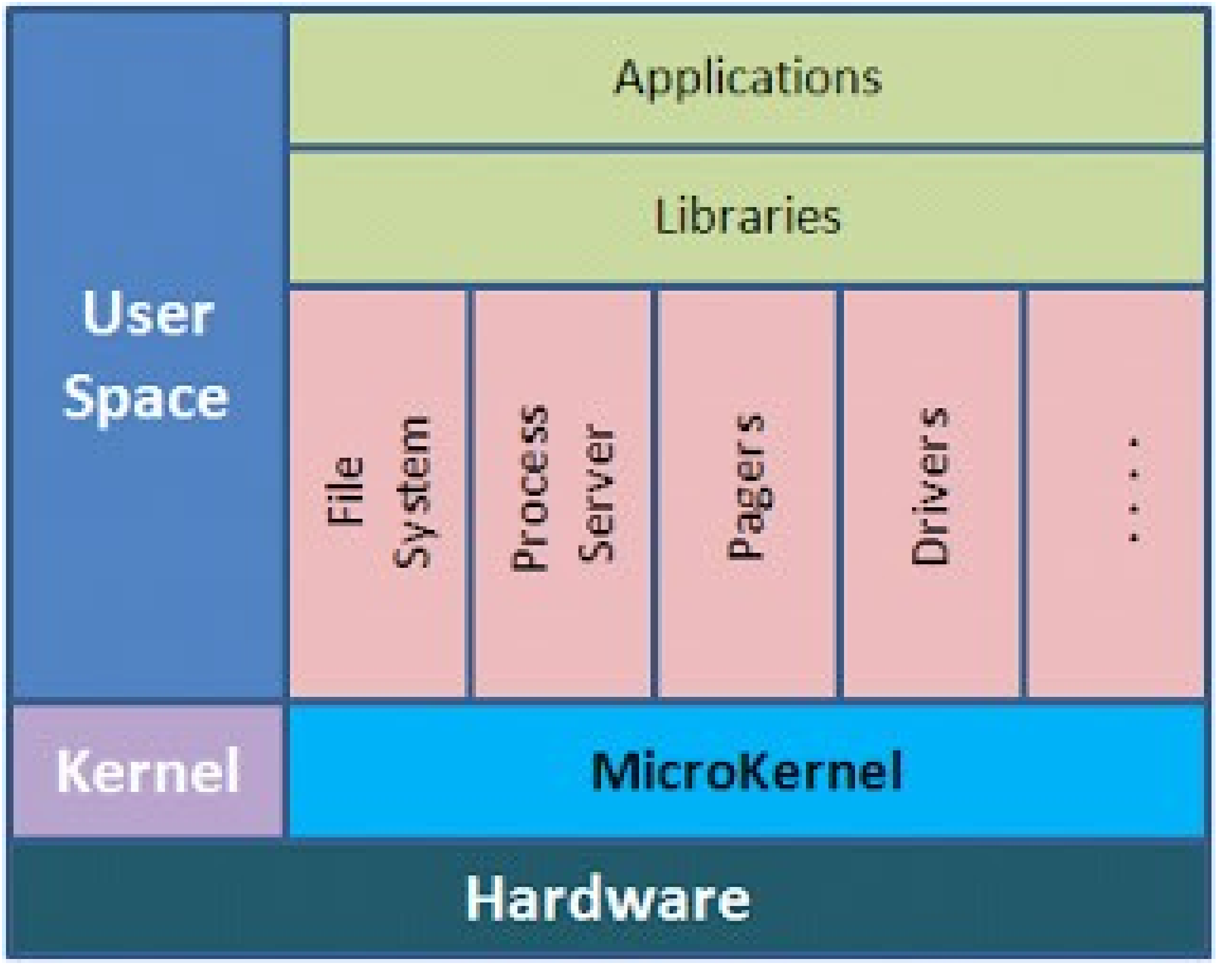
  * Large install footprint
  * Large memory footprint
  * Less secure because everything runs in supervisor or privileged mode

Micro Kernel



- A micro kernel
- Core functionality is isolated from system services and device drivers
- For instance, VFS (virtual file system) and block device file systems are separate processes that run outside kernel's space,
  - Using IPC to communicate with kernel, other services and user processes
  - IPC means Interprocess Communication
- Example of Micro kernel? Minix

Micro Kernel Architecture



Kernel Mode

Micro Kernel Pros and Cons

- Pros
  - Portability
  - Small memory footprint
  - Security Better - not as much runs in supervisor or privileged mode
- Cons
  - Hardware is more abstracted through drivers
  - Hardware may react slower because drivers are in user mode
  - Processes have to wait in a queue to get information
  - Processes can’t get access to other processes without waiting

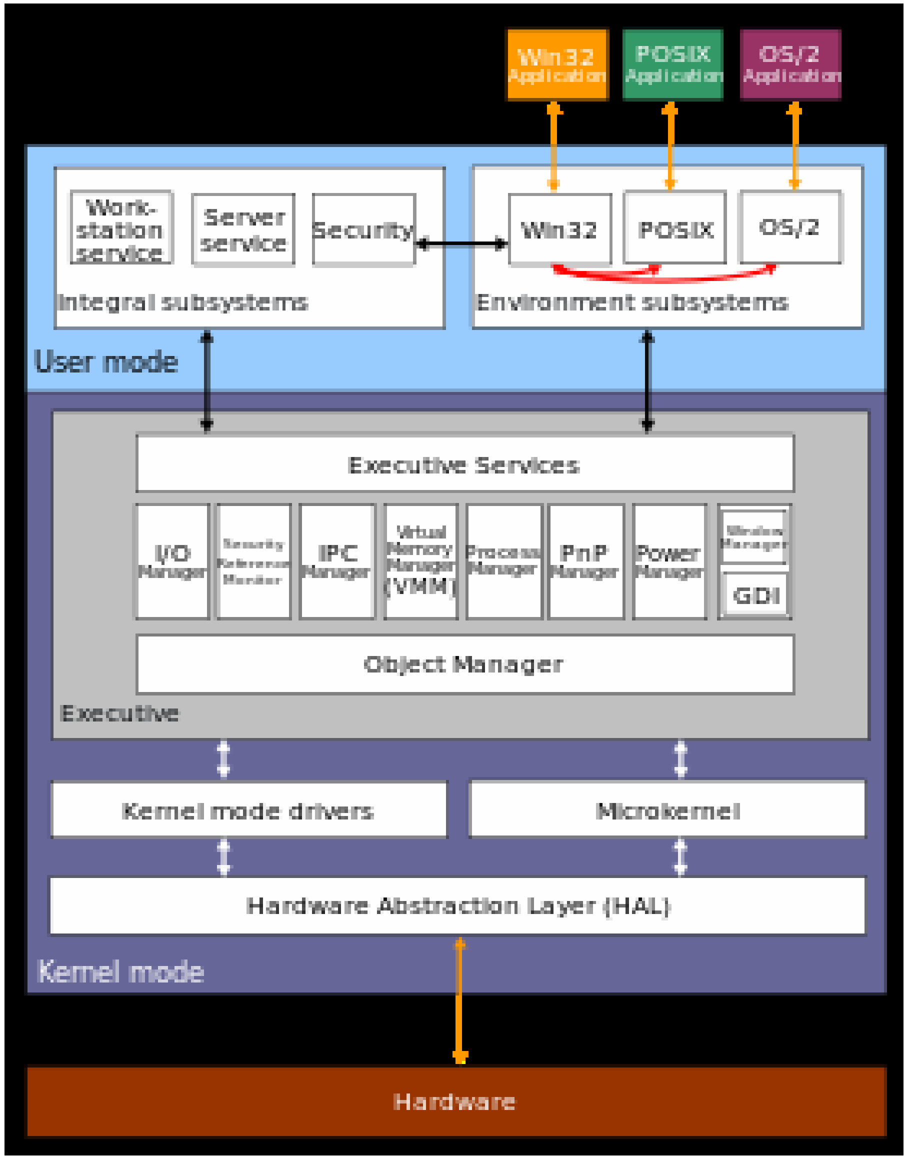
Hybrid Kernel

- Hybrid kernel is architecture based on combining aspects of microkernel and monolithic kernel architectures
- A hybrid kernel runs some services in kernel space to reduce performance overhead of microkernel, while still running kernel code as “servers” in user space
- For instance, hybrid kernel design may keep Virtual File System and bus controllers inside kernel and file system drivers and storage drivers as user mode programs outside the kernel
  - Such a design keeps the performance and design principles of a monolithic kernel.
- Whats an example of a Hybrid kernel?

  Windows NT, Windows 2000, Windows XP, Windows Server 2003,

  Windows Vista, Windows Server 2008 and Windows 7

Hybrid Kernel Architecture



Hybrid Kernel Pros and Cons

- Pros
  * Developer can pick and choose what runs in user mode and what runs in supervisor mode
  * Smaller install footprint than monolithic kernel
  * More flexible than other models

- Cons
  * Can suffer from same process lag as microkernel
  * Device drivers need to be managed by user (typically)

Designing for Security

Principles of Secure Design

- Least Privilege
- Fail Safe Defaults
- Economy of Mechanism
- Complete Mediation
- Defense in depth
- Open Design
- Separation of Privilege
- Least Common Mechanism
- Psychological Acceptability

- Where did these principles come from?

Where did the Principles Originate?

- Multics time-sharing system was an early multi-user system put significant effort into ensuring security
- Jerome Saltzer, security researcher, wrote an article outlining the security mechanisms in Multics Operating system in 1974
- Following year, Saltzer and Michael Schroeder expanded article into a tutorial titled

“The Protection of Information in Computer Systems” (Saltzer and Schroeder, 1975)

First section of paper introduced “basic principles” of information protection, including confidentiality, integrity, and availability, plus a set of design principles

http://www.cs.virginia.edu/~evans/cs551/saltzer/

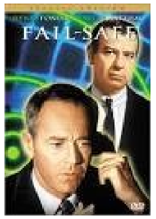
Principle of Least Privilege

- A subject should only be given privileges it needs to complete its task and no more
- Privileges should be controlled by function, not identity
  - Similar to need to know principle from military secrecy
- System benefits of Least Privilege?

Principle of Least Privilege

- Purpose ...
- As with most security mechanisms, to make it difficult for unauthorized access to occur
    - Without inconveniencing legitimate access
    - Need-to-know also aims to discourage "browsing" of sensitive material by limiting access to smallest possible number of people.

Principle of Fail-Safe Defaults



- Unless explicit access has been granted, access should be denied
  - Why is this better for security?

Reasons for Fail-Safe Defaults

A conservative design must be based on arguments why objects should be accessible, rather than why they should not.

In a large system some objects will be inadequately considered, so a default of lack of permission is safer

Principle of Fail-Safe Defaults

From Matt Bishop .. Security Textbook Author, UC Davis Professor

“This principle requires that the default access to an object is none Whenever access, privileges, or some security related attribute is not explicitly granted, it should be denied”

Furthermore, if subject is unable to complete its action or task, before subject terminates, it should undo those changes it made to the security state of the system. This way, even if the program fails, the system is still safe “

Principle of Fail-Safe Defaults

Example

- If mail server is unable to create a file in spool directory, it should close the network connection, issue an error message, and stop
- It should not try to store the message elsewhere, nor expand its privileges to save the message in another location
- Protections on mail spool directory itself should allow create and write access to only the mail server, and read and delete access to only the local server
- No other user should have access to the directory

Principle of Economy of Mechanism

- Security mechanisms should be as simple as possible. What is the benefit of this?
- The idea behind this principle is that simple systems tend to be more secure.
- One factor in evaluating a system's security is its complexity. If the design, implementation, or security mechanisms are highly complex, then the likelihood of security vulnerabilities increases

Economy of Mechanism

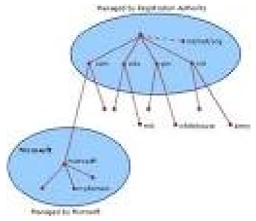
- This well-known principle applies to any system, but it deserves emphasis for protection mechanisms for this reason
  - Design and implementation errors that result in unwanted access paths will not be noticed during normal use
  - Since normal use usually does not include attempts to exercise improper access paths
    - Techniques such as line-by-line inspection of software and physical examination of hardware that implements protection mechanisms are necessary
    - For such techniques to be successful, a small and simple design is essential.

Principle of Complete Mediation

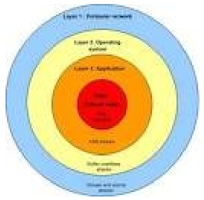
- All accesses to objects must be checked to ensure that they are still allowed
  - Whenever a subject attempts to read an object, the operating system should mediate the action
  - First, it determines if the subject can read the object. If so, it provides the resources for the read to occur.
  - If the subject tries to read the object again
  - What should happen?
  - System should again check that the subject can still read the object
  - Most systems would not make the second check. They would cache the results of the first check, and base the second access upon the cached results

Principle of Complete Mediation

- Example of Violation
- The Directory Name Service (DNS) caches information mapping hostnames into IP addresses
- If an attacker is able to "poison" the cache by implanting records associating a bogus IP address with a name, the host will route connections to that host incorrectly



Principle of Defense in Depth



- More lines of defense there are against an attacker,
  - Better the defense,
  - Additional mechanisms should be different
- Strategy based on military principle that it is more difficult for an enemy to defeat a complex and multi-layered defense system than to penetrate a single barrier
- Can you give an example related to Computer Security?

Principle of Defense in Depth

- Example: Let's use bank security. Why is typical bank more secure than the typical convenience store?
- Because there are many redundant security measures protecting the bank, and the more measures there are, the more secure the place is.
- Security cameras alone are a deterrent for some
- But if people don't care about the cameras, then a security guard is there to physically defend the bank with a gun.

Principle of Open Design

- Security of a mechanism should not depend on secrecy of its design or implementation

A system relying on security through obscurity may have theoretical or actual security vulnerabilities, but its owners or designers believe that if the flaws are not known, then attackers will be unlikely to find them

Has this proven true in reality?

Principle of Open Design

- Open Source vs. Closed Source
  - Article in SC Magazine argues that open source software is as secure or more secure than closed source
  - Why might this be true?

  http://www.scmagazine.com/open-source-software-is-more-secure-than-you-think/article/315374/

Principle of Separation of Privilege

- A system should not grant permission based on a single condition
- Checking access on only one condition may not be adequate for strong security
- If an attacker is able to obtain one privilege but not a second, he or she may not be able to launch a successful attack
- Example
  - BSD systems, su users must belong to the wheel group and know the root password

Principle of Least Common Mechanism

- Mechanisms to access resources should not be shared
- Every shared mechanism represents a potential information path between users and must be designed with great care to be sure it does not unintentionally compromise security

Principle of Least Common Mechanism

- Example
- A web site provides electronic commerce services for a major company
- Attackers want to deprive company of revenue they obtain from that web site
- They flood site with messages, and tie up the electronic commerce services. Legitimate customers are unable to access the web site and, as a result, take their business elsewhere
- Here, the sharing of the Internet with the attackers' sites caused the attack to succeed. The appropriate countermeasure would be to restrict the attackers access to the segment of the Internet connected to the web site. Techniques such as proxy servers or traffic throttling would help

Principle of Psychological Acceptability

- Security mechanisms should not make it more difficult to access a resource.
- If security mechanisms hinder usability or accessibility of resources, then users may opt to turn off those mechanisms
- Where possible, security mechanisms should be transparent to the users of the system or at most introduce minimal obstruction
- Is this principle followed in most Operating Systems?

Summary

- Operating Systems by design helps or hinders security
- Want concepts of security design implemented as much as possible in most of the systems we use …
- Up to developers to design security in from the beginning

The End

- Next Time: Vulnerabilities
- Lab: Kali Linux