CYENG 312/GECE 594 :
Trusted Operating System (OS)

Lecture 1: Introduction

Instructor: Shayan (Sean) Taheri, Ph.D.

Assistant Professor

The Department of Electrical and Cyber Engineering (ECE)

The Institute for Health and Cyber Knowledge (I-HACK)

The Gannon University (GU)

Personal Information

- Name: Shayan (Sean) Taheri.
- Date of Birth: July/28/1991.
- Past Position: Postdoctoral Fellow at University of Florida.
- Ph.D. Degree: Electrical Engineering from the University of Central Florida.
- M.S. Degree: Computer Engineering from the Utah State University.
- University Profile: https://www.gannon.edu/FacultyProfiles.aspx?profile=taheri001

Useful Information

- Prerequisite: Operating System (OS) and Programming
- Office hours: Please refer to the Blackboard system.
- Course Components: Laboratory Assignments, Theoretical Assignments, Exams, and Projects.
- Read the textbook materials related to topics that will be covered.
- Study the slides very well.

Class Policies

DO

- Read
  - Book, lab, and datasheets
- Try before seeking help
- Follow announcements
- Discuss material with the instructor
- Do research
- Track due dates

DON'T

- Don't cheat!
- **Never look at another student's code** (current or previous)
- Don't let your partner do all the work
- Don't copy software from book or web without attribution
- Don't expect handholding

What's this course about?

- Some challenging and interesting topics
  - *Learn about attacks on operating systems*
  - *Learn about defenses for operating systems*
- Lectures on many topics
  - *Application security*
  - *Operating system security*
  - *(Software) Application-level, Network-level, and Hardware-level security threats and countermeasures for operating systems*

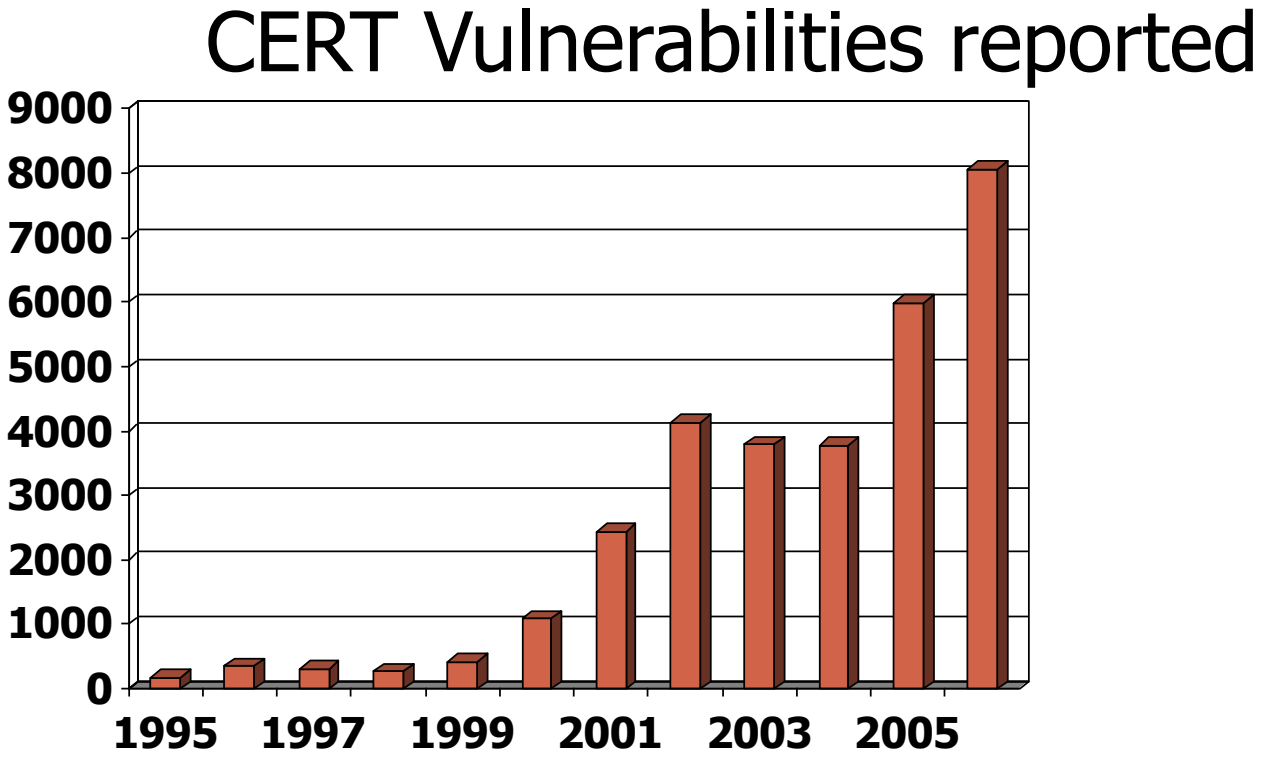
Trends: Some things in the news

- Nigerian letter (419 Scams) still works:
  - *Michigan Treasurer Sends 1.2MUSD of State Funds !!!*
- Many zero-day attacks in 2007
  - *Google, Excel, Word, Powerpoint, Office …*
- Numerous stolen or lost laptops, storage media, containing customer information
  - *Second-hand computers (hard drives) pose risk*
- Phishing fraudsters get flashy
- Vint Cerf estimates ¼ of PCs on Internet are bots
- Undetected for 50 days, Gozi trojan steals data from SSL streams
- Any other attacks?

Example: Steal cars with a laptop

- NEW YORK - Security technology created to protect luxury vehicles may now make it easier for tech-savy thieves to drive away with them.
- In April ‘07, high-tech criminals made international headlines when they used a laptop and transmitter to open the locks and start the ignition of an armor-plated BMW X5 belonging to soccer player David Beckham, the second X5 stolen from him using this technology within six months.
- Beckham's BMW X5s were stolen by thieves who hacked into the codes for the vehicles' RFID chips.

How big is the security problem?

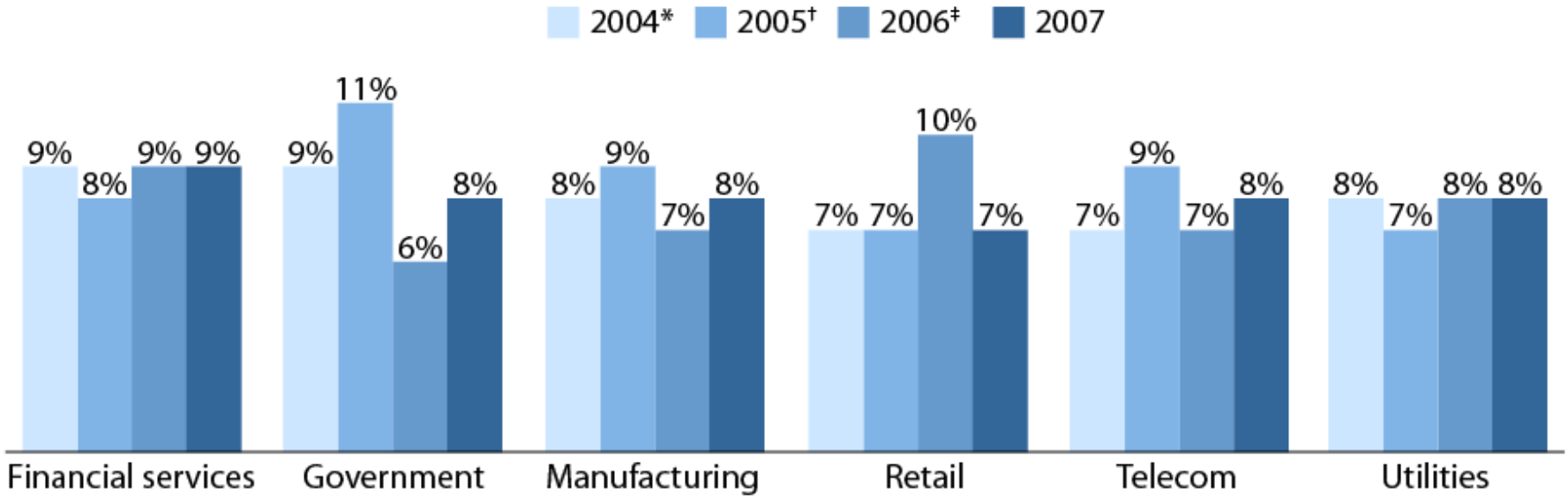


http://www.cert.org/stats/

Security Spending Variance By Industry

January 2007, Trends **"2007 Security Budgets Increase: The Transition To Information Risk Management Begins"**

"Approximately what percentage of your company's overall IT spending will go to security?"

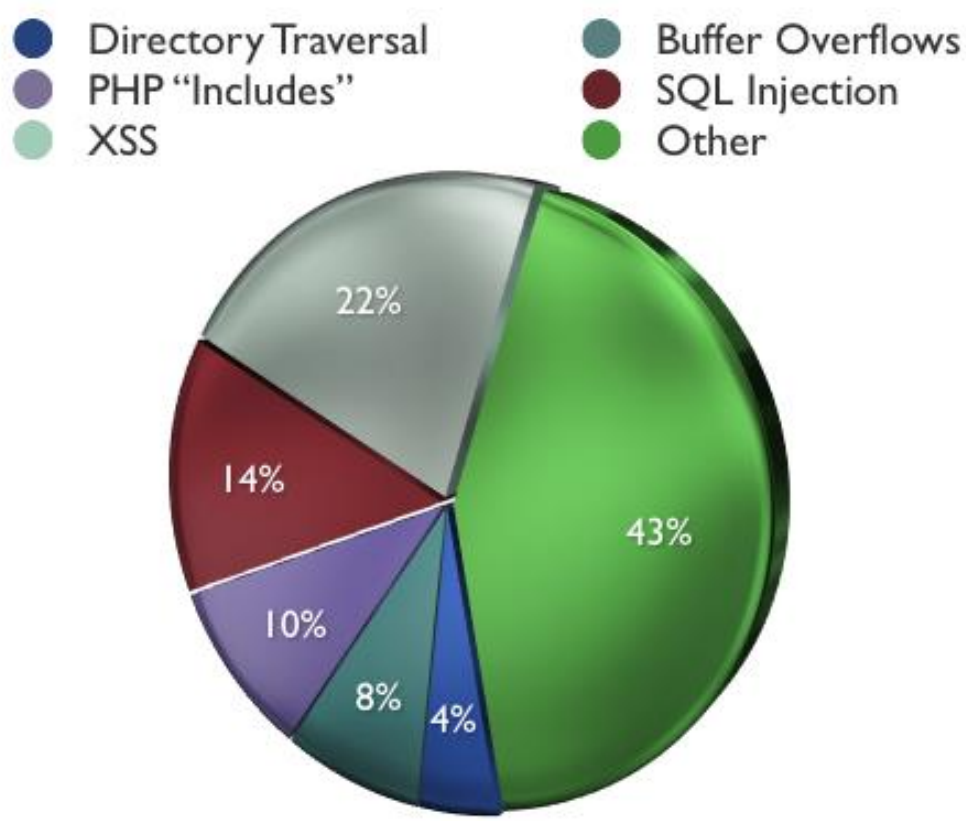


*Base: 604 executives at North American and European enterprises
†Base: 840 executives at North American and European enterprises
‡Base: 616 executives at North American and European enterprises
Base: 447 executives at North American and European enterprises

*Source: Forrester's Business Technographics® June 2004 North American And European Benchmark Study
†Source: Forrester's Business Technographics November 2004 North American And European Benchmark Study
‡Source: Business Technographics November 2005 North American And European Enterprise IT Budgets And Spending Survey
Source: Business Technographics November 2006 North American And European Enterprise IT Budgets And Spending Survey

Most-common attacks on systems

- 2006 MITRE CVE stats:
  - *21.5 % of CVEs were XSS*
  - *14 percent SQL injection*
  - *9.5 percent php "includes"*
  - *7.9 buffer overflow.*



2005 was the first year that XSS jumped ahead of buffer overflows …

Why are there security problems?

- Lots of buggy software...
  - *Why do programmers write insecure code?*
  - *Awareness is the main issue*

- Some contributing factors
  - *Few courses in computer security*
  - *Programming text books do not emphasize security*
  - *Few security audits*
  - *C is an unsafe language*
  - *Programmers are lazy*
  - *Legacy software (some solutions, e.g. Sandboxing)*
  - *Consumers do not care about security*
  - *Security is expensive and takes time*

Ethical use of security information

- We discuss vulnerabilities and attacks
  - *Most vulnerabilities have been fixed*
  - *Some attacks may still cause harm*
  - *Do not try these at home*
- Purpose of this class
  - *Learn to prevent malicious attacks*
  - *Use knowledge for good purposes*

Law enforcement

- Sean Smith
  - *Melissa virus: 5 years in prison, $150K fine*
- Ehud Tenenbaum (“The Analyzer”)
  - *Broke into US DoD computers*
  - *6 mos service, suspended prison, $18K fine*
- Dmitry Sklyarov
  - *Broke Adobe ebooks*
  - *Prosecuted under DMCA*

Difficult Problem: Insider Threat

- Easy to hide code in large software packages
  - *Virtually impossible to detect back doors*
  - *Skill level needed to hide malicious code is much lower than needed to find it*
  - *Anyone with access to development environment is capable*

Example Insider Attack (IA) – 1

- Hidden trap door in Linux, Nov 2003
  - *Allows attacker to take over a computer*
  - *Practically undetectable change*
  - *Uncovered by anomaly in CVS usage*
- Inserted line in wait4()

```
if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
            retval = -EINVAL;
```

  - *Looks like a standard error check*
  - *Anyone see the problem?*

See: http://lwn.net/Articles/57135/

Example IA – 2

- Rob Harris case - slot machines
  - *an insider: worked for Gaming Control Board*
- Malicious code in testing unit
  - *when testers checked slot machines*
    - downloaded malicious code to slot machine
  - *was never detected*
  - *special sequence of coins activated "winning mode"*
- Caught when greed sparked investigation
  - *$100,000 jackpot*

Example IA – 3

- Breeder's cup race
  - *Upgrade of software to phone betting system*
  - *Insider, Christopher Harn, rigged software*
  - *Allowed him and accomplices to call in*
    - change the bets that were placed
    - undetectable
  - *Caught when got greedy*
    - won $3 million

http://horseracing.about.com/library/weekly/aa110102a.htm

Software Dangers

- Software is complex
  - *top metric for measuring #of flaws is lines of code*
- Windows Operating System
  - *tens of millions of lines of code*
  - *new "critical" security bug announced every week*
- Unintended security flaws *unavoidable*
- Intentional security flaws *undetectable*

Ken Thompson

- What code can we trust?
  - *Consider "login" or "su" in Unix*
  - *Is RedHat binary reliable?*
  - *Does it send your passwd to someone?*
- Can't trust binary so check source, recompile
  - *Read source code or write your own*
  - *Does this solve problem?*

Reflections on Trusting Trust, http://www.acm.org/classics/sep95/

Compiler backdoor

- This is the basis of Thompson's attack
  - *Compiler looks for source code that looks like login program*
  - *If found, insert login backdoor (allow special user to log in)*
- How do we solve this?
  - *Inspect the compiler source*

C compiler is written in C

- Change compiler source S

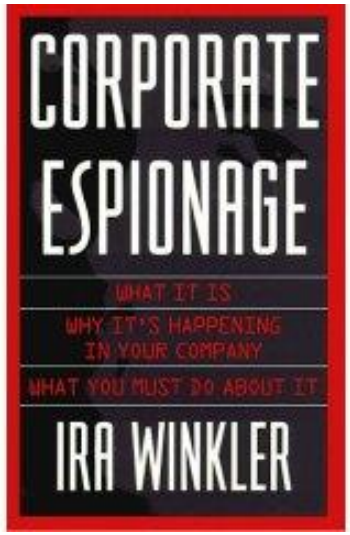
```
compiler(S) {
  if (match(S, "login-pattern")) {
        compile (login-backdoor)
        return
  }
  if (match(S, "compiler-pattern")) {
        compile (compiler-backdoor)
        return
  }
  .... /* compile as usual */
}
```

Clever trick to avoid detection

- Compile this compiler and delete backdoor tests from source
  - *Someone can compile standard compiler source to get new compiler, then compile login, and get login with backdoor*
- Simplest approach will only work once
  - *Compiling the compiler twice might lose the backdoor*
  - *But can making code for compiler backdoor output itself*
    - (Can you write a program that prints itself? Recursion thm)
- Read Thompson's article
  - *Short, but requires thought*

Social engineering

- Many examples
  - *We are not going to talk about social engineering a lot, but good to remember that there are many attacks that don't use computers*
  - *Call system administrator*
  - *Dive in the dumpster*
  - *Online version*
    - send trojan in email
    - picture or movie with malicious code



Trusted Operating System

- An OS is **trusted** if we rely on it for
  - Memory protection
  - File protection
  - Authentication
  - Authorization
- Every OS does these things
- But if a trusted OS fails to provide these, our security fails

Trust Vs. Security

- **Trust** implies reliance
- Trust is binary
- Ideally, only trust secure systems
- All trust relationships should be explicit

- **Security** is a judgment of effectiveness
- Judged based on specified policy
- Security depends on trust relationships

- Note: Some authors use different terminology!

Trusted Operating Systems

- **Trust** implies reliance
- A trusted system is relied on for security
- An untrusted system is not relied on for security
- If all untrusted systems are compromised, your security is unaffected
- Ironically, **only a trusted system can break your security!**

Trusted OS

- OS mediates interactions between subjects (users) and objects (resources)
- Trusted OS must decide
  - Which objects to protect and how
  - Which subjects are allowed to do what

General Security Principles

- Least privilege — like “low watermark”
- Simplicity
- Open design (Kerchoffs Principle)
- Complete mediation
- White listing (preferable to black listing)
- Separation
- Ease of use
- But commercial OSs emphasize features
  - Results in complexity and poor security

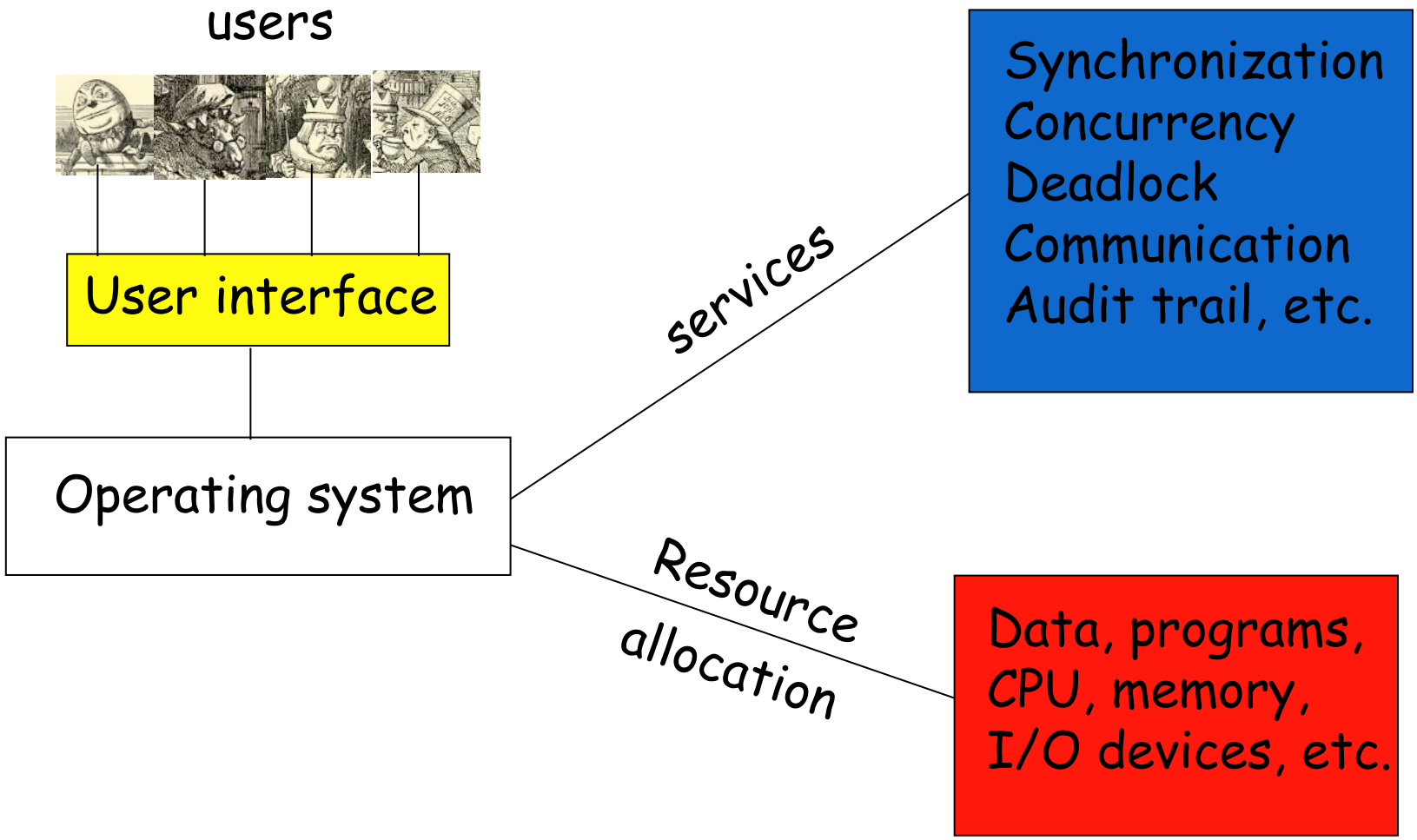
General Security Principles

- Least privilege — like “low watermark”
- Simplicity
- Open design (Kerchoffs Principle)
- Complete mediation
- White listing (preferable to black listing)
- Separation
- Ease of use
- But commercial OSs emphasize features
  - Results in complexity and poor security

OS Security

- Any OS must provide some degree of
  - Authentication
  - Authorization (users, devices and data)
  - Memory protection
  - Sharing
  - Fairness
  - Inter-process communication/synchronization
  - OS protection
  - Results in complexity and poor security

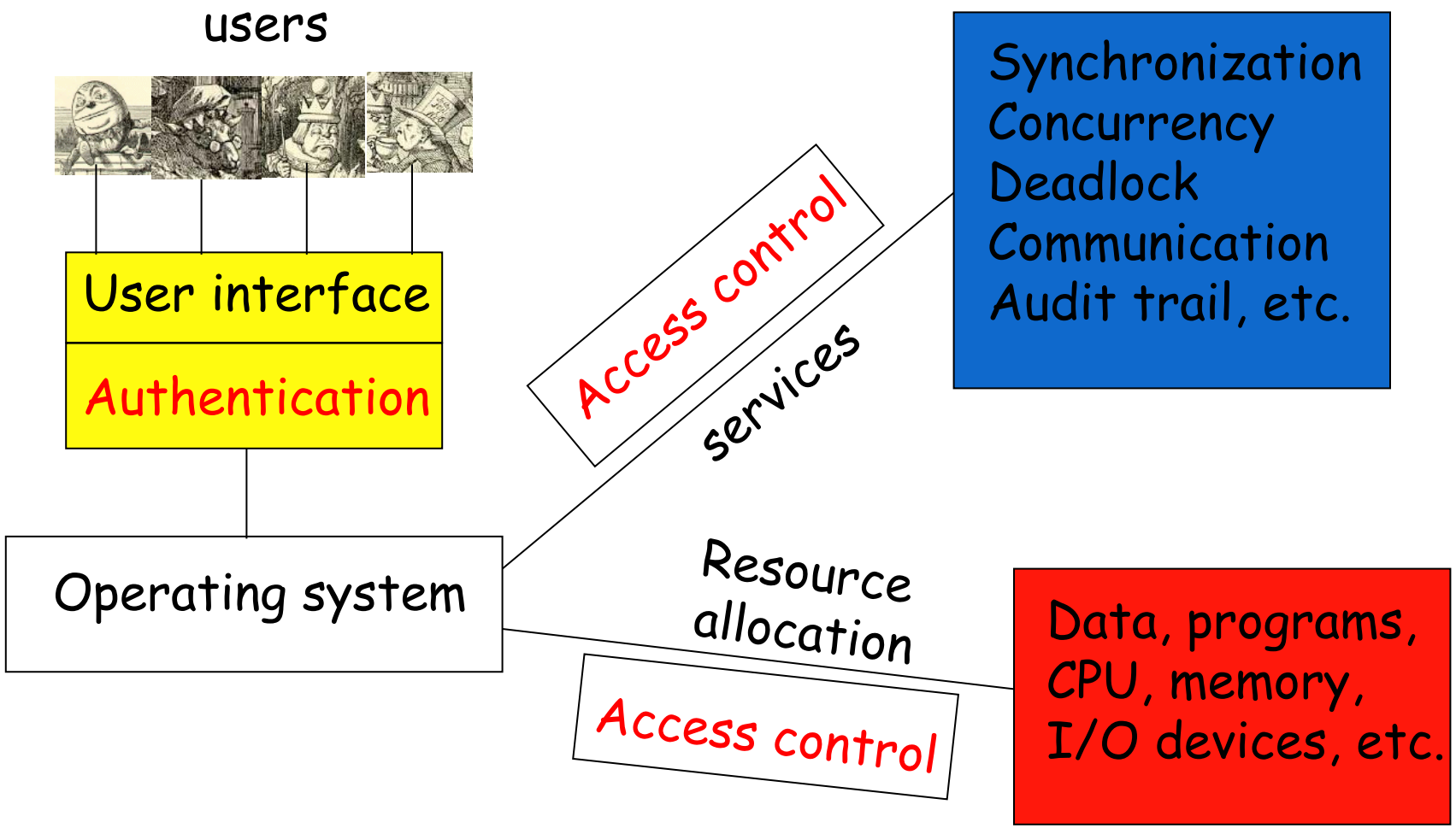
OS Services


users
User interface
Operating system
services
Synchronization
Concurrency
Deadlock
Communication
Audit trail, etc.
Resource
allocation
Data, programs,
CPU, memory,
I/O devices, etc.

Trusted OS

- A trusted OS also provides some or all of
  - User authentication/authorization
  - Mandatory access control (MAC)
  - Discretionary access control (DAC)
  - Object reuse protection
  - Complete mediation - access control
  - Trusted path
  - Audit/logs

Trusted OS Services



MAC and DAC

- Mandatory Access Control (MAC)
  - Access not controlled by owner of object
  - Example: User does not decide who holds a TOP SECRET clearance
- Discretionary Access Control (DAC)
  - Owner of object determines access
  - Example: UNIX/Windows file protection
- If DAC and MAC both apply, MAC wins

Object Reuse Protection

- OS must prevent leaking of info
- Example
  - User creates a file
  - Space allocated on disk
  - But same space previously used
  - "Leftover" bits could leak information
  - Magnetic remanence is a related issue

Trusted Path

- Suppose you type in your password
  - *What happens to the password?*
- Depends on the software!
- How can you be sure software is not evil?
- Trusted path problem

> *"I don't know how to to be confident even of a digital signature I make on my own PC, and I've worked in security for over fifteen years. Checking all of the software in the critical path between the display and the signature software is way beyond my patience. "*
>
> —*Ross Anderson*

Audit

- System should log security-related events
- Necessary for postmortem
- What to log?
  - *Everything? Who (or what) will look at it?*
  - *Don't want to overwhelm administrator*
  - *Needle in haystack problem*
- Should we log incorrect passwords?
  - *"Almost" passwords in log file?*
- Logging is not a trivial matter

Security Kernel

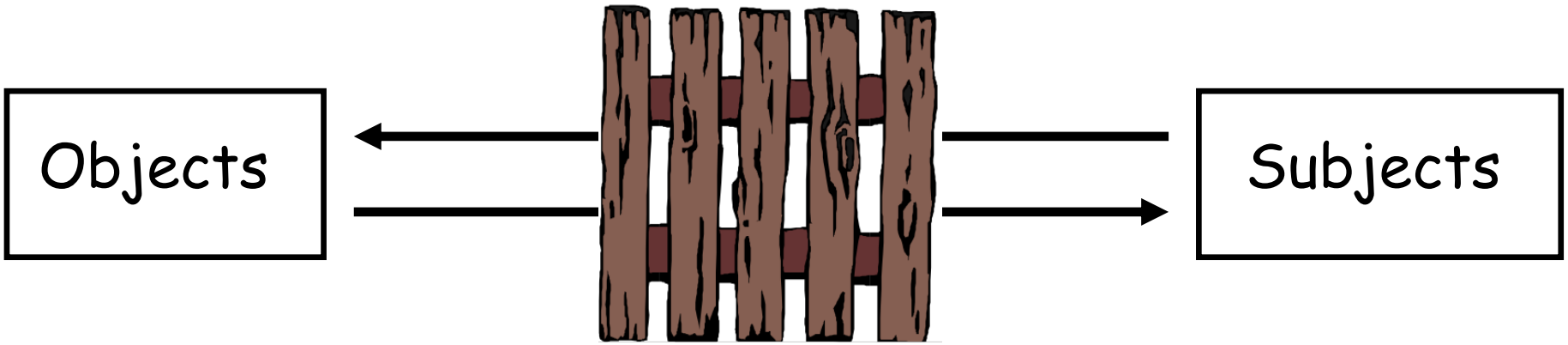
- **Kernel** is the lowest-level part of the OS
- Kernel is responsible for
  - *Synchronization*
  - *Inter-process communication*
  - *Message passing*
  - *Interrupt handling*
- The **security kernel** is the part of the kernel that deals with security
- Security kernel contained within the kernel

Security Kernel

- Why have a security kernel?
- All accesses go thru kernel
  - *Ideal place for access control*
- Security-critical functions in one location
  - *Easier to analyze and test*
  - *Easier to modify*
- More difficult for attacker to get in “below” security functions

Reference Monitor

- The part of the security kernel that deals with access control
  - *Mediates access of subjects to objects*
  - *Tamper-resistant*
  - *Analyzable (small, simple, etc.)*



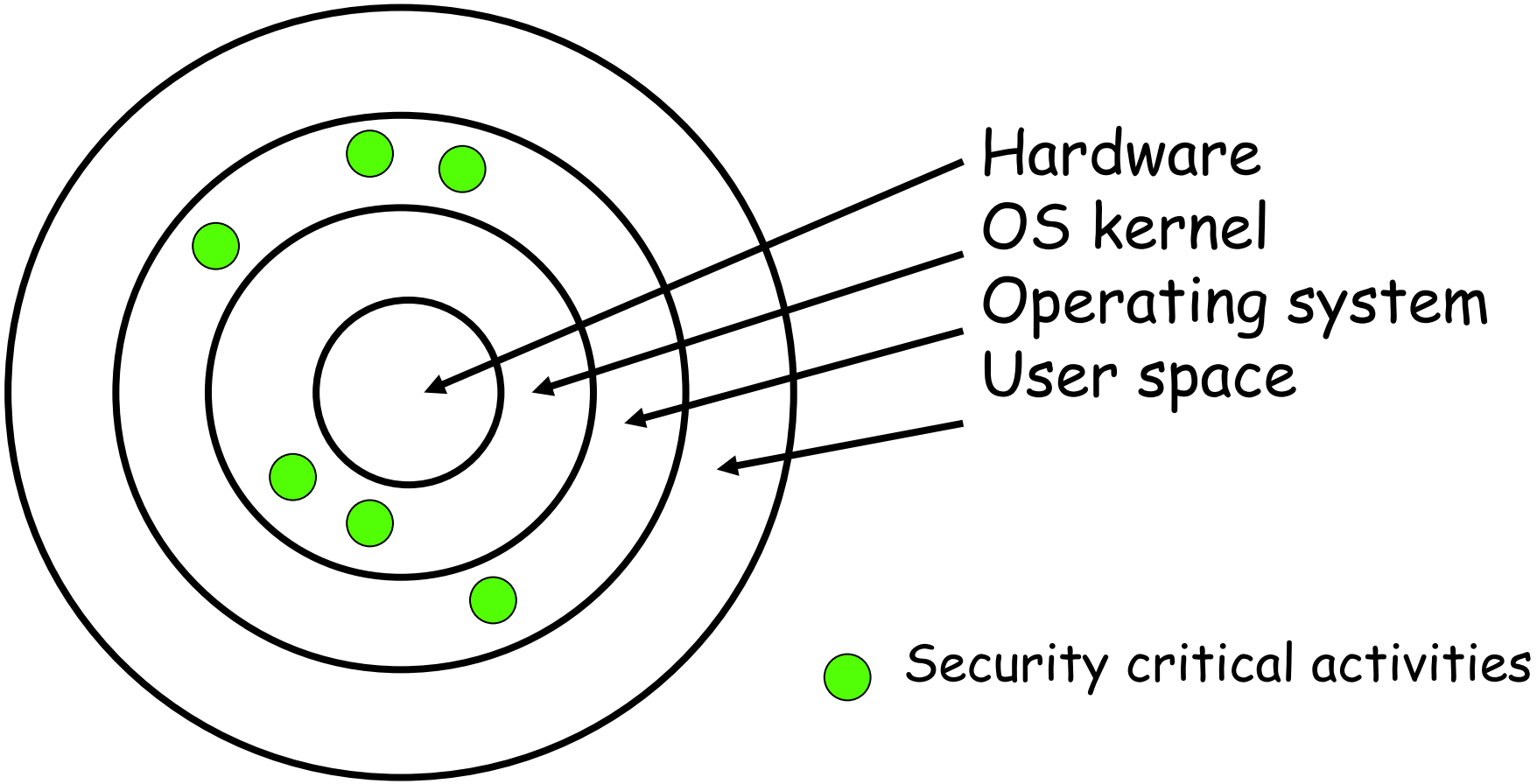
Trusted Computing Base

- **TCB** — everything in the OS that we rely on to enforce security
- If everything outside TCB is subverted, trusted OS would still be trusted
- TCB protects users from each other
  - *Context switching between users*
  - *Shared processes*
  - *Memory protection for users*
  - *I/O operations, etc.*

TCB Implementation

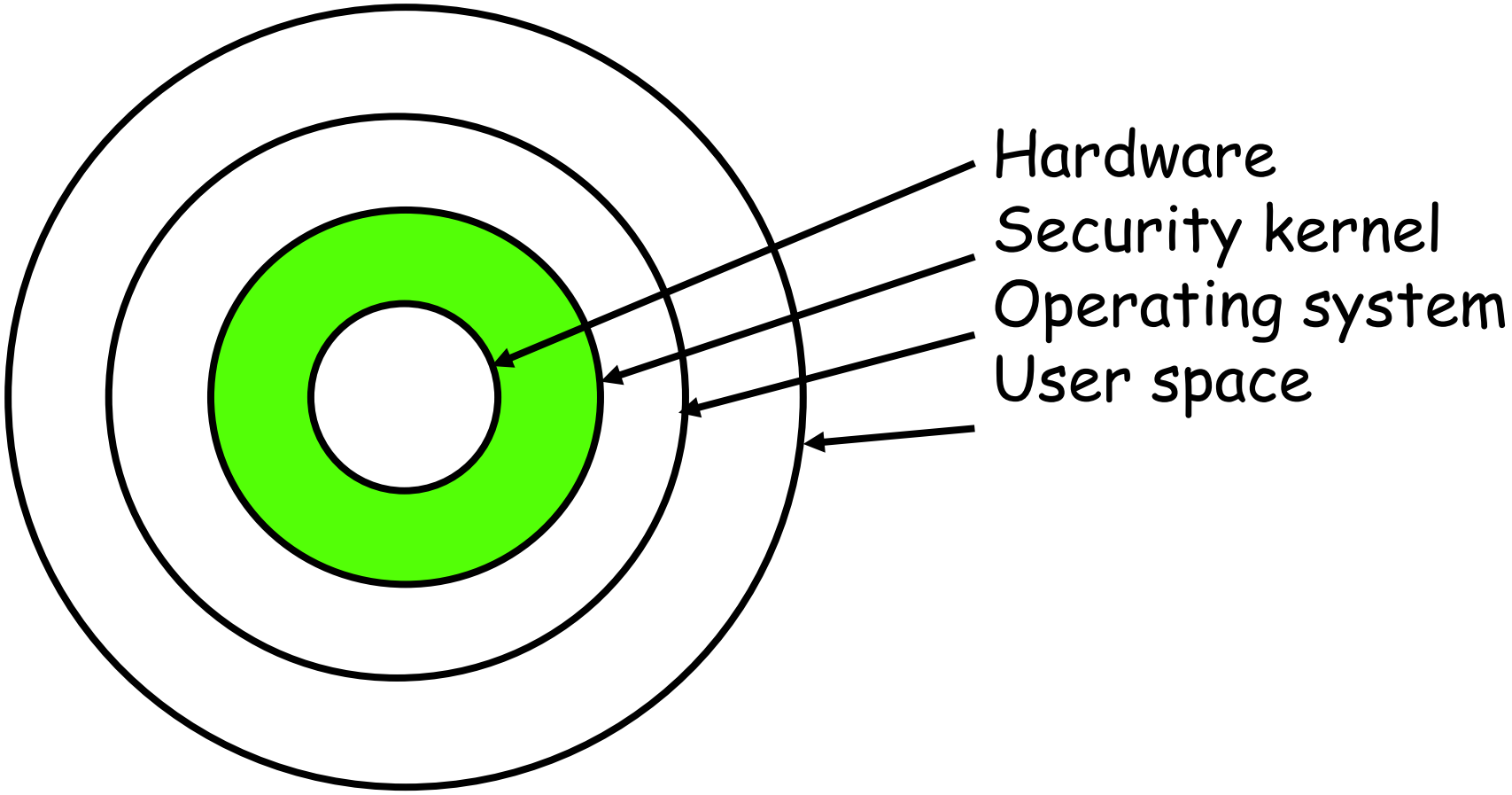
- Security may occur many places within OS
- Ideally, design security kernel first, and build the OS around it
  - *Reality is usually the other way around*
- Example of a trusted OS: **SCOMP**
  - *Developed by Honeywell*
  - *Less than 10,000 LOC in SCOMP security kernel*
  - *Win XP has 40,000,000 lines of code!*

Poor TCB Design



Problem: No clear security layer

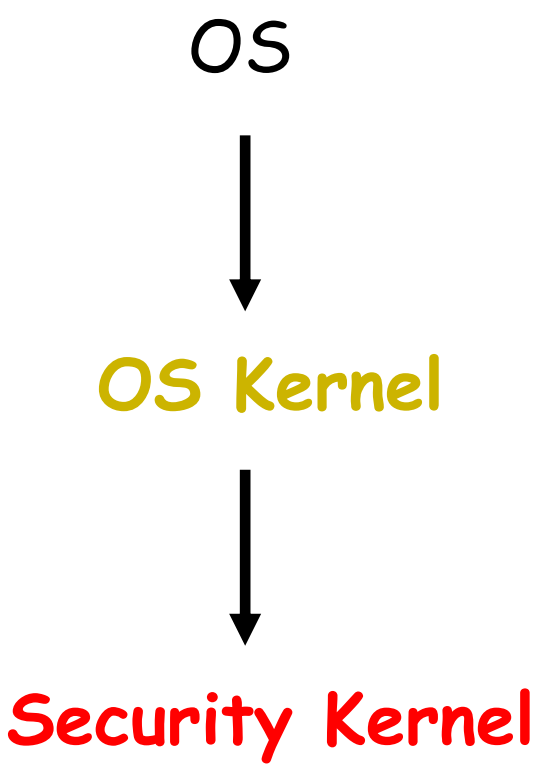
Better TCB Design



Security kernel is the security layer

Trusted OS Summary

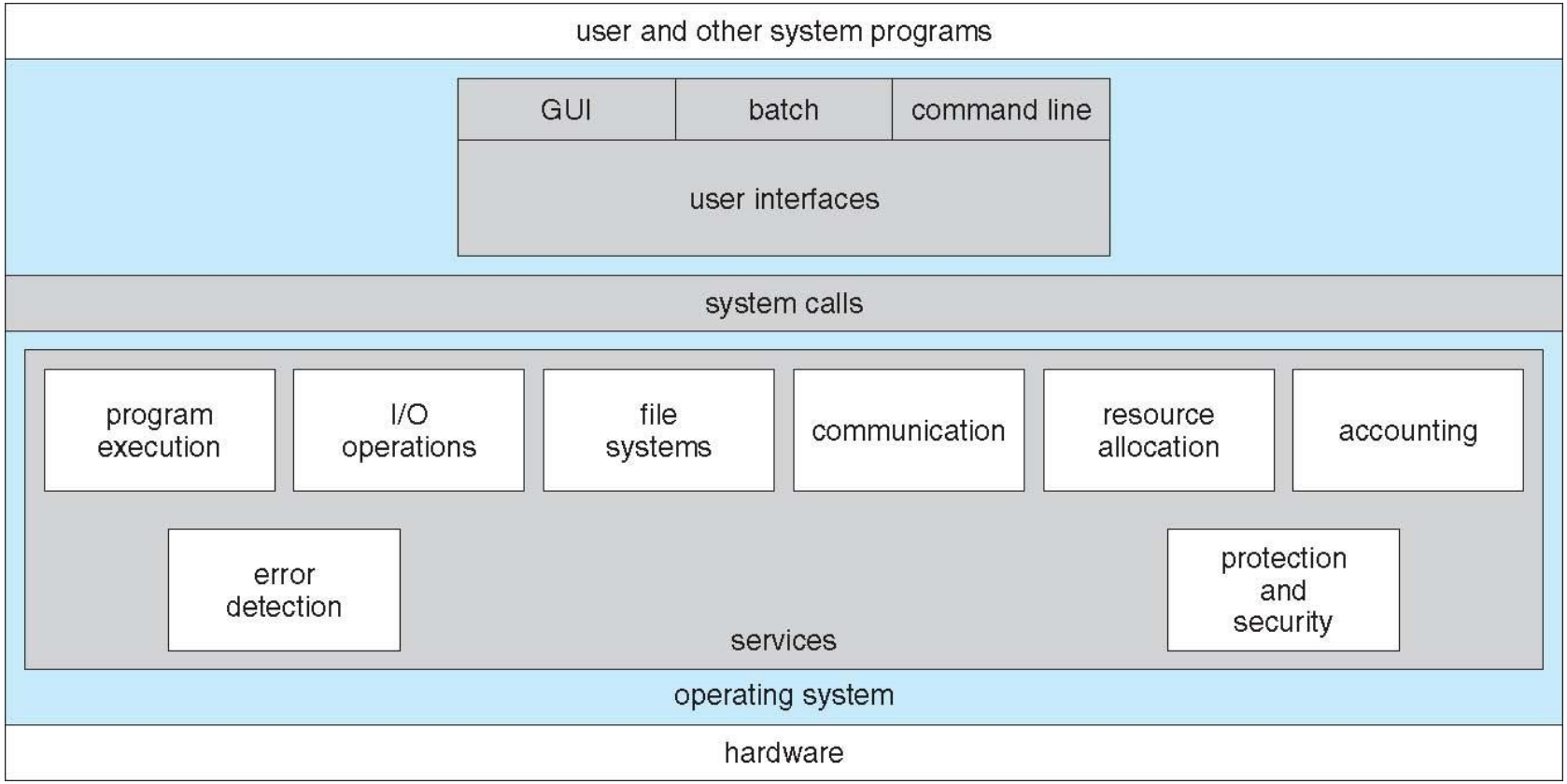
- Trust implies reliance
- TCB (trusted computing base) is everything in OS we rely on for security
- If everything outside TCB is subverted, we still have trusted system
- If TCB subverted, security is broken



Operating System Services

- One set of operating-system services provides functions that are helpful to the user:
  - *User interface - Almost all operating systems have a user interface (UI)*
    - Varies between Command-Line (CLI), Graphics User Interface (GUI), Batch
  - *Program execution - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)*
  - *I/O operations - A running program may require I/O, which may involve a file or an I/O device*
  - *File-system manipulation - The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.*

A View of Operating System Services



Operating System Services (Cont)

- One set of operating-system services provides functions that are helpful to the user (Cont):
  - *Communications – Processes may exchange information, on the same computer or between computers over a network*
    - Communications may be via shared memory or through message passing (packets moved by the OS)
  - *Error detection – OS needs to be constantly aware of possible errors*
    - May occur in the CPU and memory hardware, in I/O devices, in user program
    - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
    - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

Operating System Services (Cont)

- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
  - ***Resource allocation*** *- When multiple users or multiple jobs running concurrently, resources must be allocated to each of them*
    - Many types of resources - Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code
  - ***Accounting*** *- To keep track of which users use how much and what kinds of computer resources*
  - ***Protection and security*** *- The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other*
    - **Protection** involves ensuring that all access to system resources is controlled
    - **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts
    - If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

System Boot

- When power initialized on system, execution starts at a fixed memory location
  - *Firmware ROM used to hold initial boot code*
- Operating system must be made available to hardware so hardware can start it
  - *Small piece of code – **bootstrap loader**, stored in **ROM** or **EEPROM** locates the kernel, loads it into memory, and starts it*
  - *Sometimes two-step process where **boot block** at fixed location loaded by ROM code, which loads bootstrap loader from disk*
- Common bootstrap loader, **GRUB**, allows selection of kernel from multiple disks, versions, kernel options
- Kernel loads and system is then **running**

Starting the Computer

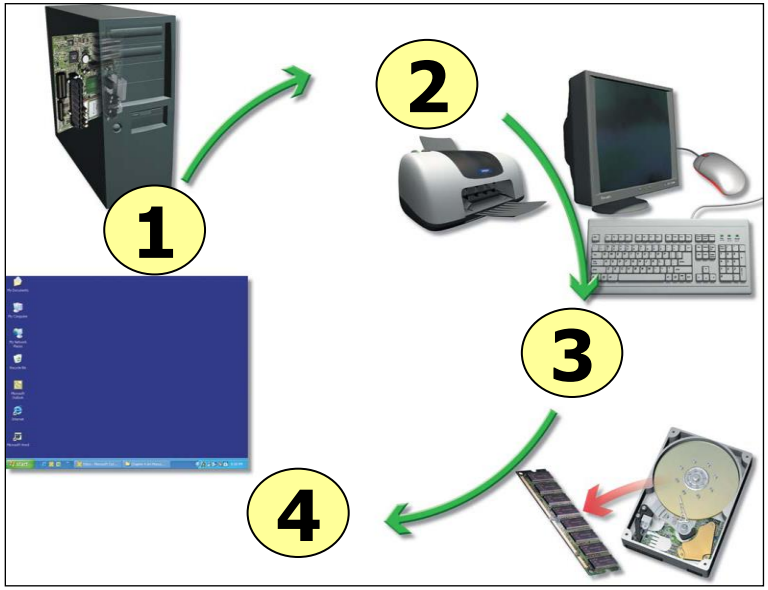
The Boot Process

Step 1: The basic input/output system (BIOS) is activated.

Step 2: A Power-on self-test (POST) checks attached hardware.

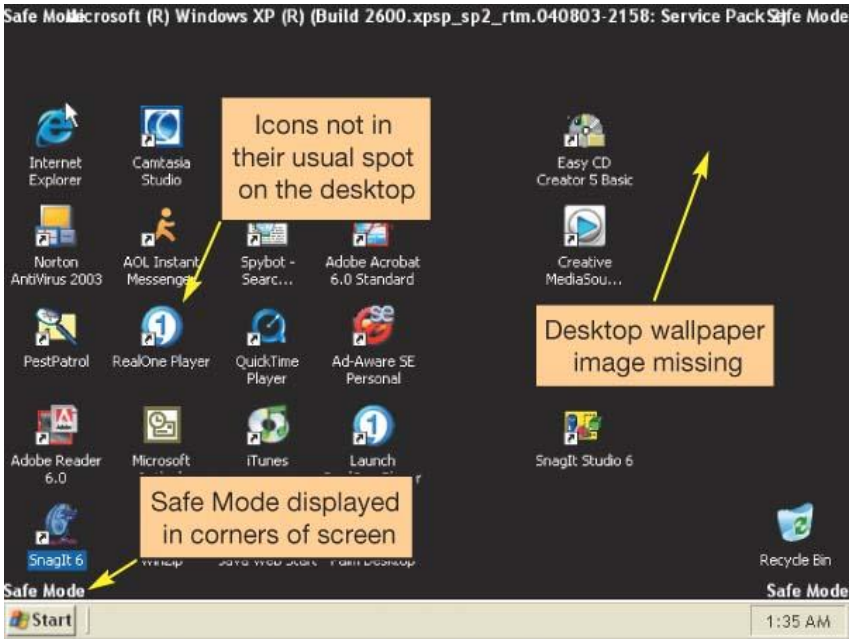
Step 3: The operating system loads into memory.

Step 4: Configuration and customization settings are checked.



Handling Errors in the Boot Process

- Non-system disk or disk error
  - *Remove the floppy from the drive and press any key*
- POST errors
  - *Single beep: Everything is loading properly*
  - *Series of beeps: Hardware problem*
- Safe mode
  - *Windows does not boot properly*
  - *Uninstall any new devices or software*

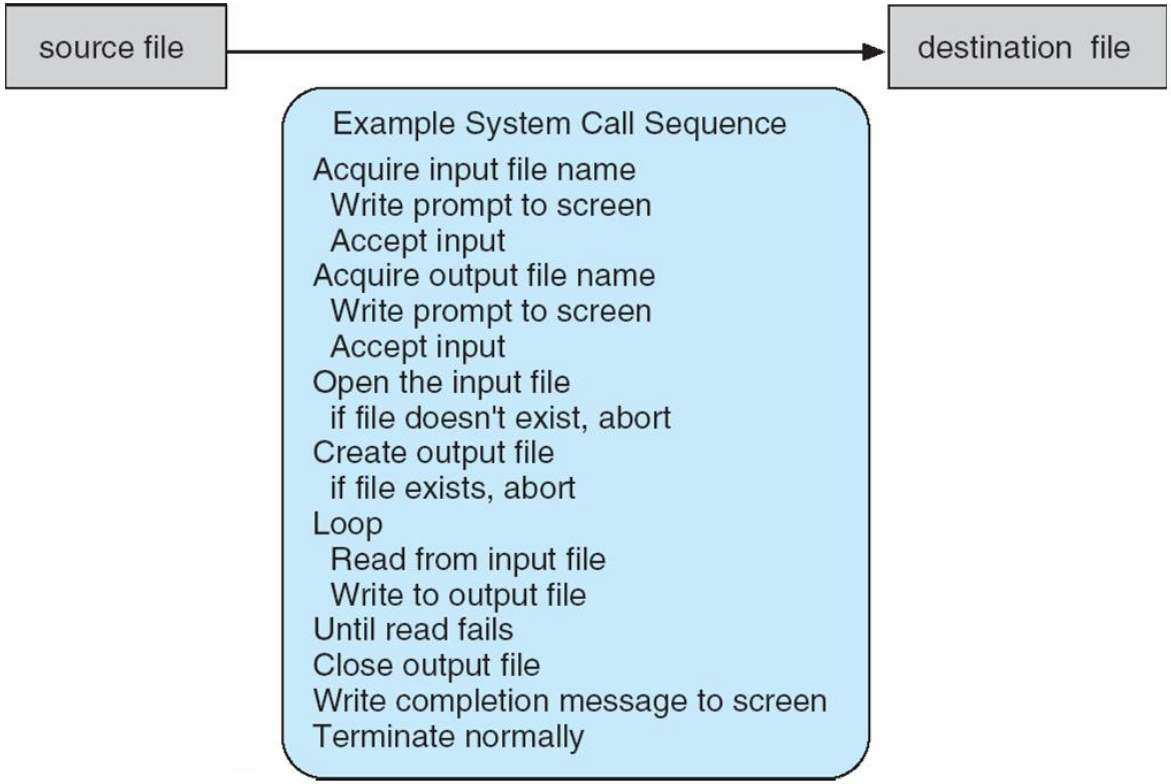


System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use
- Three most common APIs are: Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)
- Why use APIs rather than system calls?

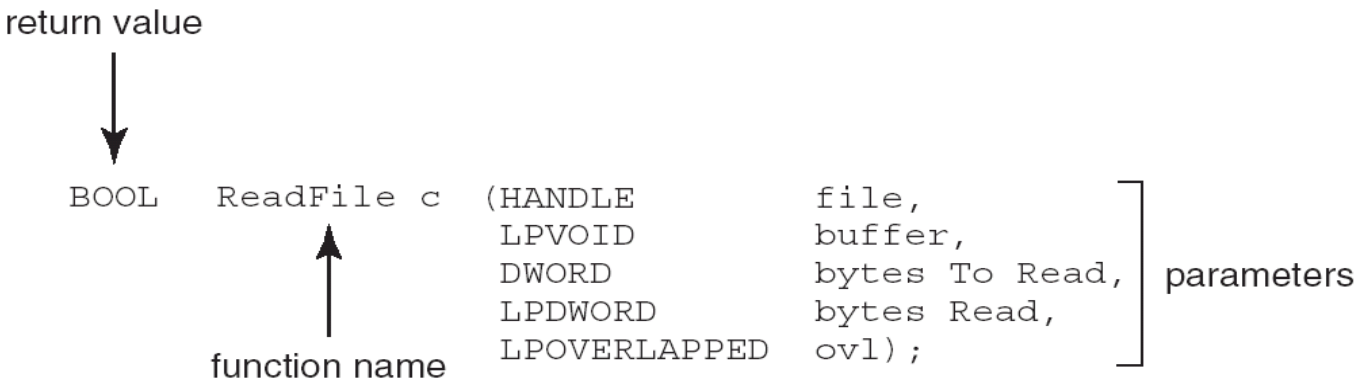
Example of System Calls

- System call sequence to copy the contents of one file to another file



Example of Standard API

- Consider the ReadFile() function in the
- Win32 API—a function for reading from a file

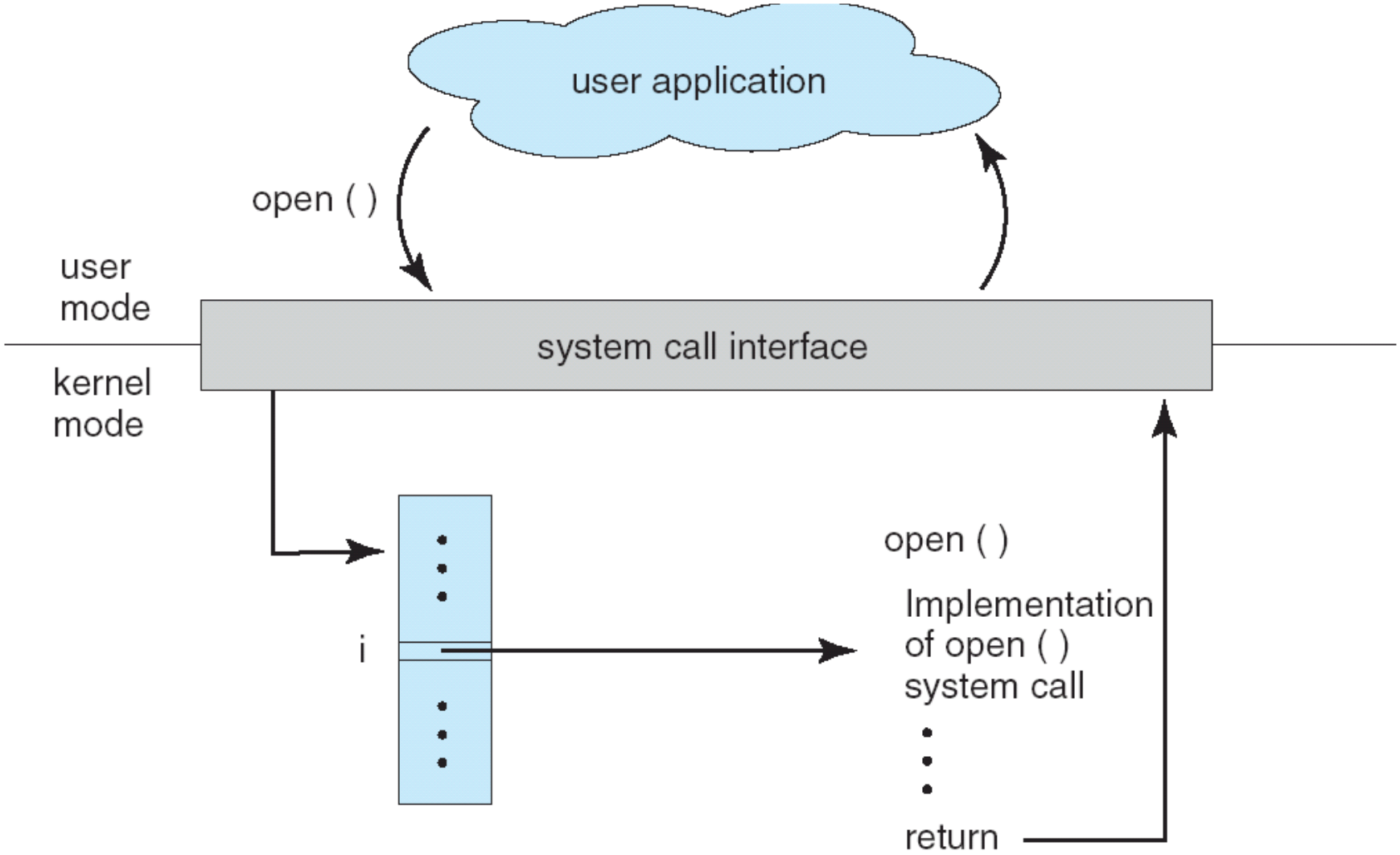


- A description of the parameters passed to ReadFile()
  - *HANDLE file—the file to be read*
  - *LPVOID buffer—a buffer where the data will be read into and written from*
  - *DWORD bytesToRead—the number of bytes to be read into the buffer*
  - *LPDWORD bytesRead—the number of bytes read during the last read*
  - *LPOVERLAPPED ovl—indicates if overlapped I/O is being used*

System Call Implementation

- Typically, a number associated with each system call
  - *System-call interface maintains a table indexed according to these numbers*
- The system call interface invokes intended system call in OS kernel and returns status of the system call and any return values
- The caller need know nothing about how the system call is implemented
  - *Just needs to obey API and understand what OS will do as a result call*
  - *Most details of OS interface hidden from programmer by API*
    - Managed by run-time support library (set of functions built into libraries included with compiler)

API – System Call – OS Relationship



Types of System Calls

System calls can be grouped roughly into six major categories:

- 1- Process control
  - *create process, terminate process*
  - *end, abort*
  - *load, execute*
  - *get process attributes, set process attributes*
  - *wait for time*
  - *wait event, signal event*
  - *allocate and free memory*
  - *Dump memory if error*
  - ***Debugger*** *for determining* ***bugs, single step*** *execution*
  - ***Locks*** *for managing access to shared data between processes*

Types of System Calls (Contd.)

- 2- File management
  - *create file, delete file*
  - *open, close file*
  - *read, write, reposition*
  - *get and set file attributes*
- 3- Device management
  - *request device, release device*
  - *read, write, reposition*
  - *get device attributes, set device attributes*
  - *logically attach or detach devices*
- 4- Information maintenance
  - *get time or date, set time or date*
  - *get system data, set system data*
  - *get and set process, file, or device attributes*

Types of System Calls (Contd.)

- 5- Communications
  - *create, delete communication connection*
  - *send, receive messages if* ***message passing model*** *to* ***host name*** *or* ***process name***
    - From **client** to **server**
  - ***Shared-memory model*** *create and gain access to memory regions*
  - *transfer status information*
  - *attach and detach remote devices*
- 6- Protection
  - *Control access to resources*
  - *Get and set permissions*
  - *Allow and deny user access*

System Programs

- System programs provide a convenient environment for program development and execution. They can be divided into:
  - *File manipulation and modification*
  - *Status information sometimes stored in a File modification*
  - *Programming language support*
  - *Program loading and execution*
  - *Communications*
  - *Background services*
  - *Application programs*
- Most users′ view of the operation system is defined by system programs, not the actual system calls

System Programs (Contd.)

- Provide a convenient environment for program development and execution
  - *Some of them are simply user interfaces to system calls; others are considerably more complex*
- **File management** - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories
- **Status information**
  - *Some ask the system for info - date, time, amount of available memory, disk space, number of users*
  - *Others provide detailed performance, logging, and debugging information*
  - *Typically, these programs format and print the output to the terminal or other output devices*
  - *Some systems implement a **registry** - used to store and retrieve configuration information*

System Programs (Contd.)

- **File modification**
  - *Text editors to create and modify files*
  - *Special commands to search contents of files or perform transformations of the text*
- **Programming-language support** - Compilers, assemblers, debuggers and interpreters sometimes provided
- **Program loading and execution**- Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language
- **Communications** - Provide the mechanism for creating virtual connections among processes, users, and computer systems
  - *Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another*

System Programs (Contd.)

- **Background Services**
  - *Launch at boot time*
    - Some for system startup, then terminate
    - Some from system boot to shutdown
  - *Provide facilities like disk checking, process scheduling, error logging, printing*
  - *Run in user context not kernel context*
  - *Known as **services**, **subsystems**, **daemons***
- **Application programs**
  - *Don't pertain to system*
  - *Run by users*
  - *Not typically considered part of OS*
  - *Launched by command line, mouse click, finger poke*

Process/Task/Application Management

- A process is a program in execution. Program is a *passive entity*, process is an *active entity*.
- Process needs resources to accomplish its task
  - *CPU time*
- Representation of process
  - *Process has one **program counter** specifying location of next instruction to execute*
  - *Data structure (stores information of a process)*
- Many processes may be associated with the same program
- Typically system has many processes
  - *some user processes,*
  - *some operating system processes*
- Life cycle of a process
  - *States*
  - *Arrival, Computation, I/O, I/O completion, termination*

Process/Task/Application Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Process scheduling
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

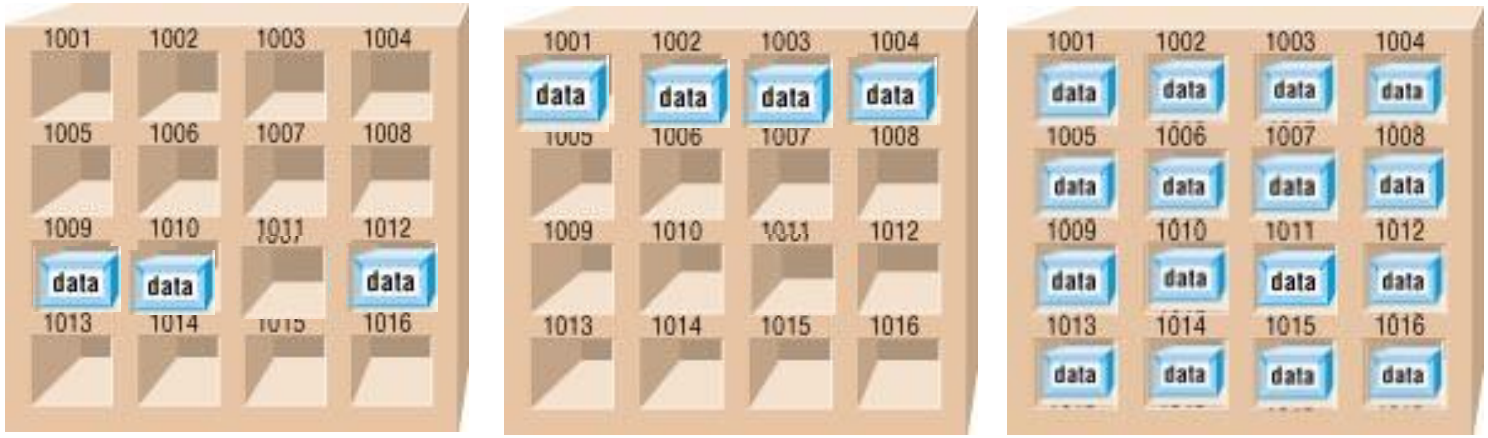
Managing Processor Tasks

- Controls the timing of events the processor works on
  - *Interrupts*
  - *Interrupt handler*
  - *Interrupt table*
  - *Stack*

Managing Memory Tasks

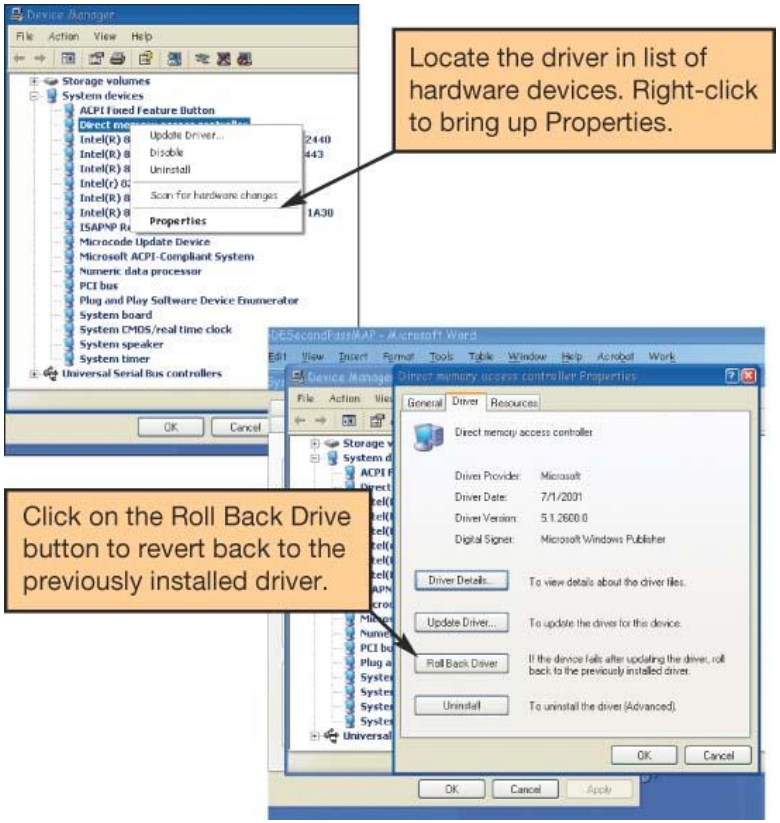
- The operating system allocates space in RAM for instructions and data

RAM



Managing Software Tasks

- Device drivers:
  - *Programs that enable the operating system to communicate with peripheral devices*
  - *Provided by the manufacturer of the device*
- Plug and Play:
  - *Hardware and software standard*
  - *Facilitates the installation of new hardware*

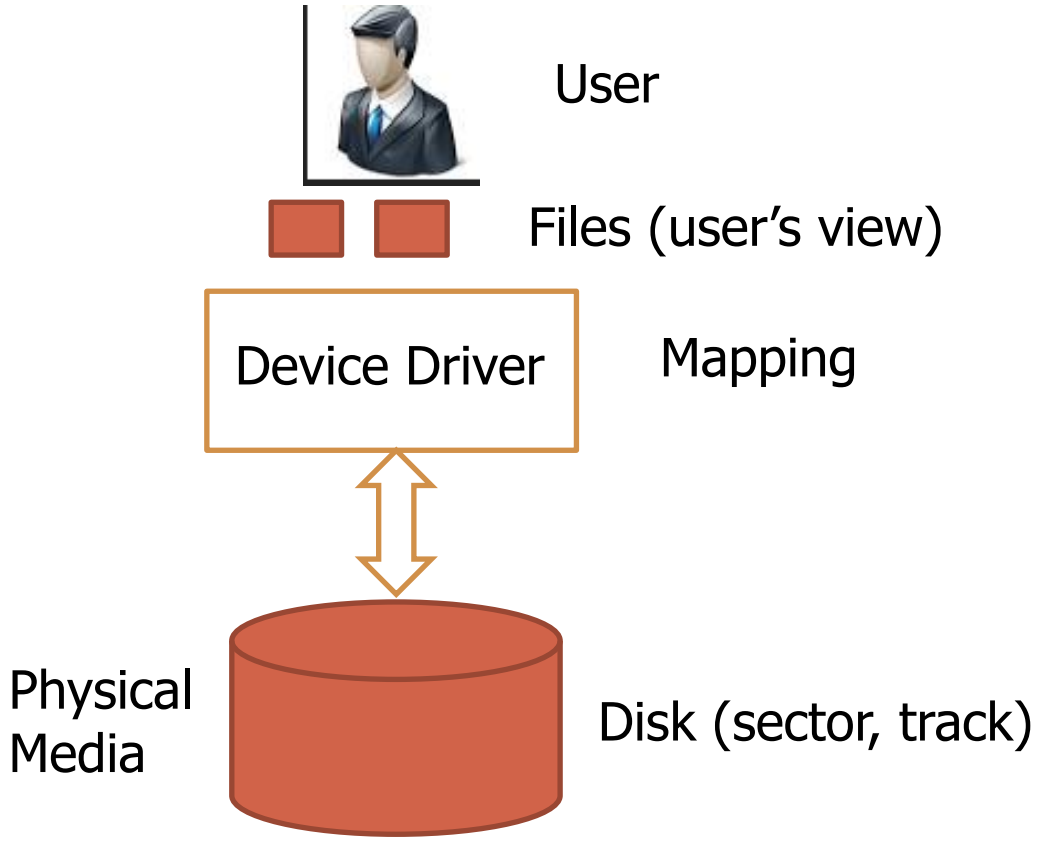


Managing Software Tasks

- Application programming interfaces (APIs):
  - *Blocks of code contained in the operating system*
  - *Coordinates the operating system with software applications*
    - Similar toolbars and menus
  - *Microsoft Direct X*

File Management

- OS provides uniform, logical view of information storage
  - *Abstracts physical properties to logical storage unit* - ***File***
  - *File => Collection of related information defined by the creator*
  - *Each medium is controlled by device (i.e., disk drive, tape drive)*
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)



File Management

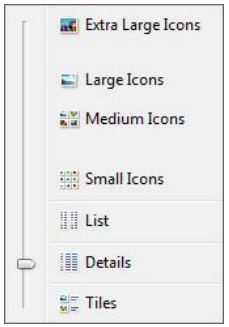
- OS provides uniform, logical view of information storage
  - *Abstracts physical properties to logical storage unit - **file***
  - *Each medium is controlled by device (i.e., disk drive, tape drive)*
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

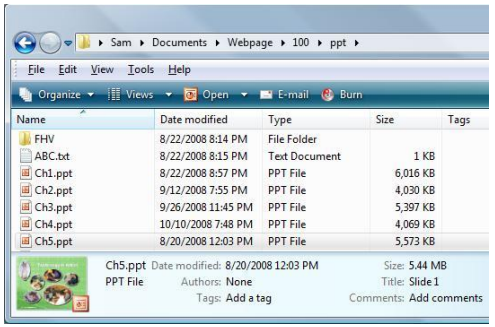
- OS implements the abstract concept of file by managing mass storage media (disk etc) and devices that control them

- Files usually organized into directories
- Access control on most systems to determine who can access what
- File-System management
  - *Creating and deleting files and directories*
  - *Primitives to manipulate files and dirs*
  - *Mapping files onto secondary storage*

File Management

- The operating system provides an organizational structure to the computer's contents
- Hierarchical structure of directories:
  - *Drives*
    - Folders
      - Subfolders
        - Files





- Viewing and Sorting Files and Folders: Windows Explorer and Views
- Naming Files
  - *Name assigned plus filename extension*
  - *only characters not legal in filenames are:*

    \ / : * ? " < > ¦
  - *all others are allowed*
- File Path: Location of the file → Drive, Primary Folder, Subfolders, and File Name.

C:\My Documents\Tech in Action\TIA Pics\dotmatrix.gif

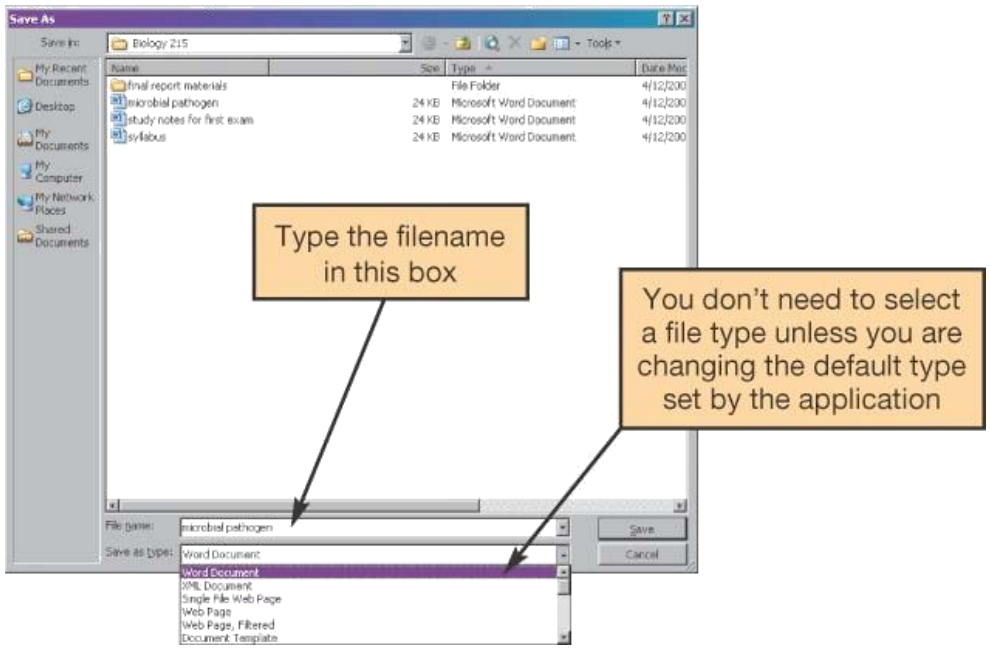
Filename Extensions

- Filename extensions:
  - *Used by programs*

Extension	Type of Document	Application
.doc	Word processing document	Microsoft Word; Corel WordPerfect
.xls	Workbook	Microsoft Excel
.ppt	PowerPoint presentation	Microsoft PowerPoint
.mdb	Database	Microsoft Access
.bmp	Bitmap image	Windows
.zip	Compressed file	WinZip
.pdf	Portable Document Format	Adobe Acrobat
.htm or .html	Web page	Hypertext Markup Language

Working with Files

- File management actions:
  - *Open*
  - *Copy*
  - *Move*
  - *Rename*
  - *Delete*
- Recycle bin



Saving files

File Commands

- Caution: the above commands do not prompt for confirmation
  - *easy to overwrite/delete a file; this setting can be overridden (how?)*

- *Exercise* : Given several albums of `.mp3` files all in one folder, move them into separate folders by artist.
- *Exercise* : Modify `a .java file` to make it seem as though you finished writing it on March 15 at 4:56am.

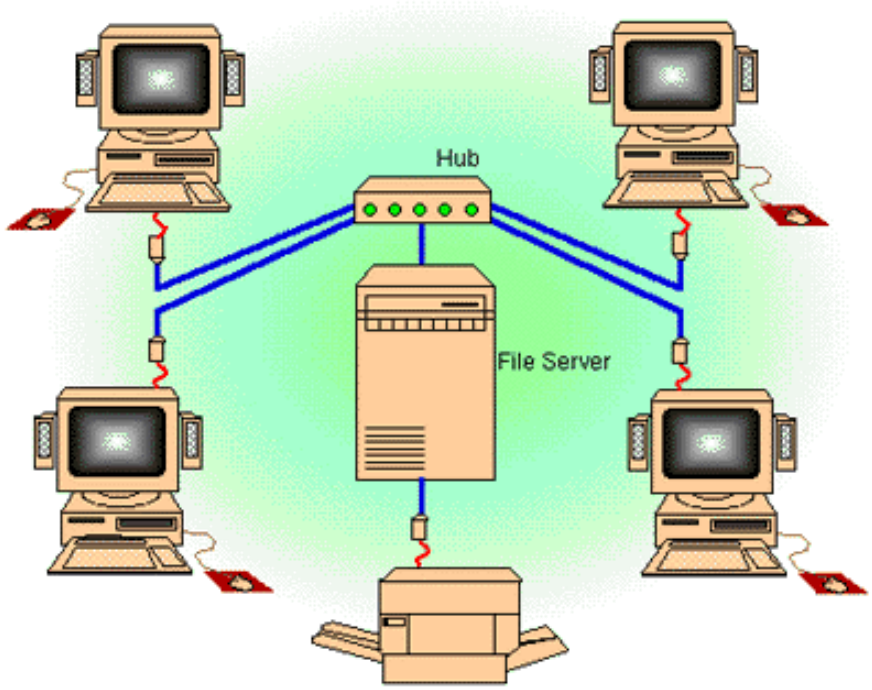
command	description
`cp`	copy a file
`mv`	move or rename a file
`rm`	delete a file
`touch`	create a new empty file, or update its last-modified time stamp

Disk Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a "long" period of time
  - *Most of the programs are stored on disk*
- Proper management is of central importance
- Entire speed of computer operation depends on disk subsystem and its algorithms
- OS activities
  - *Storage allocation (logical blocks)*
  - *Free-space management*
  - *Disk scheduling*

Multiuser Operating Systems

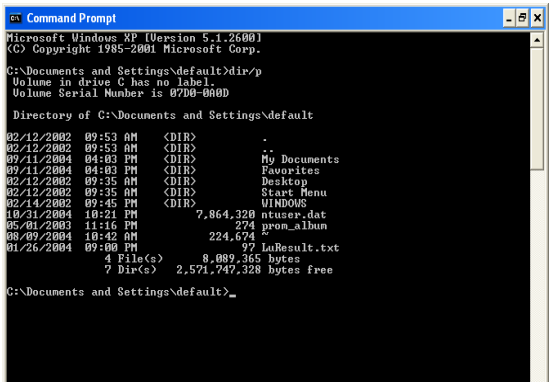
- Known as network operating systems
- Allow access to the computer system by more than one user
- Manage user requests
- Systems include:
  - *UNIX*
  - *Linux*
  - *Novell Netware*
  - *Windows Server 2003*



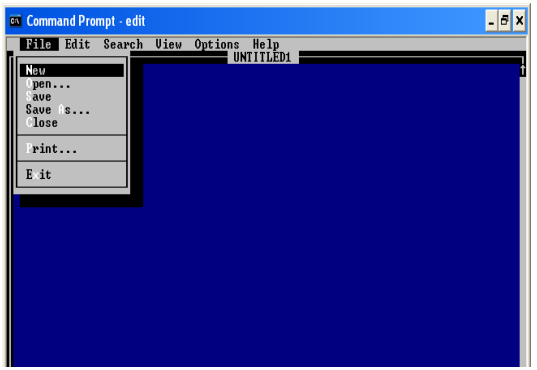
The User Interface

- Enables you to interact with the computer
- Types of interfaces:
  - *Command-driven interface*
  - *Menu-driven interface*
  - *Graphical user interface*

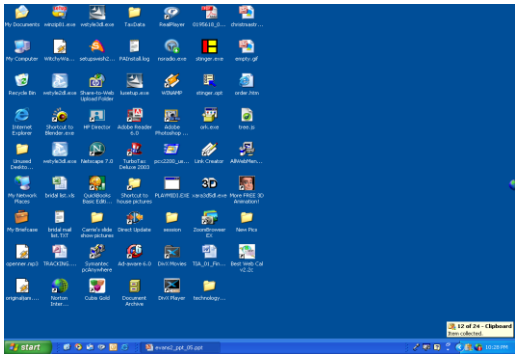
Command-driven



Menu-driven



Graphical



Questions?