Protection

Module 5.1

Protection

- Mechanism for controlling the access of programs, processes or users to the resources defined by a computer system

- Provide a means of specification of controls(policies) and means of enforcement of policies governing resource use

- Prevent mischievous, intentional violation of an access restriction by a user.

Goals of Protection

- Operating system consists of a collection of objects, hardware or software
- Each object has a unique name and can be accessed through a well-defined set of operations.
- **Protection problem** - ensure that each object is accessed correctly and only by those processes that are allowed to do so.

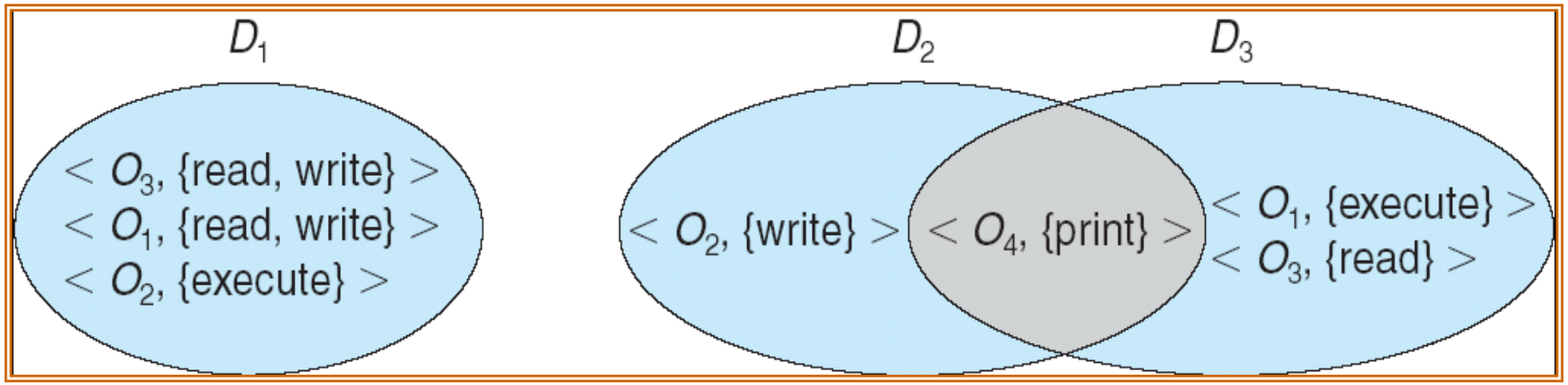
Domain Structure

- A process operates within a protection domain

- Access-right = <*object-name*, *rights-set*>
  where *rights-set* is a subset of all valid operations that can be performed on the object.

- Domain = set of objects with access-rights

- **A domain may be realized as:**
  - User
  - Process
  - procedure


D_1
D_2
D_3
< O_3, {read, write} >
< O_1, {read, write} >
< O_2, {execute} >
< O_2, {write} >
< O_4, {print} >
< O_1, {execute} >
< O_3, {read} >

Access Matrix

- View protection as a matrix (*access matrix*)
- Rows represent domains
- Columns represent objects
- *Access(i, j)* is the set of operations that a process executing in $Domain_i$ can invoke on $Object_j$

Access Matrix

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Figure A

Use of Access Matrix

- Can be expanded to dynamic protection.
  - Operations to add, delete access rights.
  - Special access rights:
    - *owner of* O_i
    - *copy op from* O_i *to* O_j
    - *control –* D_i *can modify* D_j *access rights*
    - *transfer – switch from domain* D_i *to* D_j

Use of Access Matrix

- If a process in Domain D_i tries to do "op" on object O_j, then "op" must be in the access matrix.
- Access matrix design separates mechanism from policy.
  - Mechanism
    - Operating system provides access-matrix + rules.
    - If ensures that the matrix is only manipulated by authorized agents and that rules are strictly enforced.
  - Policy
    - User dictates policy.
    - Who can access what object and in what mode.

Access Matrix of Figure A With Domains as Objects

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Figure B

Implementation of Access Matrix

1. Global Table

- The global table is the most basic and simple implementation of the access matrix in the operating system which consists of a set of an ordered triple <domain , object, right-set>.
- When an operation M is being executed on an object Oj within domain Di, the global table searches for a triple <Domain(Di), Object(Oj), right-set(Rk)> where M € Rk.
- If the triple is present, the operation can proceed to continue, or else a condition of an **exception** is thrown.

Implementation of Access Matrix

- There are various drawbacks to this implementation, the main **drawback** of global table implementation is that because the table is sometimes too large, it cannot be stored in the main memory, that's why input and output are required additionally.

Implementation of Access Matrix

2. Access Lists for Objects

- Every access matrix column may be used as a single object's access list.
- It is possible to delete the blank entries.
- For each object, the resulting list contains ordered pairs **<domain, rights-set>** that define all domains for that object and a nonempty set of access rights.
- We may start by checking the default set and then find the access list. If the item is found, we enable the action; if it isn't, we verify the default set. If M is in the default set, we grant access. Access is denied if this is not the case, and an extraordinary scenario arises.

Implementation of Access Matrix

3. Capability list

- A domain's capability list is a collection of objects and the actions that can be done on them.
- A capacity is a name or address that is used to define an object.
- If you want to perform operation M on object **Oj,** the process runs operation M, specifying the capability for object **Oj.** The simple possession of the capability implies that access is allowed.

Implementation of Access Matrix

- In most cases, capabilities are separated from other data in one of two ways.
- Every object has a tag to indicate its type as capability data. Alternatively, a program's address space can be divided into two portions.
- The programs may access one portion, including the program's normal instructions and data.
- The other portion is a capability list that is only accessed by the operating system.

Implementation of Access Matrix

4. Lock- Key Mechanism

- **Locks**:
  - Locks represent the objects or resources you want to protect. These could be files, directories, memory segments, devices, or any other system resources. Each lock corresponds to an object in the Access Matrix.
- **Keys**:
  - Keys represent the permissions or access rights that subjects (users, processes, or entities) need to access the objects (locks). In the context of the Access Matrix, keys can be thought of as specific operations like "read," "write," "execute," etc.

Implementation of Access Matrix

- Here's how it works:
- Each object (lock) has associated keys (access rights) that determine what actions can be performed on the object.
- These keys are essentially the permissions defined in the Access Matrix. For example, a file may have keys for "read," "write," and "execute."
- Subjects (users or processes) are given keys that correspond to the objects they are allowed to access and the operations they are permitted to perform.
- This is similar to users being given keys to unlock certain locks.

Implementation of Access Matrix

- To access an object (unlock a lock), a subject must possess the appropriate key (permission) for that object. If they have the right key, they can perform the associated operation on the object. If they don't have the key (permission), they are denied access to the object.
- In this way, the lock-key mechanism provides a straightforward way to enforce access control based on the Access Matrix. It ensures that only authorized subjects (those with the correct keys) can access specific objects and perform specific operations.
- The operating system's security mechanism, which checks whether a subject has the required key (permission) before allowing access to an object, is responsible for enforcing the lock-key system.

Revocation of Access Rights

- Revocation of access rights refers to the process of taking away or revoking previously granted access permissions from a subject (user, process, or entity) to a particular resource or object within an operating system.
- This is an important security measure, as it allows administrators or system owners to manage and control access to resources dynamically, especially when access is no longer needed or poses a security risk.
- Here are the key steps involved in the revocation of access rights:

- **Identify the Access to Be Revoked:**
- Determine which subjects have access to the specific resource or object for which access rights need to be revoked. This may involve reviewing access control lists (ACLs), capabilities, or other access control mechanisms.
- **Decide the Scope of Revocation:**
- Determine whether you want to revoke all access rights completely or just specific permissions. You can choose to revoke specific operations (e.g., "write" permission) or revoke all access entirely.
- **Authenticate and Authorize:**
- Ensure that the request to revoke access is authenticated and authorized. Only authorized personnel or administrators should be allowed to perform access revocation.

- **Modify Access Control Lists (ACLs) or Capabilities:**
- If you are using ACLs or capabilities, update the access control entries to remove the subject's access rights. This may involve removing the subject from the ACL, removing specific permissions, or invalidating the subject's capabilities.
- **Update the Access Matrix:**
- If you are using an Access Matrix, update the matrix to reflect the changes in access rights. Modify the appropriate cell in the matrix to reflect the revoked access.
- **Notify Affected Subjects:**
- In some cases, it's important to inform the affected subjects that their access rights have been revoked. This can be for transparency and to avoid confusion.

- **Implement the Revocation:**
- Actually enforce the revocation. The operating system's access control mechanisms should now deny access to the subject according to the changes made in step 4 or 5.
- **Monitor and Audit:**
- Continuously monitor and audit the system to ensure that the access rights have been effectively revoked and that there are no unintended consequences.

- **Regular Review:**
- Periodically review access rights and permissions to ensure that revoked access remains revoked and that access is adjusted as needed to reflect changes in security requirements.
- It's important to note that the process of revoking access rights should be carefully managed and documented to maintain the integrity and security of the system. In some cases, there may be legal or compliance requirements regarding access revocation, so it's essential to adhere to those as well.