

If a computer program is run by an unauthorized user, then he may cause very damage to computer or data stored in it. So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc.

We're going to discuss following topics in this chapter.

- Authentication
- One Time passwords
- Program Threats
- System Threats
- Computer Security Classifications

Authentication

Authentication refers to identifying each user of the system and associating the executing programs with those users. It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic.

Operating Systems generally identifies/authenticates users using following three ways –

- **Username / Password** – User need to enter a registered username and password with Operating system to login into the system.
- **User card/key** – User need to punch card in card slot, or enter key generated by key generator in option provided by operating system to login into the system.
- **User attribute - fingerprint/ eye retina pattern/ signature** – User need to pass his/her attribute via designated input device used by operating system to login into the system.

One Time passwords

One-time passwords provide additional security along with normal authentication. In One-Time Password system, a unique password is required every time user tries to login into the system. Once a one-time password is used, then it cannot be used again.

One-time password are implemented in various ways.

- **Random numbers** – Users are provided cards having numbers printed along with corresponding alphabets. System asks for numbers corresponding to few alphabets randomly chosen.
- **Secret key** – User are provided a hardware device which can create a secret id mapped with user id. System asks for such secret id which is to be generated every time prior to login.
- **Network password** – Some commercial applications send one-time passwords to user on registered mobile/ email which is required to be entered prior to login.

Program Threats

If a user program made these process do malicious tasks, then it is known as **Program Threats**. One of the common example of program threat is a program installed in a computer which can store and send user credentials via network to some hacker. Following is the list of some well-known program threats.

- **Trojan Horse** – Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
- **Trap Door** – If a program which is designed to work as required, have a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.
- **Logic Bomb** – Logic bomb is a situation when a program misbehaves only when certain conditions met otherwise it works as a genuine program. It is harder to detect.
- **Virus** – Virus as name suggest can replicate themselves on computer system. They are highly dangerous and can modify/delete user files, crash systems. A virus is generally a small code in a program. As user accesses the program, the virus starts getting embedded in other files/ programs and can make system unusable for user

System Threats

System threats refers to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as program attack. System threats creates such an environment that operating system user files are misused.

Following is the list of some well-known system threats.

- **Worm** – Worm is a process which can choked down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worms processes can even shut down an entire network.
- **Port Scanning** – Port scanning is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.
- **Denial of Service** – Denial of service attacks normally prevents user to make legitimate use of the system. For example, a user may not be able to use internet if denial of service attacks browser's content settings.

Computer Security Classifications

As per the U.S. Department of Defense Trusted Computer System's Evaluation Criteria there are four security classifications in computer systems: A, B, C, and D.

This is widely used specifications to determine and model the security of systems and of security solutions. Following is the brief description of each classification.

Classification Type & Description

Type A

Highest Level. Uses formal design specifications and verification techniques. Grants a high degree of assurance of process security.

Type B

Provides mandatory protection system. Have all the properties of a class C2 system. Attaches a sensitivity label to each object. It is of three types.

- B1** – Maintains the security label of each object in the system. Label is used for making decisions to access control.
- B2** – Extends the sensitivity labels to each system resource, such as storage objects, supports covert channels and auditing of events.
- B3** – Allows creating lists or user groups for access-control to grant access or revoke access to a given named object.

Type C

- Provides protection and user accountability using audit capabilities. It is of two types.
- **C1** – Incorporates controls so that users can protect their private information and keep other users from accidentally reading / deleting their data. UNIX versions are mostly C1 class.
- **C2** – Adds an individual-level access control to the capabilities of a C1 level system.

Type D

- Lowest level. Minimum protection. MS-DOS, Window fall in this category.

Malicious software or Malwares

- Malicious software (malware) is any software that gives partial to full control of the system to the attacker/malware creator.
- Any **software** that brings harm to a computer system.
- **Malware**” is short for “malicious software” - **computer** programs designed to damage **computers** without the users consent.
- “**Malware**” is the general term covering all the different types of threats to your **computer** safety
-

To understand how malware works, we should first see the anatomy of a malware attack,
which is separated in five steps as shown below –

- Entry point
- Distribution
- Exploit
- Infection
- Execution

Entry Point

- A malware can enter into the system in many ways –
- The user visits his favorite website that has been infected recently. This can be an entry point for a malware.
- If a user clicks on a URL that has come in an email, it will hijack that browser.
- Malware can also enter through any infected external media such as a USB or an external hard drive.

Distribution

- The malware a process that redirects the traffic to an exploit server which checks the OS and applications such as the browser, Java, Flash player, etc.

Exploit

- In this phase, the **exploit** will try to execute based on the OS and will find a way to escalate the privilege.

Infection

- Now, the exploit that was successfully installed will upload a payload to maintain access and to manage the victim like remote access, file upload/download, etc.

Execution

- In this phase, the hacker who manages the Malware will start to steal your data, encrypt your files, etc.

They come from different functions and behave differently under various situations. Some of the most infamous and dangerous types of malwares are given below:

01.Virus

02.worm

03.Adware

04.Spyware

05.Trojan

06.Rootkits

07.Botnets

08.Ransom Ware

Various forms of malware are listed below –

01.Virus – A virus is a program that creates copies of itself and inserts these copies into other computer programs, data files, or into the boot sector of the hard-disk.

Upon successful replication, viruses cause harmful activity on infected hosts such as stealing hard-disk space or CPU time.

Types of Viruses

- can classify on basis of how they attack
 - parasitic virus
 - memory-resident virus
 - boot sector virus
 - stealth
 - polymorphic virus
 - macro virus
 - Email virus

boot sector virus

- A **boot sector virus** is a type of **virus** that infects the **boot sector** of floppy disks or the Master **Boot** Record (MBR) of hard disks (some infect the **boot sector** of the hard disk instead of the MBR).



Macro Virus

- **macro code** attached to some **data file**
- interpreted by program using file
 - eg Word/Excel macros
 - esp. using auto command & command macros
- code is now platform independent
- is a major source of new viral infections
- blurs distinction between data and program files making task of detection much harder
- classic trade-off: "ease of use" vs "security"

Macro Virus

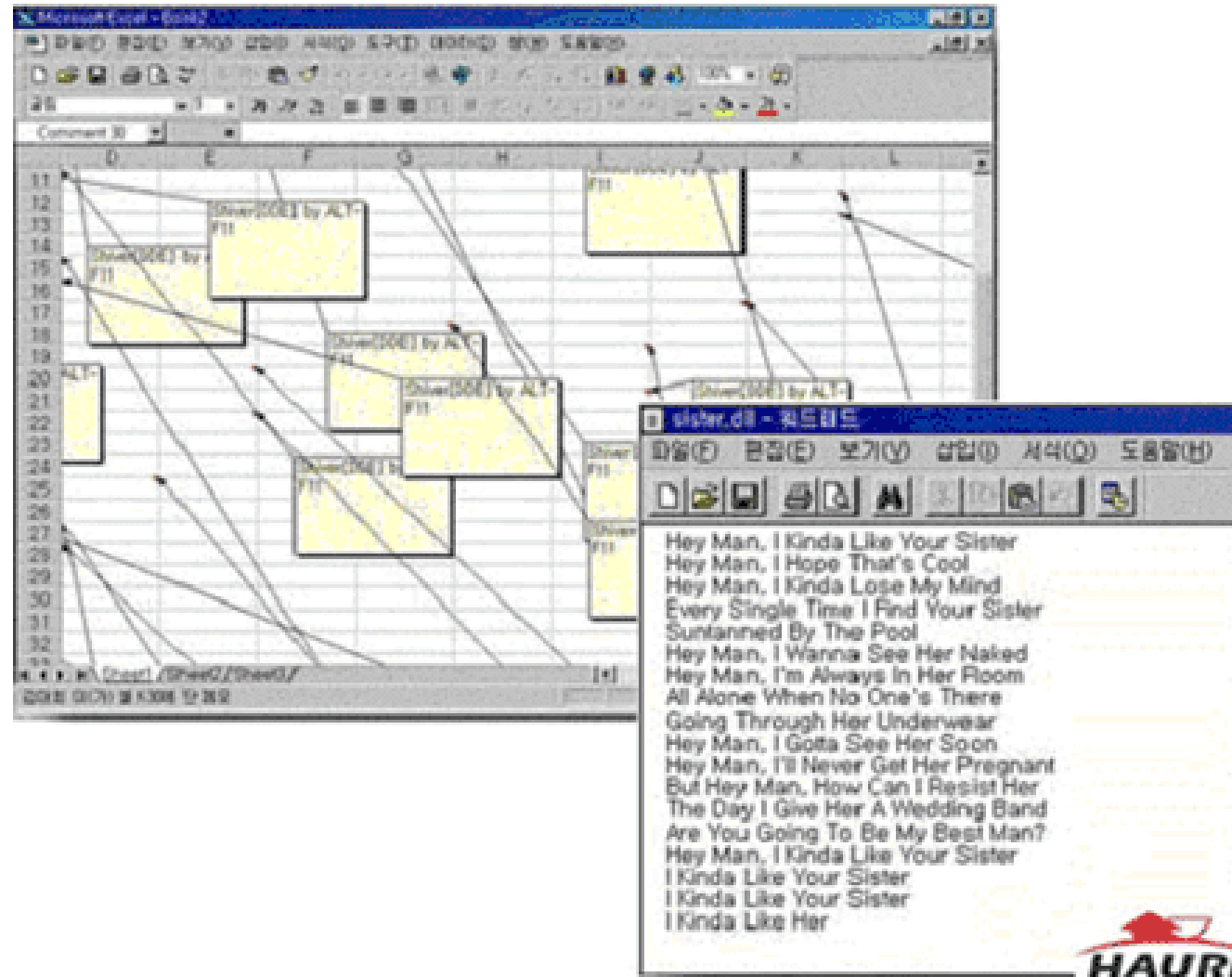


Figure 4. A screen shot of 097M.Shiver.C virus that infected MS Word 97 and Excel 97

Email Virus

- spread using email with attachment containing a macro virus
 - cf Melissa
- triggered when user opens attachment
- or worse even when mail viewed by using scripting features in mail agent
- usually targeted at Microsoft Outlook mail agent & Word/Excel documents

E mail Virus

WOW

Inbox | X



Devon

[show details](#) Sep 20

[Reply](#)



<http://rapidshare.com/files/282512175/install.exe>

Probably a
Virus

[Reply](#)

[Reply to all](#)

[Forward](#)



• Evan Wondrasek to Devon

[show details](#) Sep 20

[Reply](#)



Hi Devon,

I wanted to verify that you intended to send this email, as it looks like a potential virus. If you did not intend to send this email, I recommend sending out a notification email to its recipients immediately.

Thanks,

Evan

polymorphic virus

A **polymorphic virus** is a complicated computer **virus** that affects data types and functions.

It is a self-encrypted **virus** designed to avoid detection by a scanner. Upon infection, the **polymorphic virus** duplicates itself by creating usable, modified, copies of itself.

Virus Countermeasures

- viral attacks exploit lack of integrity control on systems
- to defend need to add such controls
- typically by one or more of:
 - **prevention** - block virus infection mechanism
 - **detection** - of viruses in infected system
 - **reaction** - restoring system to clean state

02.Trojan – Trojan is a non-self-replicating type of malware that contains malicious code, which upon execution results in loss or theft of data or possible system harm.

03.Adware – Adware, also known as freeware or pitch ware, is a free computer software that contains commercial advertisements of games, desktop toolbars and utilities. It is a web-based application and it collects web browser data to target advertisements, especially pop-ups.

04. Spyware – Spyware is software that anonymously monitors users which enables a hacker to obtain sensitive information from the user's computer. Spyware exploits users and application vulnerabilities that is quite often attached to free online software downloads or to links that are clicked by users.

05. Rootkit – A rootkit is a software used by a hacker to gain admin level access to a computer/network which is installed through a stolen password or by exploiting a system vulnerability without the victim's knowledge.

06.Botnets

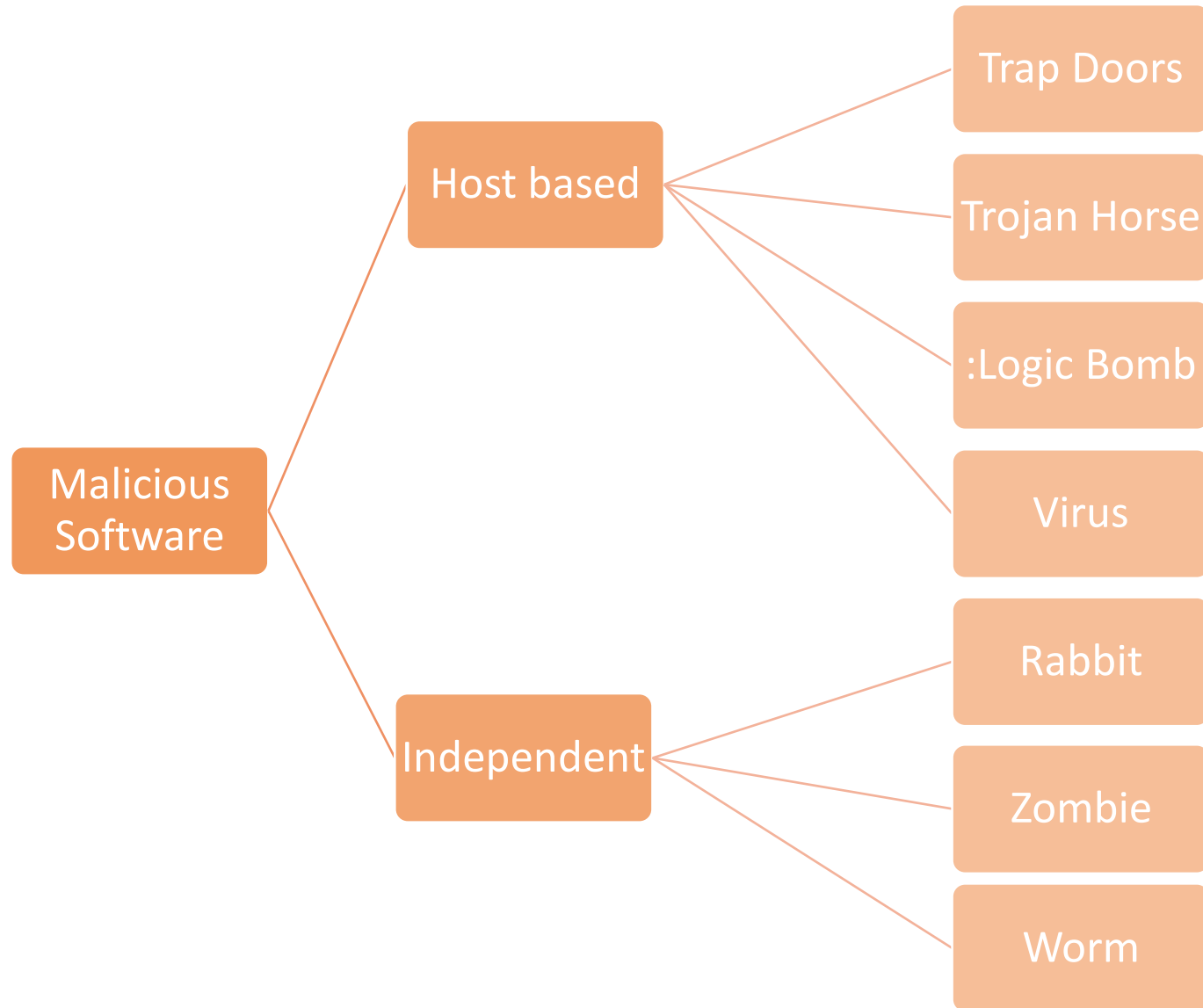
Botnet is a software installed on a computer that is connected through the internet and it can help one communicate with the other same type of programs, so that some actions can be performed. They can be same as keeping control of some IRC, which are Internet Related Charts.

07. Ransom Ware

- Ransom ware is a software that encrypts files, which are on the hard drives. Some of them can even end up with simply showing some message about payment of money to the person, who has implemented this program.

08. Worm – A worm is a type of malware which leaves a copy of itself in the memory of each computer in its path

Malicious Software Classification



Trapdoors

- secret entry point into a program
- allows those who know access bypassing usual security procedures
- have been commonly used by developers
- a threat when left in production programs allowing exploited by attackers
- very hard to block in O/S
- requires good s/w development & update

Trojan Horse

program with hidden side-effects which is usually superficially attractive.

eg game, s/w upgrade etc

when run performs some additional tasks allows attacker to indirectly gain access they do not have directly often used to propagate a virus/worm or install a backdoor or simply to destroy data



Logic Bomb

- one of oldest types of malicious software
- code embedded in legitimate program
- activated when specified conditions met
 - eg presence/absence of some file
 - particular date/time
 - particular user
- when triggered typically damage system
 - modify/delete files/disks



Viruses

- a piece of self-replicating code attached to some other code
 - cf biological virus
- both propagates itself & carries a payload
 - carries code to make copies of itself
 - as well as code to perform some covert task

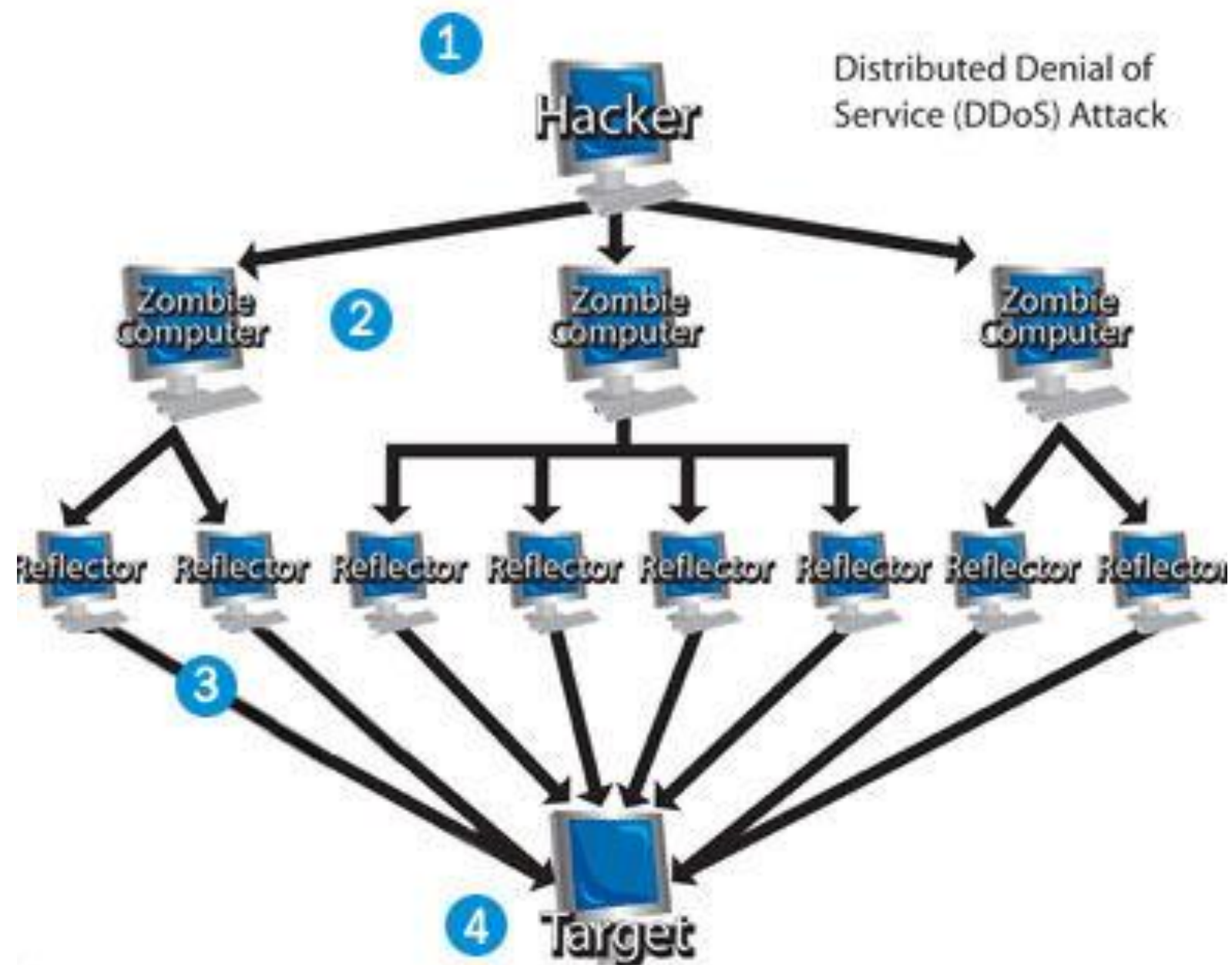
Independent

Rabbit

- Term used to describe malware that multiplies rapidly
 - also called bacteria
- two kinds of rabbits
 1. program which tries to consume all of some system resource, like disk space
 2. standalone program which replicates itself across a network from machine to machine, but deletes the original copy of itself after replication.

Zombie

- program which secretly takes over another networked computer
- then uses it to indirectly launch attacks
- often used to launch distributed denial of service (DDoS) attacks
- exploits known flaws in network systems



DDoS Countermeasures

➤ three broad lines of defense:

1. attack prevention (before)
2. attack detection & filtering (during)
3. attack source trace back (after)

➤ huge range of attack possibilities

➤ hence evolving countermeasures

Worms

- replicating but not infecting program
- typically spreads over a network
 - cf Morris Internet Worm in 1988
 - led to creation of CERTs
- using users distributed privileges or by exploiting system vulnerabilities
- widely used by hackers to create **zombie PC's**, subsequently used for further attacks, esp DoS
- major issue is lack of security of permanently connected systems, esp PC's

Worm Operation

- worm phases like those of viruses:
 - dormant
 - propagation
 - search for other systems to infect
 - establish connection to target remote system
 - replicate self onto remote system
 - triggering
 - execution

Morris Worm

- best known classic worm
- released by Robert Morris in 1988
- targeted Unix systems
- using several propagation techniques
 - simple password cracking of local pw file
 - exploit bug in finger
 - exploit debug trapdoor in send mail daemon
- if any attack succeeds then replicated self

Recent Worm Attacks

- new spate of attacks from mid-2001
- **Code Red**
 - exploited bug in MS IIS to penetrate & spread
 - probes random IPs for systems running IIS
 - had trigger time for denial-of-service attack
 - 2nd wave infected 360000 servers in 14 hours
- **Code Red 2**
 - had backdoor installed to allow remote control
- **Nimda**
 - used multiple infection mechanisms
 - email, shares, web client, IIS, Code Red 2 backdoor

Spyware

- Software which collects information from a computer without owners knowledge and transmits it to someone unauthorised to use them.
- Information may include:
 - User names and Passwords
 - E mail addresses
 - Bank account of credit card numbers
 - Software license keys

Adware

- more marketing-focused, and may pop up advertisements or redirect a user's web browser to certain web sites in the hopes of making a sale.
- Some attempt to target the advertisement to fit context of what user is doing
Ex. a search for "Calgary" may result in an unsolicited pop-up advertisement for "books about Calgary."

Virus Operation

- virus phases:
 - dormant – waiting on trigger event
 - propagation – replicating to programs/disks
 - triggering – by event to execute payload
 - execution – of payload
- details usually machine/OS specific
 - exploiting features/weaknesses

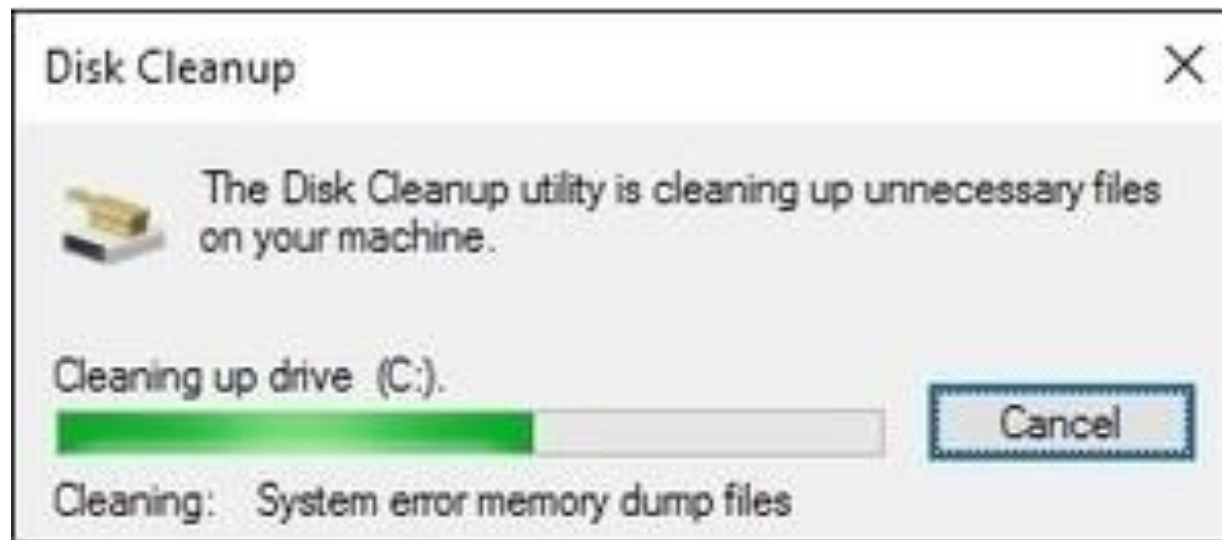
Malware Removal Process

The following measures can be taken to avoid presence of malware in a system –

- Ensure the operating system and applications are up to date with patches/updates.
- Never open strange e-mails, especially ones with attachments.
- When you download from the internet, always check what you install. Do not simply click OK to dismiss pop-up windows. Verify the publisher before you install application.
- Install anti-virus software.
- Ensure you scan and update the antivirus programs regularly.
- Install firewall.
- Always enable and use security features provided by browsers and applications.

Delete Temporary files

- Delete your temporary files. Doing this will speed up the virus scanning, free up disk space and even get rid of some malware. To use the **Disk Cleanup Utility**, included with Windows 10 just type **Disk Cleanup** in the search bar or after pressing the Start button and select the tool that appears – Disk Cleanup.



Anti-Malware Software

The following software help remove the malwares from a system –

- Microsoft Security Essentials
- Microsoft Windows Defender
- AVG Internet Security
- Spybot - Search & Destroy
- Avast! Home Edition for personal use
- Panda Internet Security
- MacScan for Mac OS and Mac OS X

Security in trusted Operating Systems

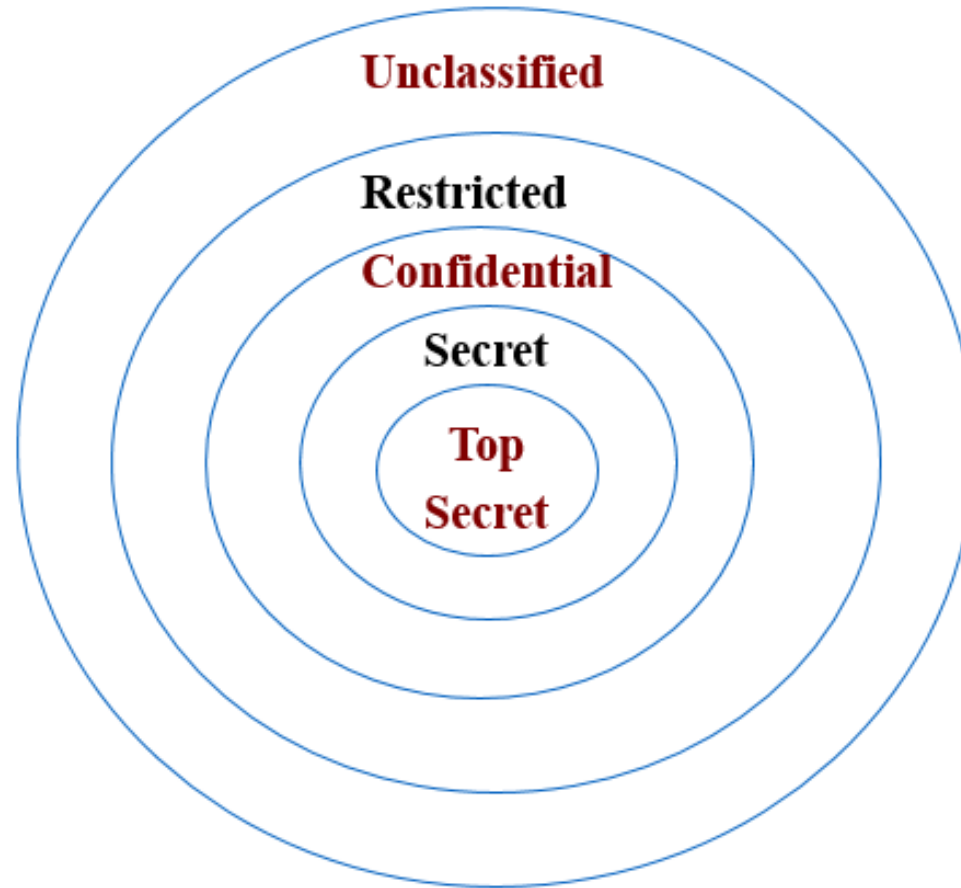
Trusted Operating System

- **Trusted Operating System** (TOS) generally refers to an **operating system** that provides sufficient support for multilevel security and evidence of correctness to meet a particular set of requirements.

What is a Trusted System?

- Functional correctness
- Enforcement of integrity
- Limited privilege
- Appropriate confidence level

Military Security policy



Trusted OS Design

- OS is a complex system
 - difficult to design
 - Adding the responsibility of security enforcement makes it even more difficult
- Clear mapping from security requirements to the design
- Design must be checked using formal reviews or simulation
- Requirements → design → testing

Security Design Principles

- Least privilege
 - users, programs, fewest privilege possible
- Economy of mechanism
 - small, simple, straight forward
- Open design
 - extensive public scrutiny
- Complete mediation
 - every attempt must be checked

Security Design Principles....

- Permission based
 - denial of access is the default
- Separation of privilege
 - more than one condition
- Least common mechanism
 - the risk of sharing
- Ease of use
 - unlikely to be avoided

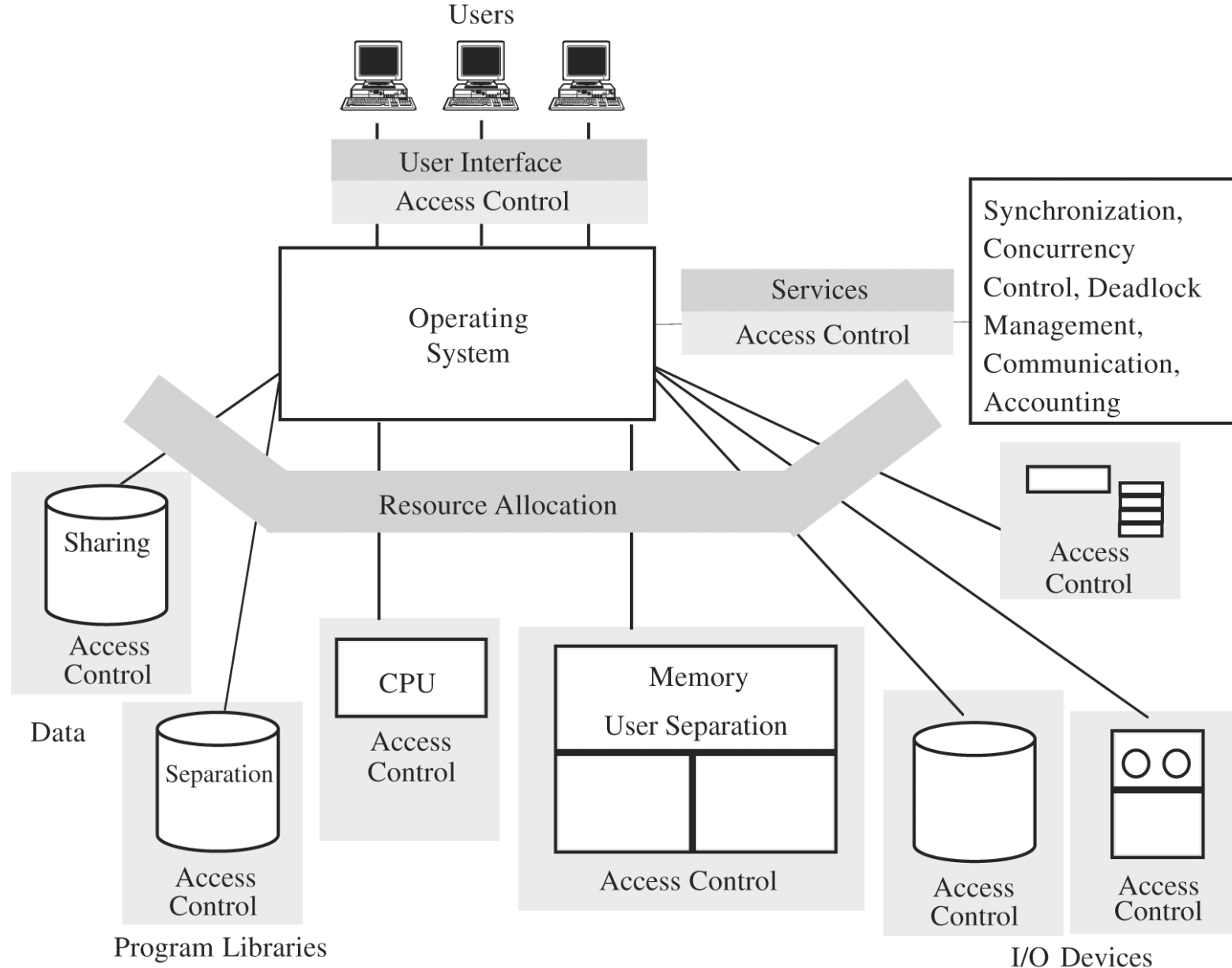
Security features in ordinary OS

- Authentication of users
 - password comparison
- Protection of memory
 - user space, paging, segmentations
- File and I/O device access control
 - access control matrix
- Allocation & access control to general objects
 - table lookup

Security features in ordinary OS...

- Enforcement of sharing
 - integrity, consistency
- Fair service
 - no starvation
- Inter process communication & synchronization
 - table lookup
- Protection of OS protection data
 - encryption, hardware control, isolation

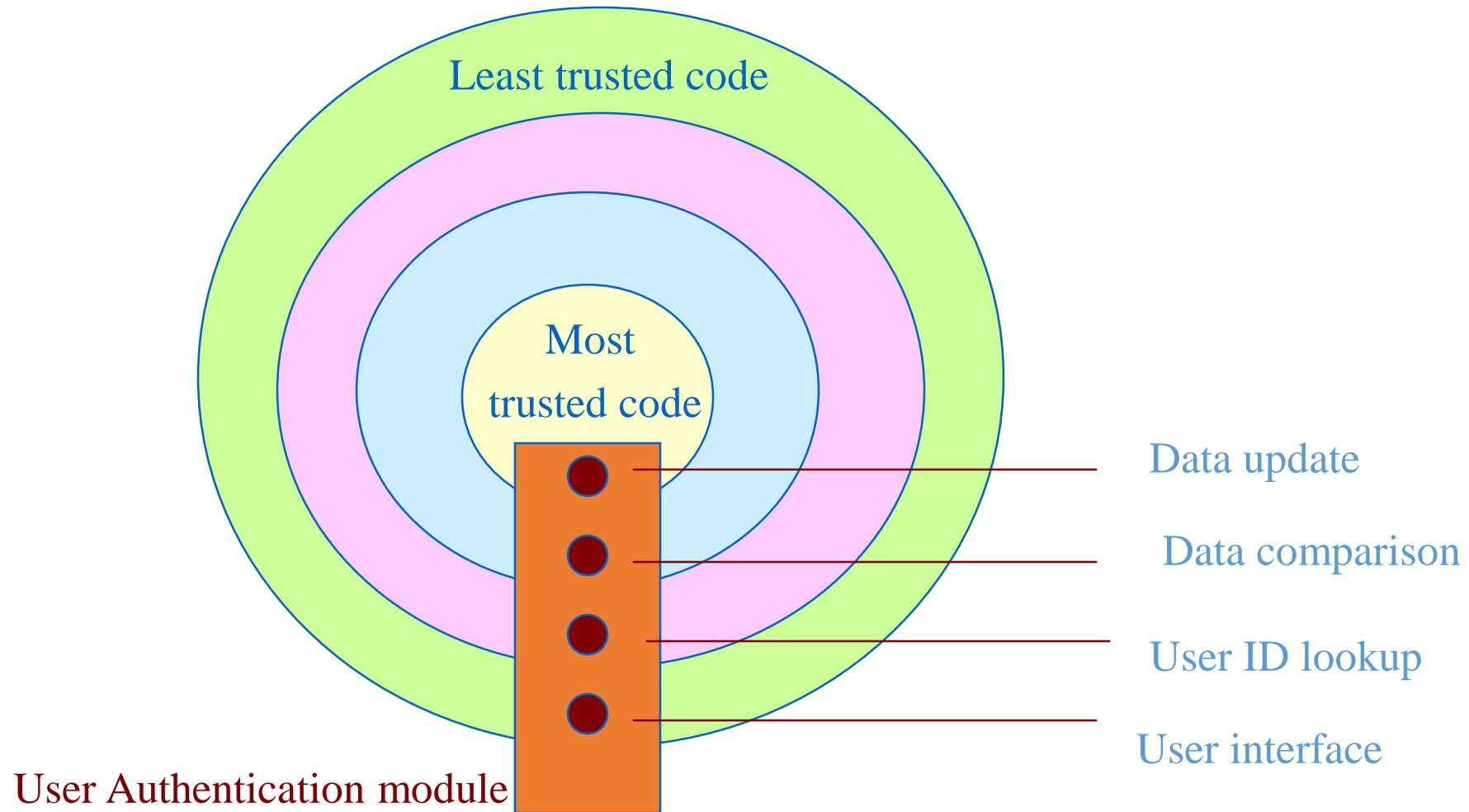
Trusted OS Functions



Security features of Trusted OS

- Identification and Authentication
- Mandatory Access Control
- Object reuse protection
- Complete mediation (all accesses are checked)
- Trusted path
- Accountability and Audit (security log)
- Audit log reduction
- Intrusion detection (patterns of normal system usages, anomalies)

Modules operating in Different Layers



Assurance

- Testing
 - based on the actual product being evaluated,
 - not on abstraction
- Verification
 - each of the system's functions works correctly
- Validation
 - developer is building the right product
 - according to the specification

Testing

- Observable effects versus internal structure
- Can demonstrate existence of a problem,
 - but passing tests does not imply absence of any
- Hard to achieve adequate test coverage within reasonable time
 - inputs & internal states
 - hard to keep track of all states
- Penetrating Testing
 - tiger team analysis, ethical hacking
 - Team of experts in design of OS tries to crack system

Formal verification

- The most rigorous method
- Rules of mathematical logic to demonstrate that a system has certain security property
- Proving a Theorem
 - Time consuming
 - Complex process

Validation

- Requirements checking
 - system does things it should do
 - also, system does not do things it is not supposed to do
- Design and code reviews
 - traceability from each requirement to design and code components
- System testing
 - data expected from reading the requirement document can be confirmed in the actual running of the system

Security in Conventional Operating Systems

Computer System Components

Hardware

- Provides basic computing resources (CPU, memory, I/O devices).

Operating system

- Controls and coordinates the use of the hardware among the various application programs.

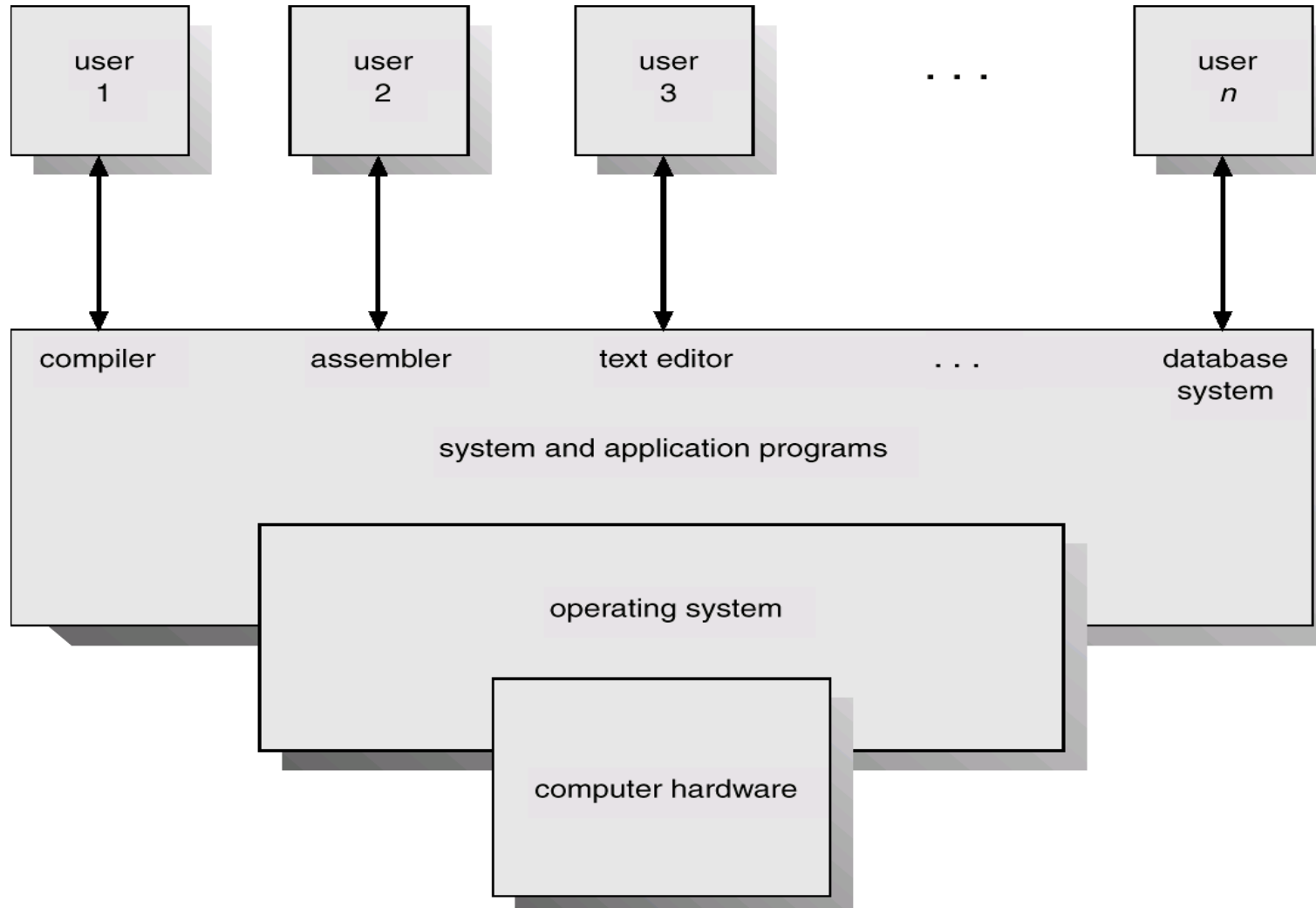
Applications programs

- Define the ways in which the system resources are used to solve the computing problems of the users.

Users

- E.g., people, machines, other computers.

Abstract View of System Components



Security Goals of Operating Systems

- Enable multiple users to securely share a computer
 - Separation and sharing of processes, memory, files, devices, etc.
- Ensure secure operation in networked environments
 - Enforce security policies while allowing resource sharing across multiple computers

How to Achieve the Security Goals

➤ Classic OS protections

- Memory and address protection
- Control of access to general objects
- File protection
- Authentication

➤ Additional OS protections

- Logging & Auditing
- Intrusion Detection
- Recovery
- Many others...

Basis of OS Protection: Separation

➤ Separation

- Keep one user's objects separate from other users

➤ Possible separations provided by the OS

- Physical separation
 - Use different physical objects (e.g., printers)
- Temporal separation
 - Executed at different times
- Logic separation
 - OS constrains programs' access so that it cannot access objects outside its permitted domain
- Cryptographic separation
 - Processes conceal their data and computation in such a way that they are unintelligible to outsiders

CPU Modes

- AKA, processor modes, privileges
- System mode (privileged mode, master mode, kernel mode)
 - Can execute any instruction and access any memory locations
 - Examples: access hardware devices, enable and disable interrupts, change privileged processor state, access memory management units, modify registers for various descriptor tables
- User mode
 - Access to memory is limited
 - Cannot execute some instructions
- Transition from user mode to system mode must be done through well defined call gates (system calls)

System Calls

- Guarded gates from user mode into kernel mode
 - Transfer control to predefined entry point in more privileged code, using a special CPU instruction (often an interruption)
 - Allow the more privileged code to specify where it will be entered as well as important processor state at the time of entry
 - The higher privileged code, by examining processor state set by the less privileged code and/or its stack, determines what is being requested and whether to allow it.

Memory and Address Protection

- **Memory protection** is a way to control **memory access** rights on a computer, and is a part of most modern instruction set architectures and operating systems.

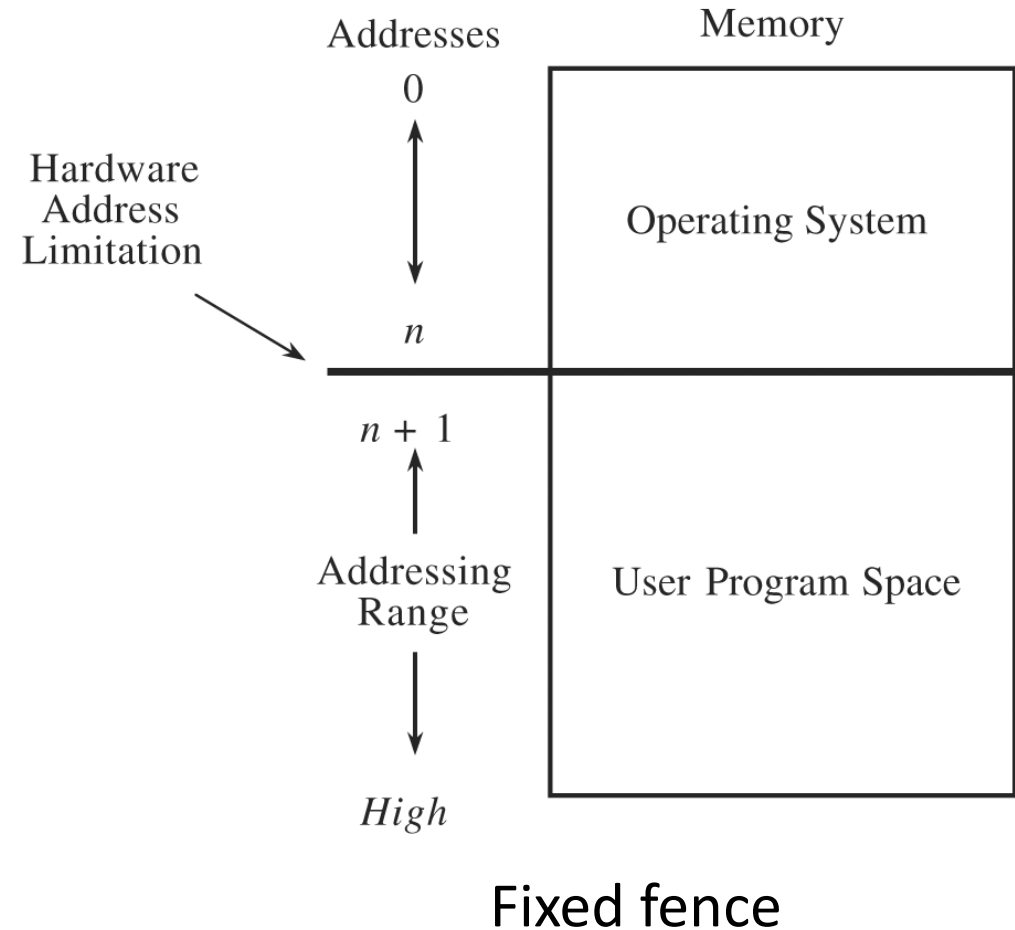
The main purpose of **memory protection** is to prevent a process from accessing **memory** that has not been allocated to it.

Various ways of memory and address protection.

- Ensures that one user's process cannot access others' memory
 - Fence
 - Relocation
 - Base/bounds register
 - Segmentation
 - Paging
 - ...
- Operating system and user processes need to have different privileges

Fence

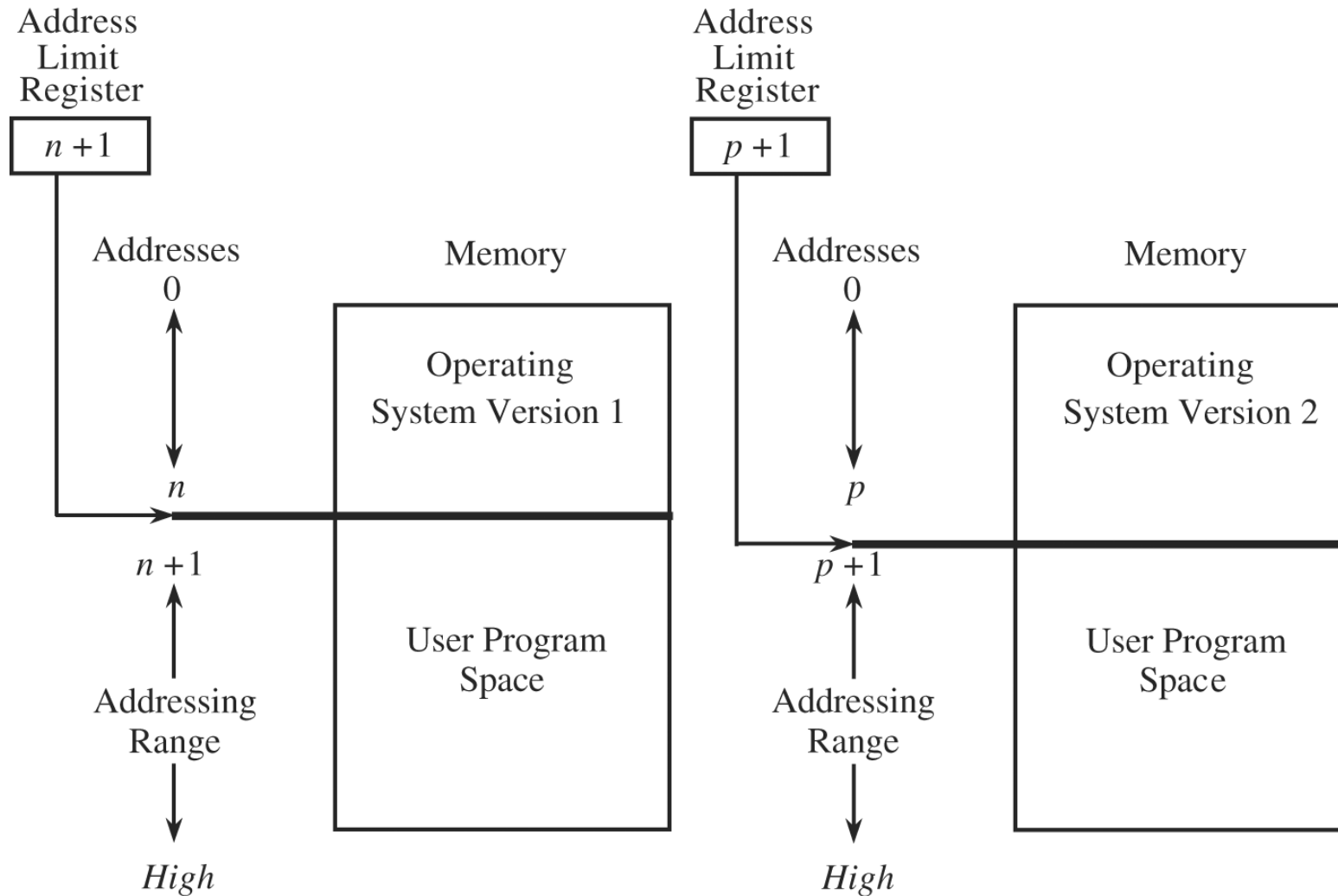
- Introduced in single-user operating systems
- Goal
 - Prevent a faulty user program from destroying part of the resident portion of the OS
- Methods
 - Predefined memory address
 - Fence register



Fence..

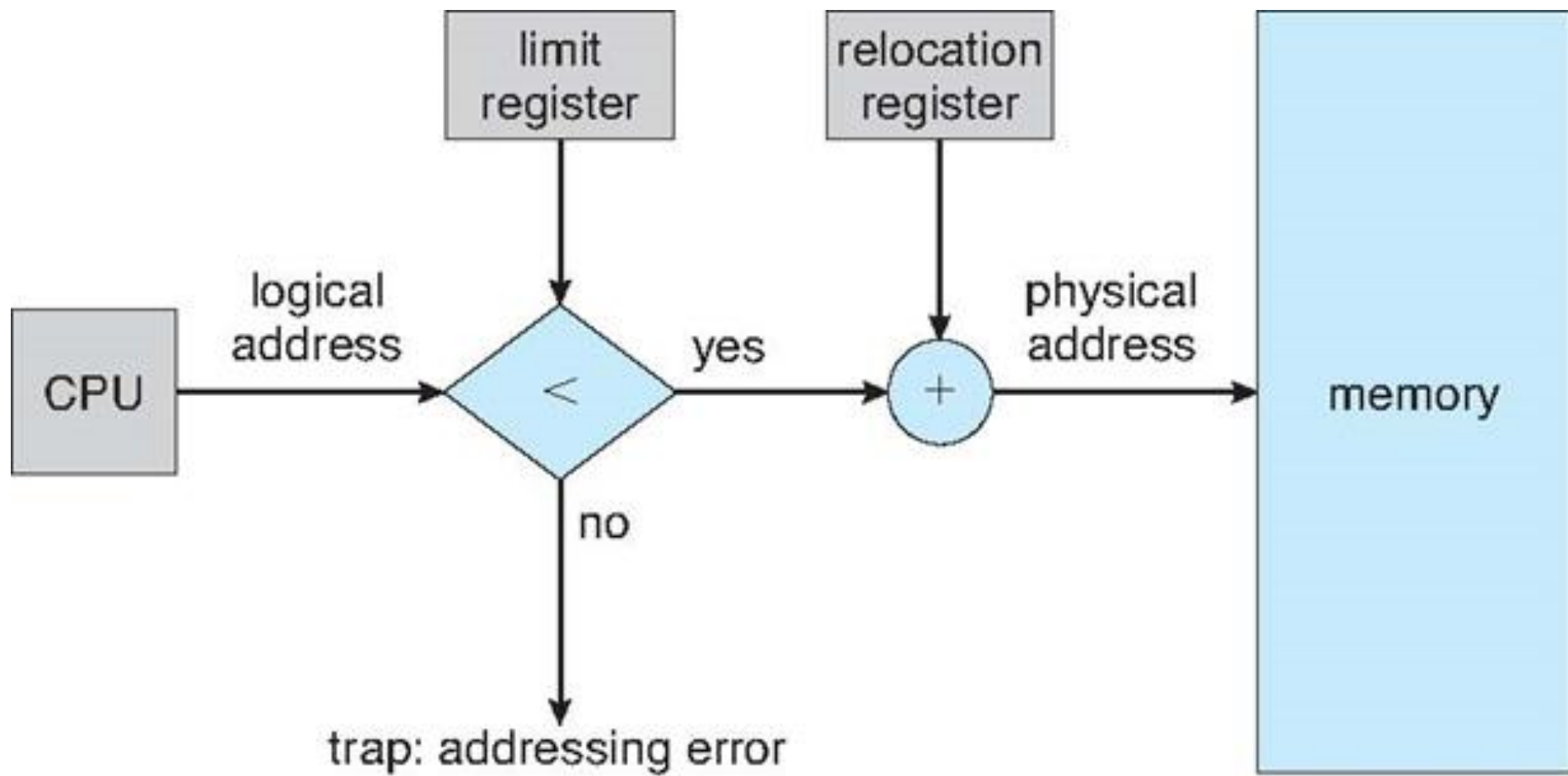
- A fence or fence address is simplest form of memory protection which can be used only for single user operating system.
- A fence is a particular address that users and their processes cannot cross. Only the OS can operate on one side of the fence and users are restricted to the other side.
- A fence could be static, in which case there is a fixed fence address. Alternatively, a dynamic fence can be used, which can be implemented using a fence register to specify the current fence address.

Variable Fence with Fence Register



Relocation

- Assuming a fixed size OS is inconvenient
- Relocation: The process of address translation
 - All programs begin at address 0
 - When loading a program into memory, change all the addresses to reflect the actual address at which the program is located
- Can be done by adding a constant relocation factor to each address of the program
- Fence register can be used for relocation



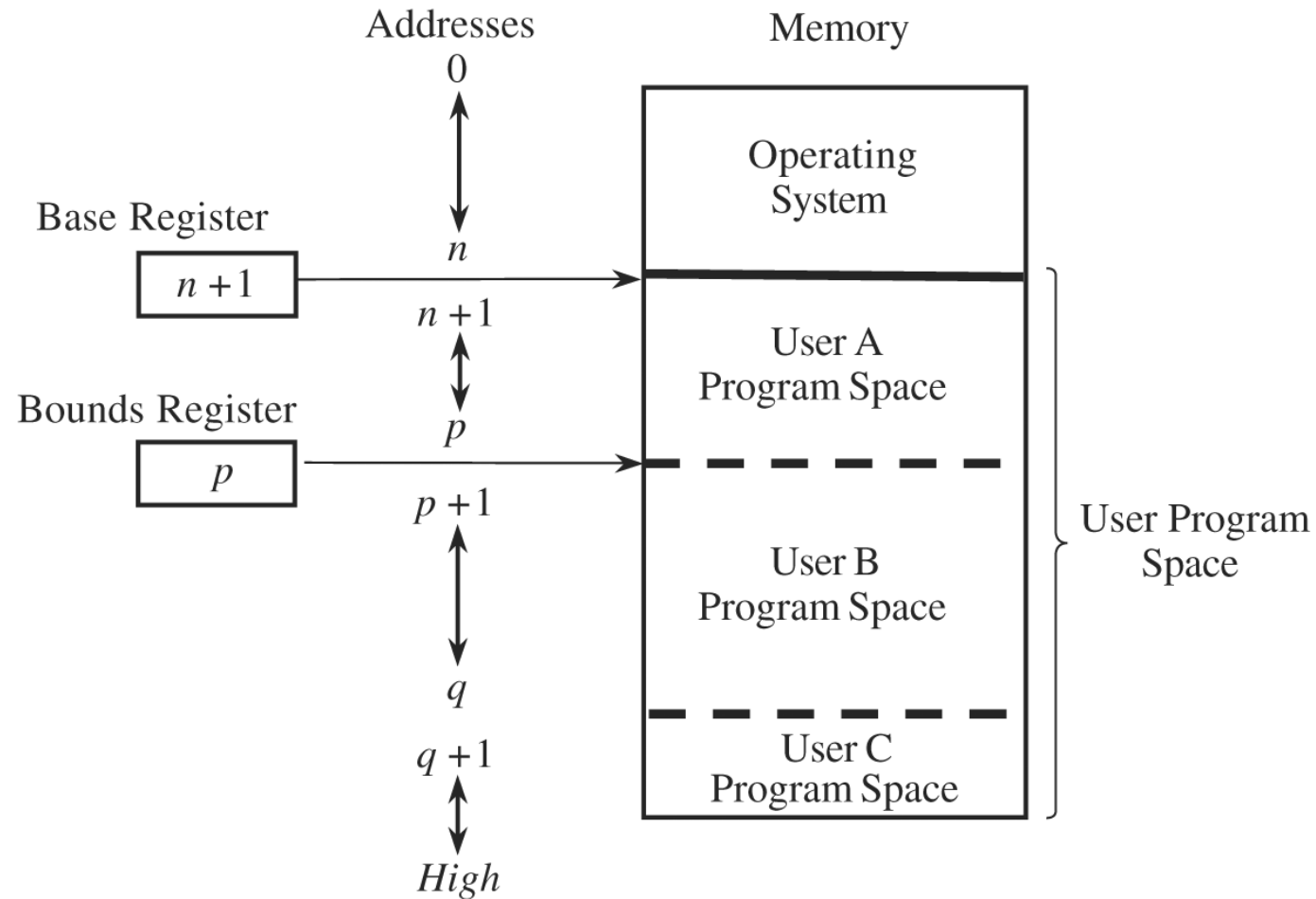
Base/Bounds Registers

- A relocation register is also called a **base register**, since it provides a base for programs
 - All addresses inside a program are offsets from the base address
 - Provides a lower bound of the program address
- Bounds register
 - Provides an upper address limit
 - Helpful for checking overflows into “forbidden” areas
- Base/bounds registers must be changed during context switch (execution changes from one program to another)

Base and Bounds registers:

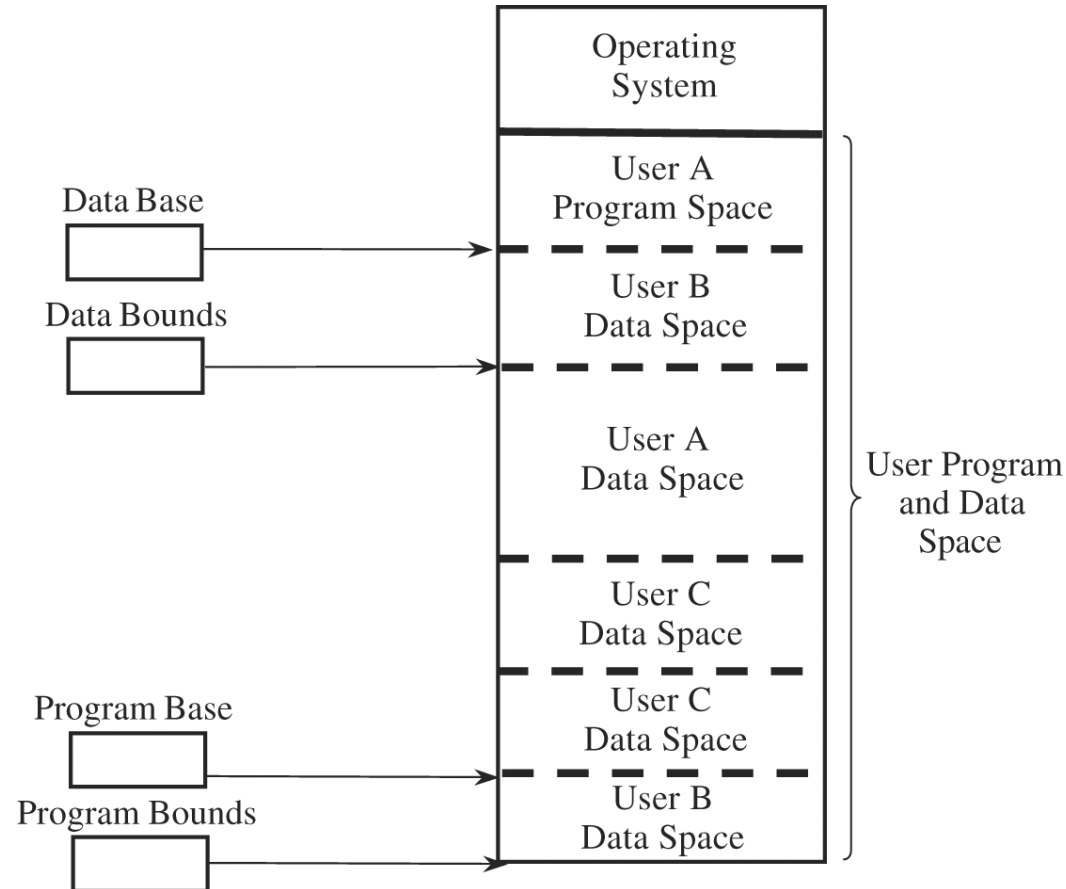
- This type of protection can be used in multi- user environment where one users program needs to be protected from the other.
- Each user has a base register which is the lower address and a Bound register which is the upper address limit.
- The base and bounds register approach implicitly assumes that the user or process space is contiguous in memory. The OS must determine what protection to apply to a specific memory location.
- In some cases it might be sufficient to apply the same protection to all of a user's memory.
- The disadvantage is that the registers confine access to consecutive range of addresses.

Base/Bounds Registers (Cont'd)








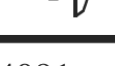
Base/Bounds Registers (Cont'd)

- Can be used to protect different regions inside a program



Tagged Architecture

- Every word of machine memory has one or more extra bits to identify the access rights to that word.
- These bits can be set by privileged instructions
- The bits are tested every time an instruction accesses that location
- Has been used in a few systems
- Expensive

Tag	Memory Word
R	0001
RW	0137
R	0099
X	
X	
X	
X	
X	
X	
R	4091
RW	0002

Code: R = Read-only RW = Read/Write
X = Execute-only

Tagging:

- This specifies the protection for each individual address. In this method of protection every word of machine memory has one or more extra bits to identify the access rights to that word.
- Only privileged instructions can set these access bits. While this is as fine-grained protection as possible, it introduces significant overhead.
- The overhead can be reduced by tagging sections of the address space instead of each individual address. Another drawback to tagging is compatibility, since tagging schemes are not in common use.

Tagged Architecture (Cont'd)

- A variation
 - One tag applies to a group of consecutive locations (e.g., 128 bytes)
 - Much reduced cost
 - Allow more bits in a tag

Segmentation

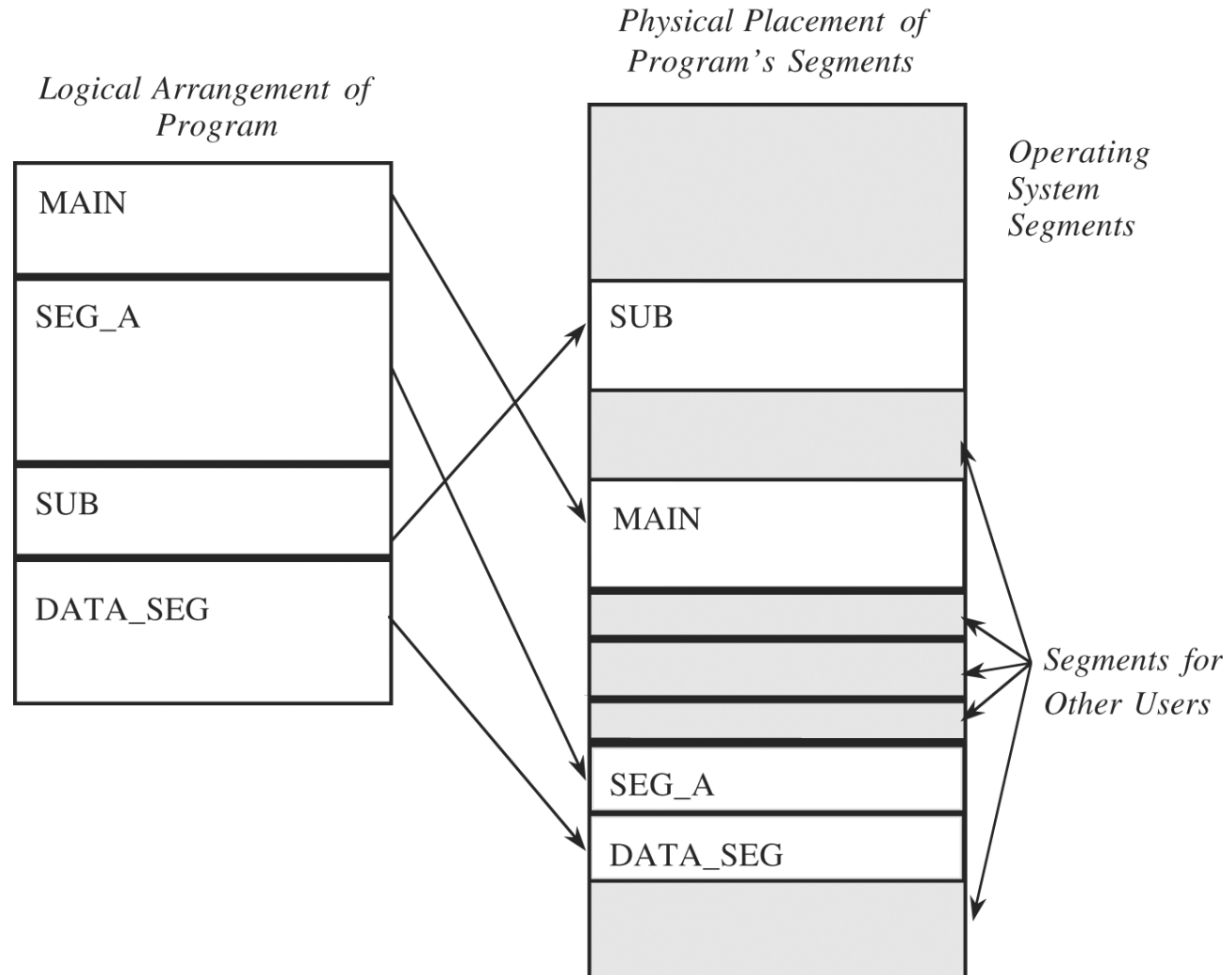
- Divide a program into several pieces
 - Example:
 - Code segment, Data segment, ...
 - A feasible means to have the effect of an unbounded number of base/bounds registers
 - Allows a program to be divided into many pieces having different access rights
 - A memory location is addressed as <name, offset>
 - Name: name of the segment
 - Offset: location within the segment

Segmentation:

- This method divides the memory into logical units such as individual procedures or the data in one array.
- Once they are divided, appropriate access control can be enforced on each segment.
- A benefit of segmentation is that any segment can be placed in any memory location provided the location is large enough to hold it. The OS must keep track of the locations of all segments, which is accomplished using <segment,offset> pairs, where the named segment specifies the segment, and the offset is the starting address of the specified segment.
- With segmentation, all address references must go through the OS, so the OS can, in this respect, achieve complete mediation. Depending on the access control applied to particular segments, users can share access to some segments or users can be restricted to specific segments.

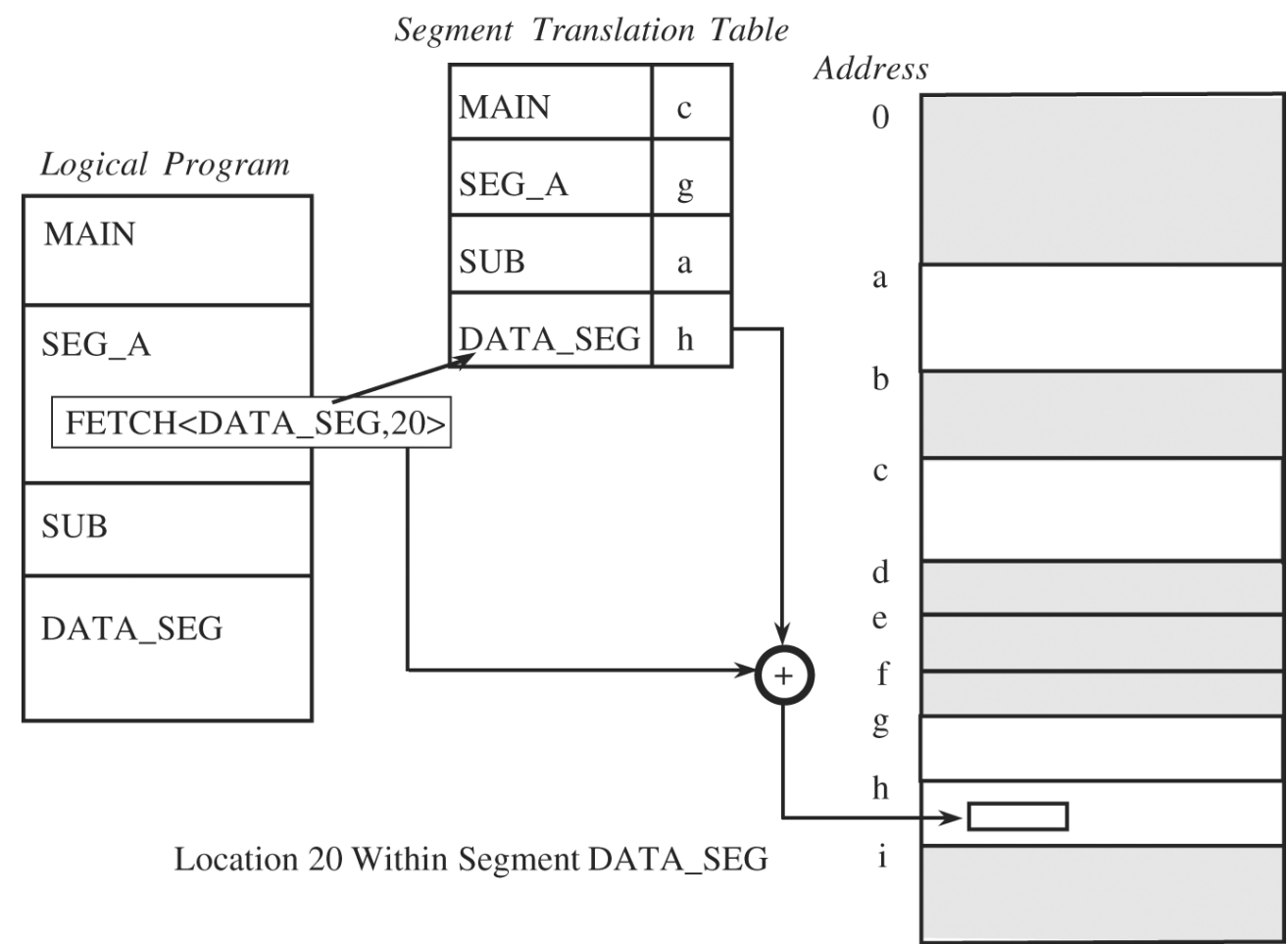
Segmentation (Cont'd)

- Logical and physical representation of segments



Segmentation (Cont'd)

- Translation of segment address



Segmentation (Cont'd)

- Benefits for the OS
 - The OS can place any segment at any location, or move any segment to any location
 - A segment can be removed from main memory if it's not being used currently
 - Every address reference passes through the OS. So there is an opportunity to check each one for protection
- Benefits for protection
 - Each address is checked for protection
 - Many different classes of data items can be assigned different levels of protection
 - Two or more users can share access to a segment, with different access rights
 - A user cannot generate an address or access to an unpermitted segment

Segmentation (Cont'd)

- Limitations
 - Overhead to deal with segmentation size
 - Segments need to be dynamic
 - Segment length must be maintained, and compared with every address generated
 - Segmentation leads to fragmentation of memory
 - Poor memory utilization
 - Compacting and updating appropriate tables take time

Paging:

- Paging discards the disadvantage of segmentation. In paging all segments are of a fixed size called as pages and the memory divided is known as page frames.
- In paging a particular page can be accessed using a pair of the form <page, offset=""> where page is the page number and offset is location within a page.
- The advantages of paging over segmentation include no fragmentation, improved efficiency, and the fact that there are no variable sizes to worry about.
- The disadvantages are that there is, in general, no logical unity to pages, which makes it more difficult to determine the proper access control to apply to a given page.

Figure 2: Memory Paging

