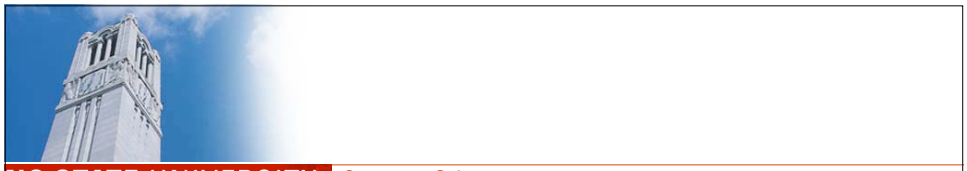


NC STATE UNIVERSITY Computer Science

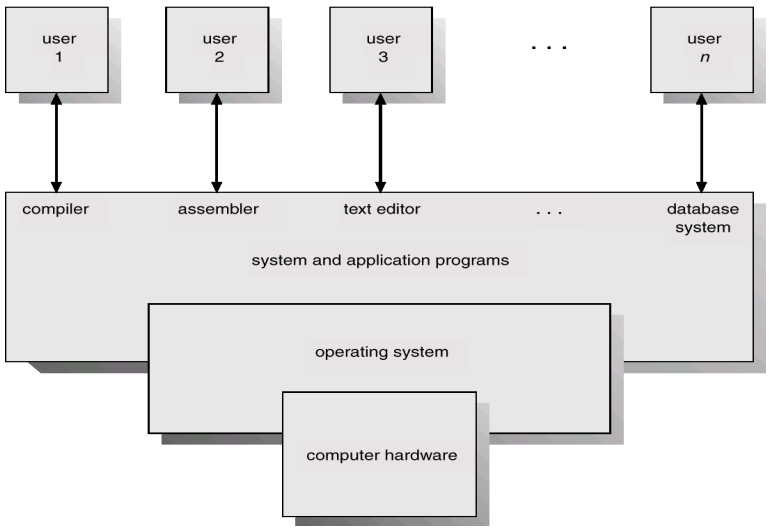
CSC 405
Introduction to Computer Security

Topic 4. Security in Conventional Operating Systems -- Part I

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 form 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 outsides 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)

Reading: http://en.wikipedia.org/wiki/CPU_modes

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.

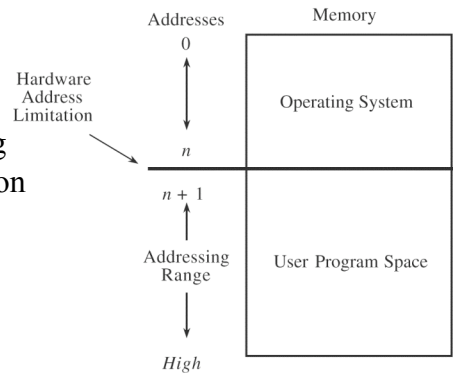
http://en.wikipedia.org/wiki/System_call

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

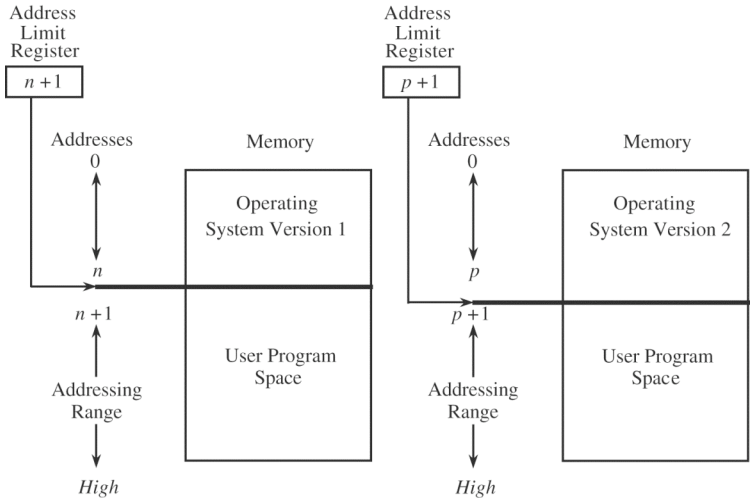
Fence

- 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



Fixed fence

Variable Fence with Fence Register



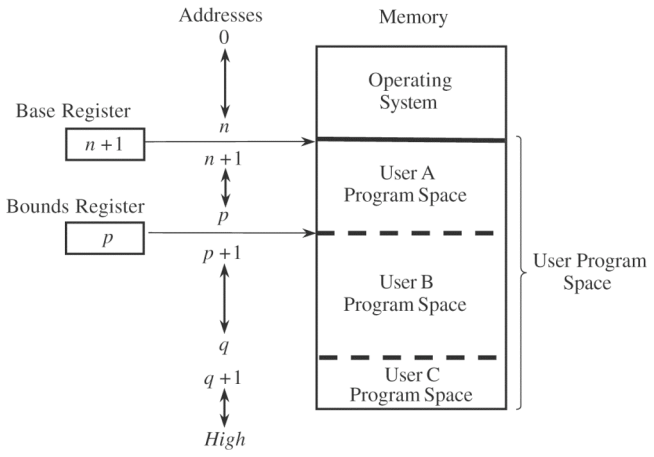
Relocation

- 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

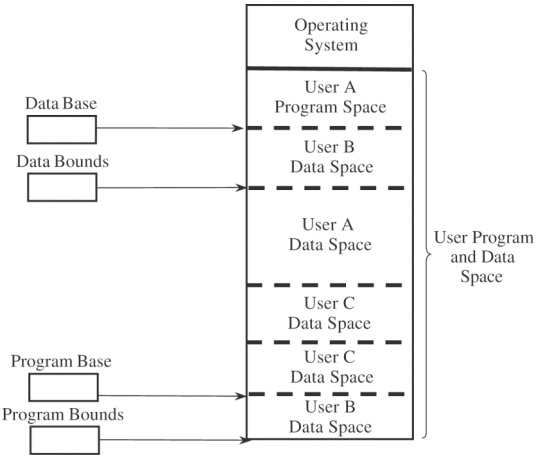
- A relocation register is also called a base register, since it provides a base (or starting address) for programs
  - All addresses inside a program are offsets from the base address
  - Provides a lower bound of the program address
- Bounds register
  - Provides an upper address limit
  - Helpful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

Base/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

Tagged Architecture (Cont'd)

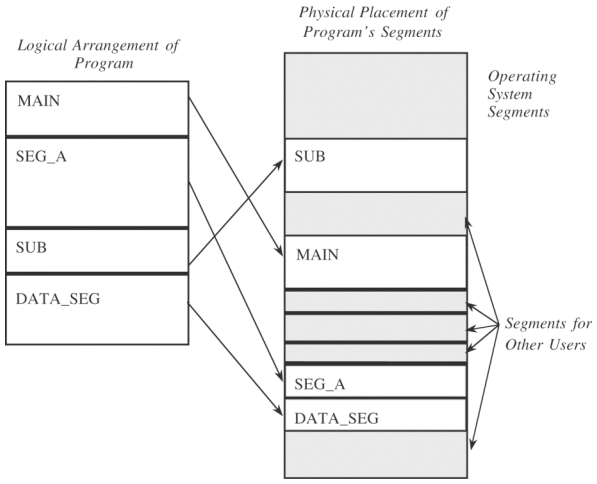
- A variation
  - One tag applies to a group of consecutive locations (e.g., 128 bytes)
  - Much reduced cost
  - Allow more bits in a tag

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

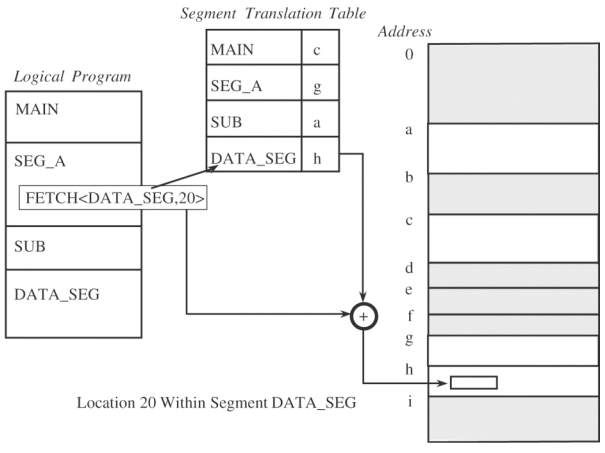
Segmentation (Cont'd)

- 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)

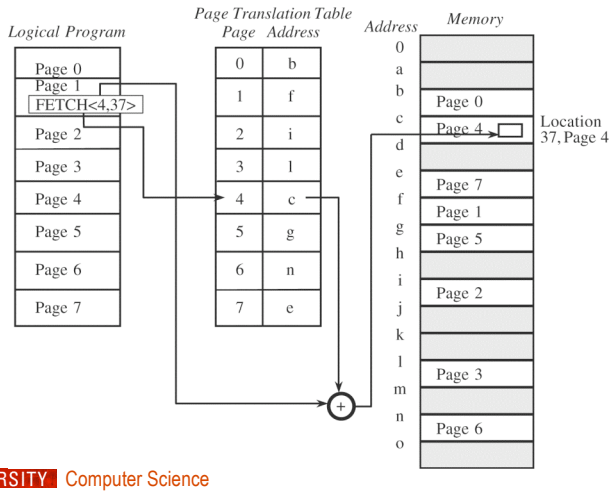
- Limitations
  - Overhead to deal with segmentation size
    - Segments need to be dynamic
    - Segment length must be maintained, and compared with every address generated
  - Segmentation leads to fragmentation of memory
    - Poor memory utilization
    - Compacting and updating appropriate tables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

Paging

- An alternative to segmentation
- Program is divided into equal-sized pieces called pages
- Memory is divided into equal-sized units, called page frames
- Each address in a paging scheme consists of <page, offset>
- All pages are of the same size
  - Fragmentation is not a problem

Paging (Cont'd)

- Address translation
  - Each page is converted to a page frame address



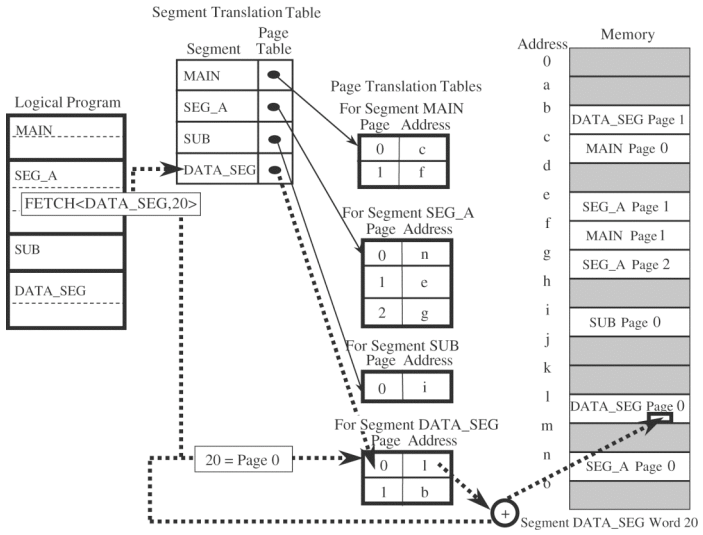
Paging (Cont'd)

- Limitations
  - No logical unit to specify protection rights
  - In contrast, segments are logical units to associate protection rights

Combined Paging with Segmentation

- Segmentation followed by paging
  - Segments offer logical units for protection rights
  - Each segment is accessed through paging

Combined Paging with Segmentation (Cont'd)



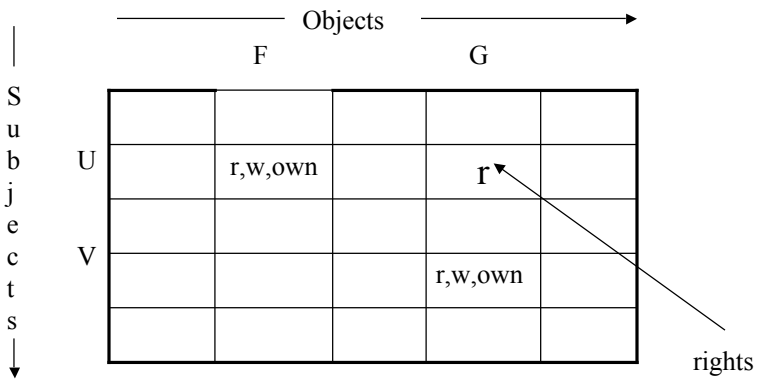
Control of Access to General Objects

- Methods
  - Access control matrix
  - Access control list (ACL)
  - Capabilities

Access Matrix Model

- Basic Abstractions
  - Subjects: A subject is a program (application) executing on behalf of a user
  - Objects: An object is anything on which a subject can perform operations (mediated by rights)
  - Rights
- Subjects and objects are relative terms
  - A subject in one context can be an object in another context
  - Example:
    - A process is a subject when it reads a file
    - The same process is an object when another process kills it.

Access Matrix Model (Cont'd)



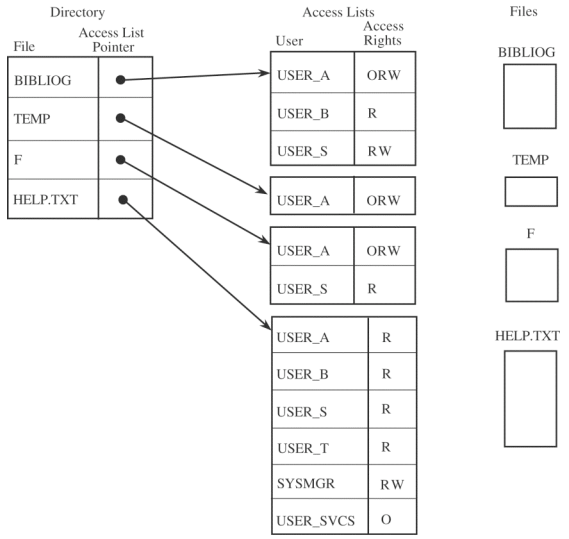
Implementation of Access Matrix Model

- Access Control Lists
- Capabilities
- Relations

Access Control List (ACL)

- Store the column of the access matrix with the objects
- Associated with each object is a list of all subjects that have access to the object and the type of access allowed
  - Default rights are given for domains not listed
  - Access is permitted if at least one entry allows it
- If too many subjects exist, they can be grouped into classes
  - The Unix group id is an example of subject class
- Only the object may modify the access list
  - Protection is implemented by the object, so revocation of rights is easy

ACL Example



Capabilities

- A capability is an unforgeable token that gives a subject certain rights to an object
- For each subject (or each group of subjects): set of objects that can be accessed + access rights
- Problem: potentially long lists of capabilities, especially for super-users.

Capabilities (Cont'd)

- To avoid tempering w/ capabilities by a subject (process), capabilities must be protected from access:
  - Tagged architecture -- capabilities have a tag bit and can only be modified in kernel mode
  - Kernel space capabilities -- capabilities are stored in the kernel and can only be accessed indirectly (e.g. file descriptors in Unix)
  - Encrypted capabilities – can be stored in user space

Capabilities (Cont'd)

- Capabilities: object-specific rights (read, write, execute) + generic rights (limit distribution and/or copy/deletion of capabilities)
- Revocation of rights is hard
  - One solution: Use version numbers incremented by objects and reject capabilities with older versions

Access Control Triples (Relations)

Subject	Access	Object
U	R	F
U	W	F
U	Own	F
U	R	G
V	R	G
V	W	G
V	Own	G

Usually in Database Management Systems.

Example: Unix

- Each subject (process) and object (e.g., file) has a user id
- Each object also has a group id and each subject has one or more group ids
- Objects have access control lists that specify read, write, and execute permissions for user, group, and world
- A subject can r,w,x an object if:
  - It has the same uid and the appropriate user permission is set
  - One of its gid's is the same as that of the object and the appropriate group permission is set, or
  - The appropriate world permission is set
  - Exception: Super-users (uid root) can do anything

ACL VS Capabilities

- Access review
  - ACL provides for superior access review on a per-object basis
  - Capabilities provide for superior access review on a per-subject basis
- Revocation
  - ACL provides for superior revocation facilities on a per-object basis
  - Capabilities provide for superior revocation facilities on a per-subject basis

ACL VS Capabilities

- The per-object basis usually wins out so most Operating Systems protect files by means of ACL
- Many Operating Systems use an abbreviated form of ACL with just three entries
  - owner
  - group
  - other

ACL's VS Capabilities

- Least Privilege
  - Capabilities provide for finer grained least privilege control with respect to subjects, especially dynamic short-lived subjects created for specific tasks

Procedure-Oriented Access Control

- Procedure-oriented access control
  - Existence of procedures that control access to objects
  - A procedure forms a capsule around the object, permitting only certain specified accesses
- Examples
  - Access to the table of valid users
    - Procedure to add a user
    - Procedure to delete a user
    - Procedure to check whether a name corresponds to a valid user

Content Dependent Controls

- Content dependent controls such as
  - You can only see salaries less than 50K, or
  - You can only see salaries of employees who report to you
- Beyond the scope of Operating Systems and are provided by Database Management Systems

Context Dependent Controls

- Context dependent controls such as
  - You cannot access classified information via a remote login
  - Salary information can be updated only at year end
  - The company's earnings report is confidential until announced at the stockholders meeting
- Can be partially provided by the Operating System and partially by the Database Management System
- More sophisticated context dependent controls such as based on past history of accesses definitely require Database support

Discretionary Access Control v.s. Mandatory Access Control

- Discretionary Access Control (DAC)
  - Allow access rights to be propagated from one subject to another
  - Possession of an access right by a subject is sufficient to allow access to the object
  - Access decision is controlled by the owner of access privileges
- Mandatory Access Control (MAC)
  - Restrict the access of subjects to objects on the basis of security labels
  - Access decision is controlled by the OS