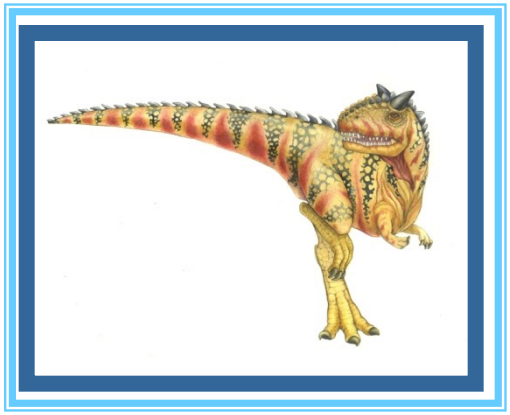
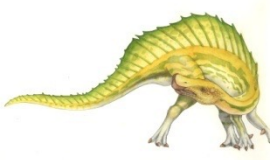
Chapter 14: Protection



Chapter 14: Protection

- Goals of Protection
- Principles of Protection
- Domain of Protection
- Access Matrix
- Implementation of Access Matrix
- Access Control
- Revocation of Access Rights
- Capability-Based Systems



Objectives

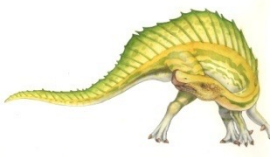
- Discuss the goals and principles of protection in a modern computer system
- Explain how protection domains combined with an access matrix are used to specify the resources a process may access
- Examine capability protection systems

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.

Principles of Protection

- Guiding principle – principle of least privilege
  - Programs, users and systems should be given just enough privileges to perform their tasks
  - 权利尽可能细化，只赋予用户所需要的那部分权利。

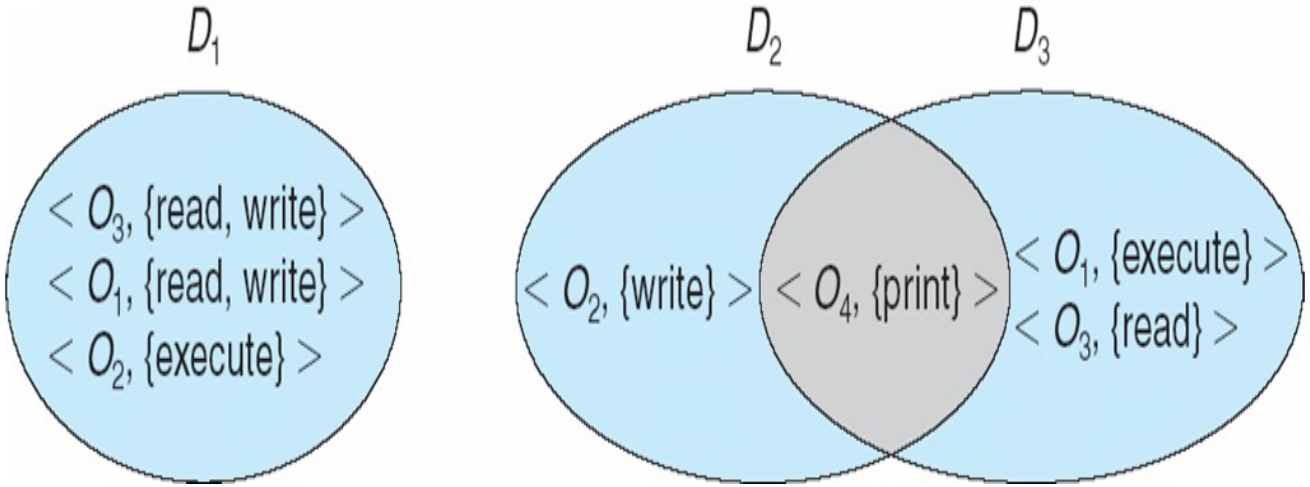


Domain Structure

- Access-right = <*object-name*, *rights-set*>

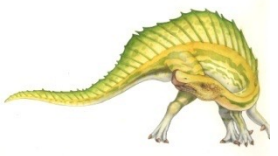
  where *rights-set* is a subset of all valid operations that can be performed on the object.

- Domain = set of access-rights



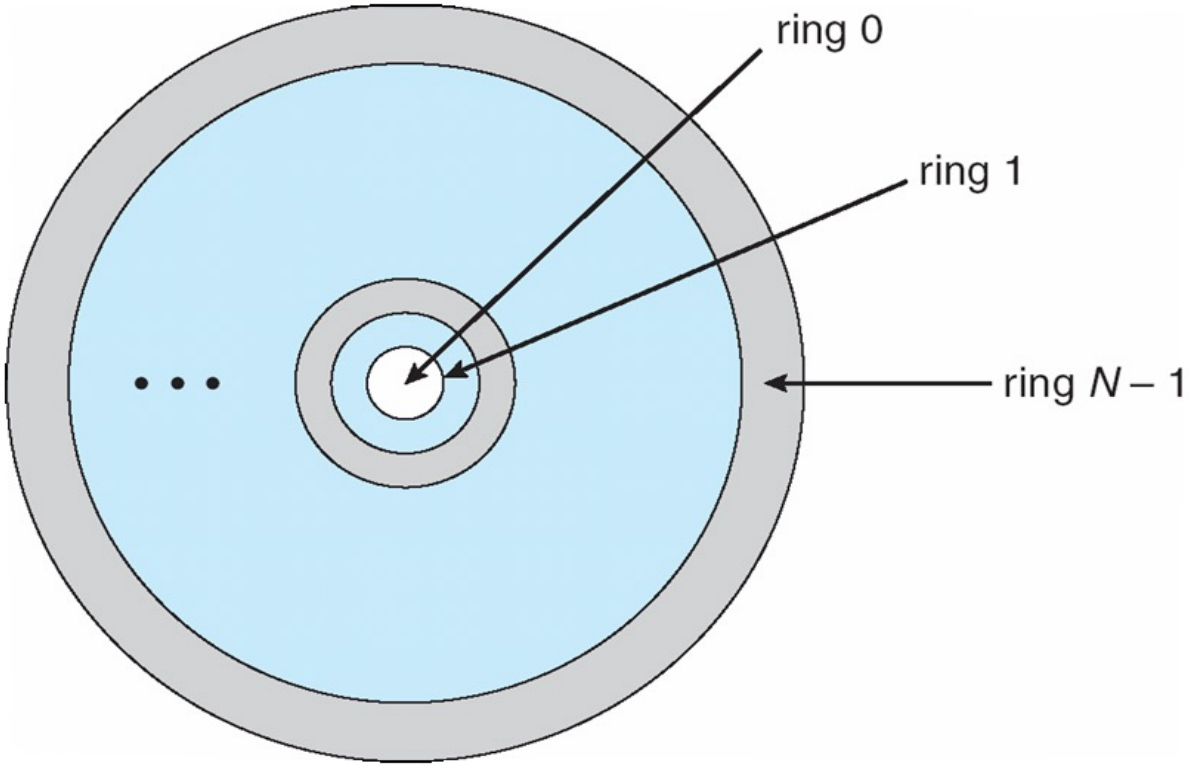
Domain Implementation (UNIX)

- System consists of 2 domains:
  - User
  - Supervisor
- UNIX
  - Domain = user-id
  - Domain switch accomplished via file system.
    - Each file has associated with it a domain bit (setuid bit).
    - 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.



Domain Implementation (MULTICS)

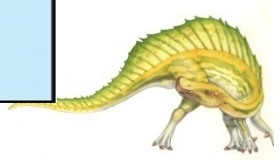
- MULTICS：the protection domains are organized hierarchically into a ring structure.
- Let D_i and D_j be any two domain rings. If $j < i \Rightarrow D_i \subseteq D_j$



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 $Object_j$

domain \ object	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:
    - *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
- 特殊权限： *owner*、 *copy*、 *control*、*transfer*

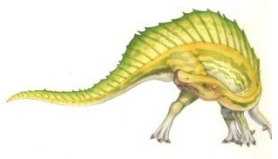
Access Matrix With *Owner* Rights

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)



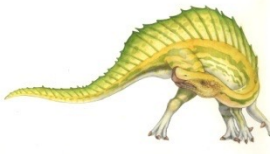
Access Matrix with *Copy* Rights

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

(a)

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

(b)



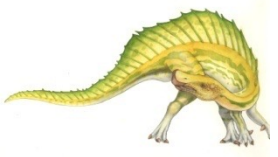
Access Matrix of Figure A With Domains as Objects

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			

域作为对象图

Modified Access Matrix of Figure A

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 control
D_3		read	execute					
D_4	write		write		switch			



Use of Access Matrix (Cont.)

- Access matrix design separates mechanism from policy.
  - Policy（What will be done）
    - User dictates policy.
    - Who can access what object and in what mode.
  - Mechanism（How someting will be done）
    - Operating system provides access-matrix + rules.
    - If ensures that the matrix is only manipulated by authorized agents and that rules are strictly enforced.

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

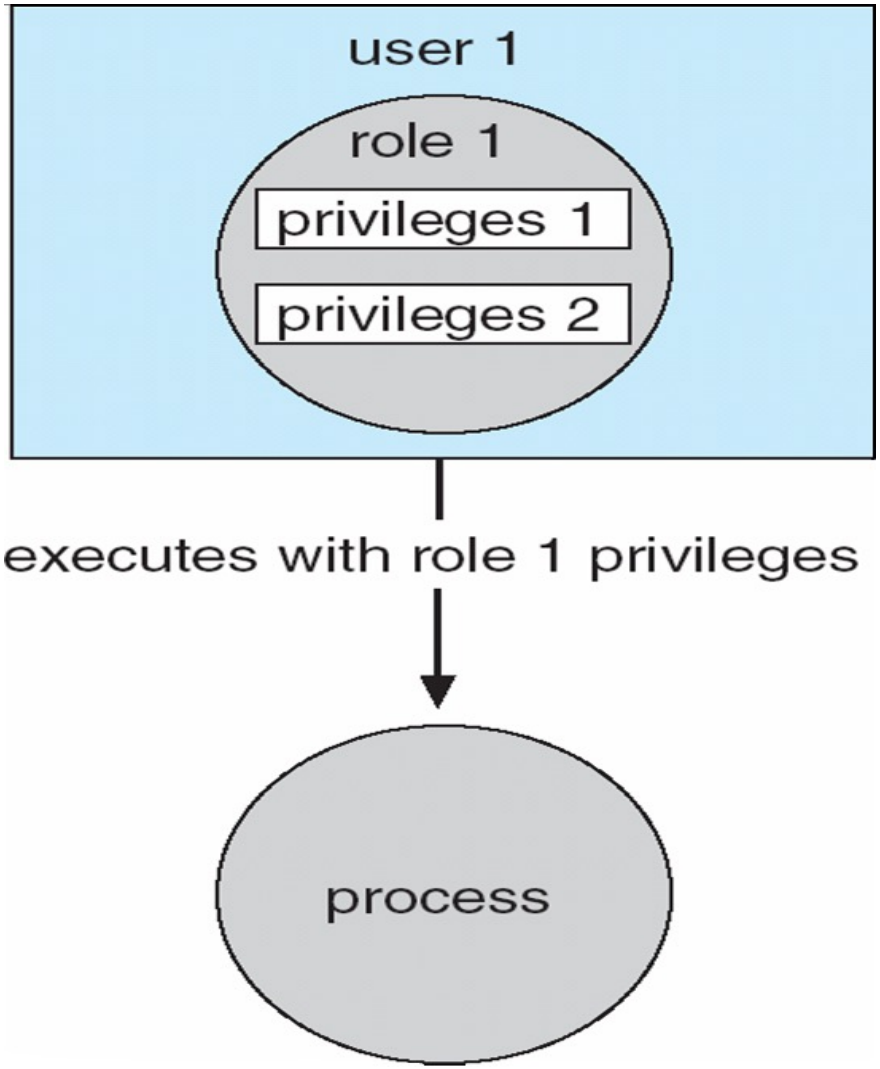
  Object 5 – Read, Write, Delete, Copy

Access Control

- Protection can be applied to non-file resources
- Solaris 10 provides **role-based access control** 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

在用户集合与权限集合之间建立一个角色集合，每一种角色对应一组相应的权限。用户被分配了角色后，就拥有此角色的操作权限。简化用户权限管理与系统开销。

Role-based Access Control in Solaris 10



End of Chapter 14