Chapter 14: Protection

Chapter 14: Protection

- Goals of Protection
- Principles of Protection
- Domain of Protection
- Access Matrix
- ~~Implementation of Access Matrix~~
- Access Control

Goals of Protection

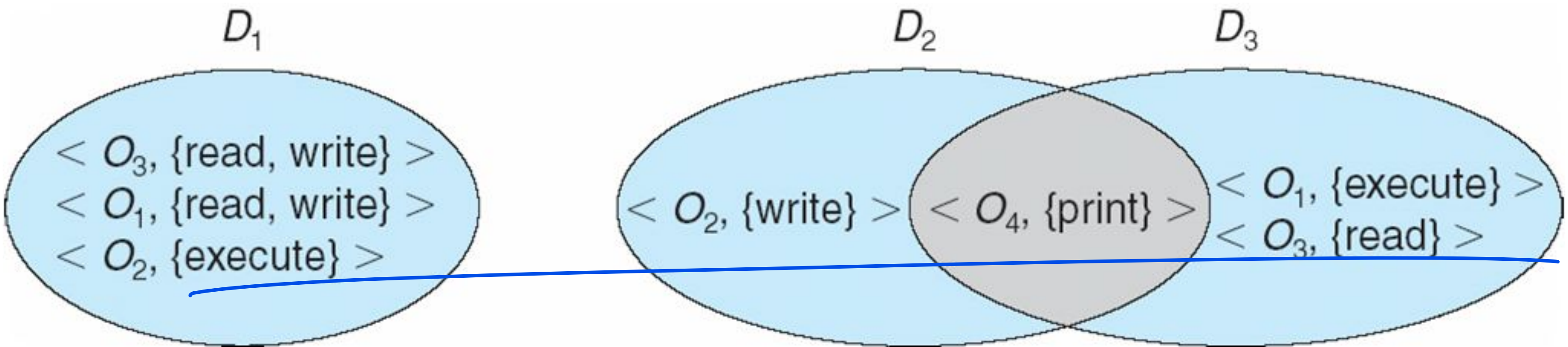
- Operating system consists of a collection of objects.
- Objects can be hardware objects(such as the CPU, memory segments, printers, disks, and tape drives), or software objects(such as files, programs, and semaphores).
- Each object has a unique name and can be accessed through a well-defined set of operations
- Protection problem - ensure that each object is accessed correctly and only by those processes that are allowed to do so
- A process should be allowed to access only those resources for which it has authorization. At any time, a process should be able to access only those resources that it currently requires to complete its task. This second requirement, commonly referred to as the **need-to-know principle**, is useful in limiting the amount of damage a faulty process can cause in the system

Principles of Protection

- Guiding principle – principle of least privilege
  - Programs, users and systems should be given **just enough** privileges to perform their tasks

Domain Structure

- A process operates within a protection domain which specifies the resources that the process may access.
- Each domain defines a set of objects and the types of operations that may be invoked on each object.
- The ability to execute an operation on an object is an Access right.
- 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 access-rights



D_1

< O_3, {read, write} >
< O_1, {read, write} >
< O_2, {execute} >

D_2

< O_2, {write} >

< O_4, {print} >

D_3

< O_1, {execute} >
< O_3, {read} >

Domain Implementation (UNIX)

- System consists of 2 domains:
  - User
  - Supervisor
- UNIX
  - Domain = user-id
  - Domain switch accomplished via file system
    - 4 Each file has associated with it a domain bit (setuid bit)
    - 4 When file is executed and setuid = on, then user-id is set to owner of the file being executed. When execution completes user-id is reset

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	

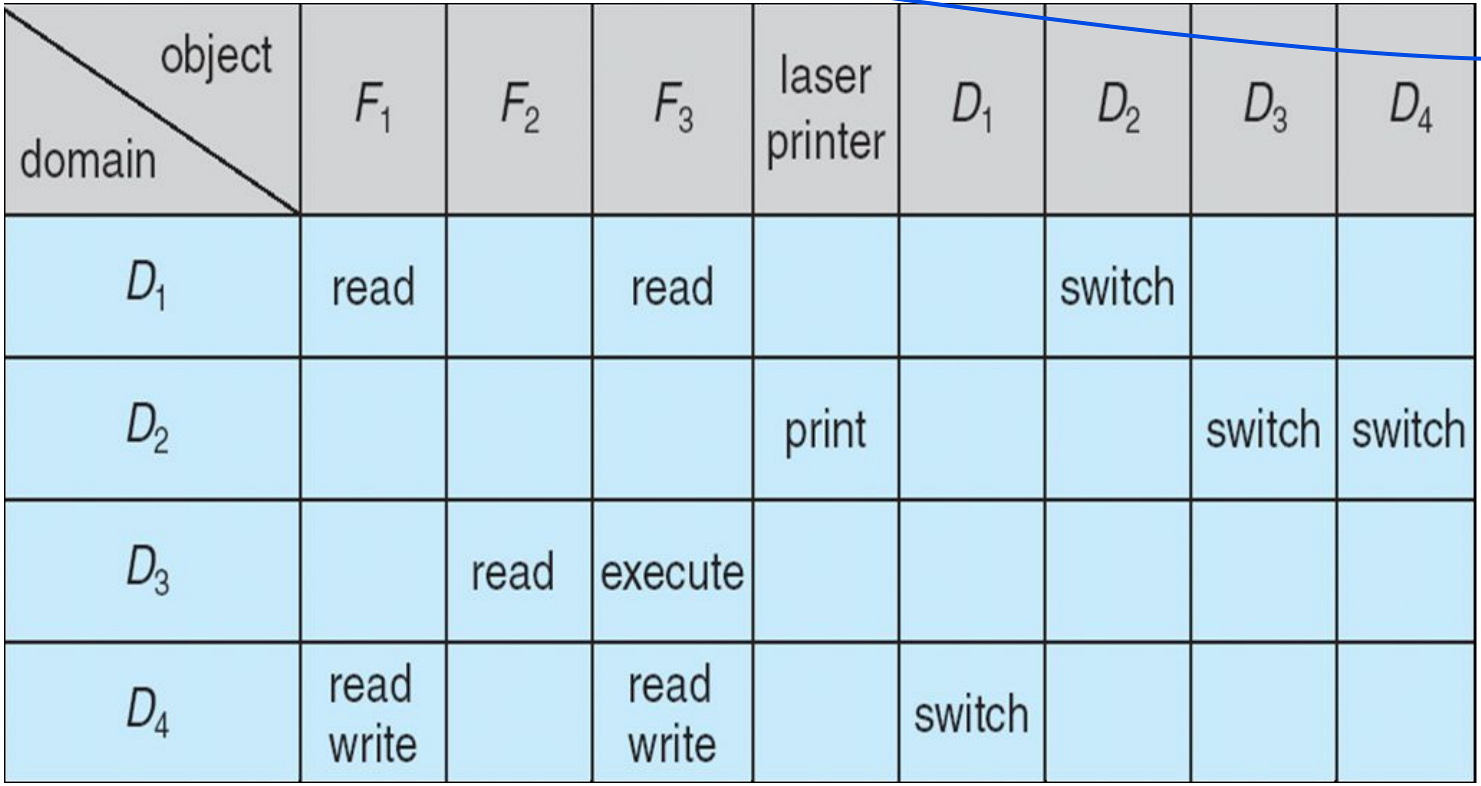
Use of Access Matrix

- If a process in Domain D_i tries to do "op" on object O_j, then "op" must be in the access matrix
- Can be expanded to dynamic protection
  - Operations to add, delete access rights
  - Special access rights:
    - 4 *owner of O_i*
    - 4 *copy op from O_i to O_j*
    - 4 *control – D_i can modify D_j access rights*
    - 4 *transfer – switch from domain D_i to D_j*

Use of Access Matrix (Cont)

- 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

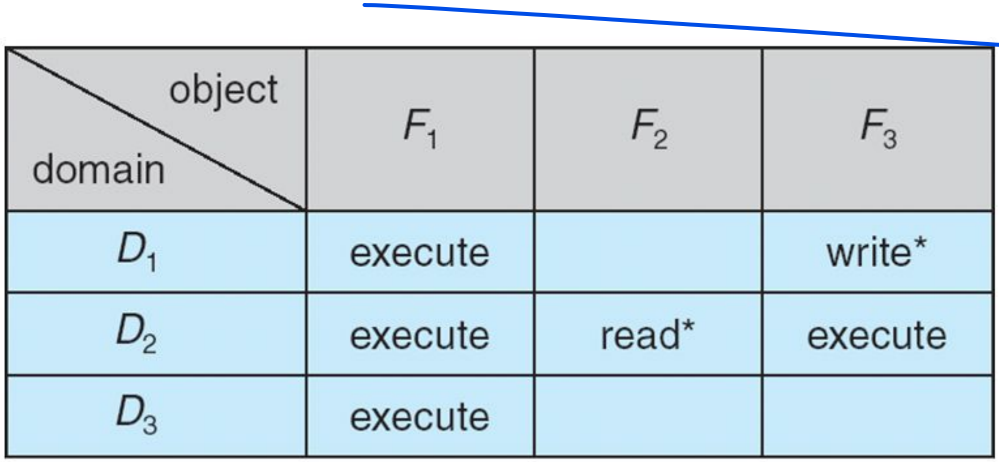
Access Matrix of Figure A With Domains as Objects



object domain	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			

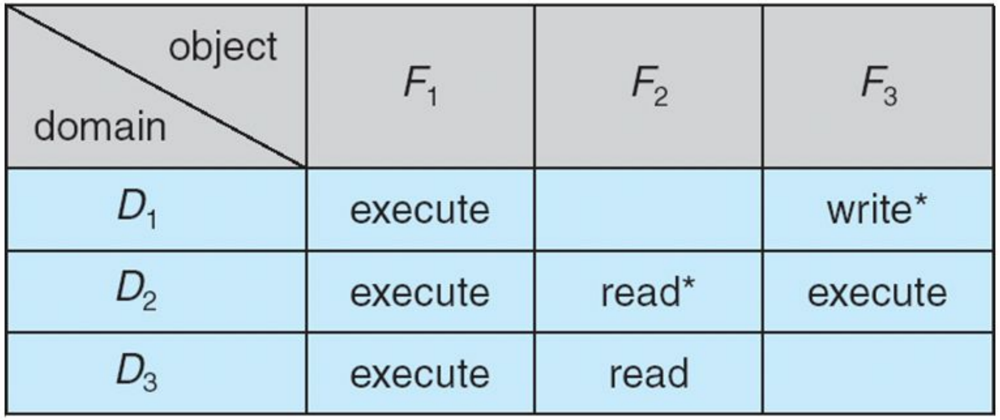
Figure B

Access Matrix with *Copy* Rights



object domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

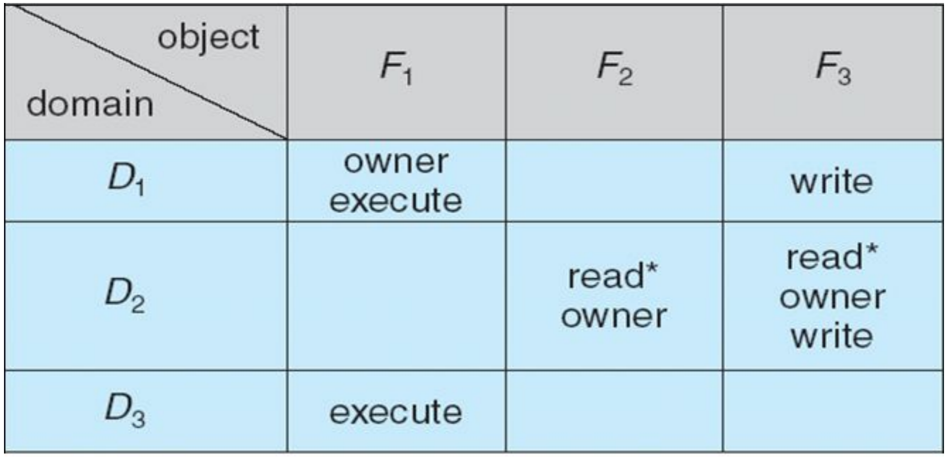
(a)



object domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

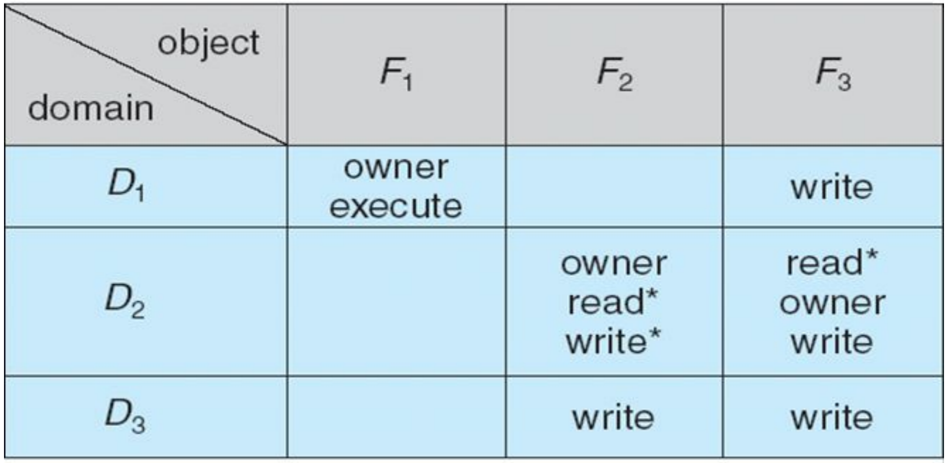
(b)

Access Matrix With *Owner* Rights



object domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

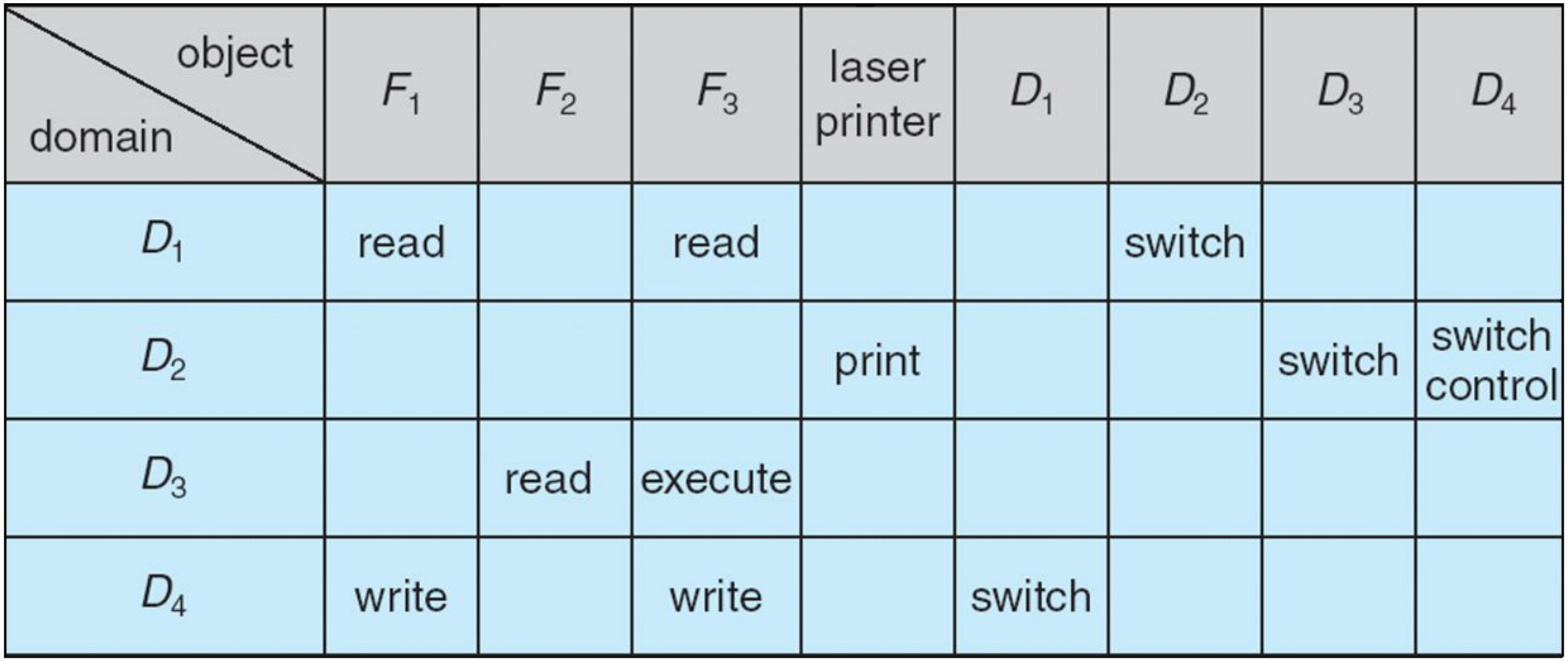
(a)



object domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

Modified Access Matrix of Figure B



object domain	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 control
D_3		read	execute					
D_4	write		write		switch			

Implementation of Access Matrix

Methods for implementing access matrix

- Global Table
- Access Lists for Objects
- Capability Lists for Domains
- A Lock-Key Mechanism

Global Table

- The simplest implementation of the access matrix is a global table consisting of a set of ordered triples<domain , object, rights-set>.
- Whenever an operation M is executed on an object Oj within domain Di, the global table is searched for a triple <Di,Oj,Rk>, with M belons to Rk.
- If this triple is found, the operation is allowed to continue; otherwise, an exception (or error) condition is raised .

Drawbacks:

- The table is usually large and thus cannot be kept in main memory, so additional I/0 is needed

Implementation of Access Matrix

- Each column = Access-control list for one object
  Defines who can perform what operation.

  Domain 1 = Read, Write
  Domain 2 = Read
  Domain 3 = Read

  □

- Each Row = Capability List (like a key)
  Fore each domain, what operations allowed on what objects.

  Object 1 – Read

  Object 4 – Read, Write, Execute

  Object 5 – Read, Write, Delete, Copy

- A Lock-Key Mechanism

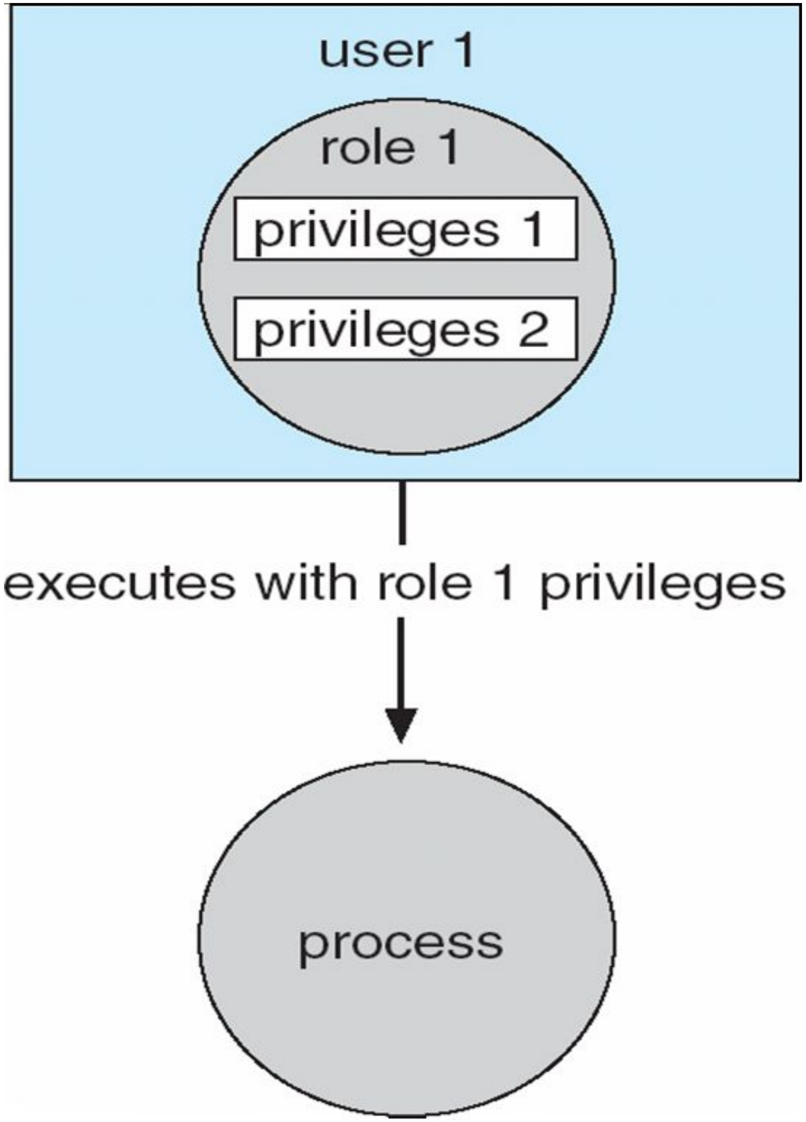
Each object has a list of unique bit patterns, called Locks. Similarly, each domain has a list of unique bit patterns, called keys.

A process executing in a domain can access an object only if that domain has a key that matches one of the locks of the object.

Access Control

- Protection can be applied to non-file resources
- Solaris 10 provides role-based access control (RBAC) to implement least privilege
  - Privilege is right to execute system call or use an option within a system call
  - Can be assigned to processes
  - Users assigned roles granting access to privileges and programs

Role-based Access Control in Solaris 10


user 1
role 1
privileges 1
privileges 2
executes with role 1 privileges
process