Protection

Giuseppe Anastasi
g.anastasi@iet.unipi.it
Pervasive Computing & Networking Lab. (PerLab)
Dept. of Information Engineering, University of Pisa

Based on original slides by Silberschatz, Galvin and Gagne

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

Overview

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

Concurrent Machine

- A machine with as many (virtual) processors as concurrent activities

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

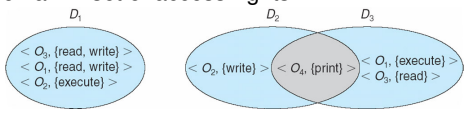
- Guiding principle – principle of least privilege
  - Programs, users and systems should be given just enough privileges to perform their tasks

Policies vs. Mechanisms

- Policies
  - User-specified rules
  - *Who* can access what object and in *what* mode
  - May change over time
- Mechanisms
  - *How* to enforce policies
  - Allow to support different policies

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

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

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	

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

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:
    - *switch* from domain D_i to D_j
    - *copy* op from O_i to O_j
    - *owner* of O_i
    - *control* – D_i can modify D_j access rights

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

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 With *Owner* Rights

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

(a)

domain \ object	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 *Control* Rights

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	read write		read write		switch			

Access Matrix With *Control* Rights

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			

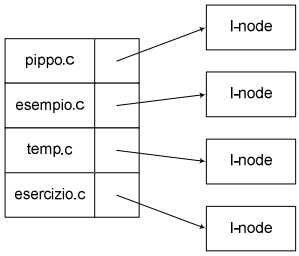
Modified Access Matrix of Figure B

File Protection in UNIX

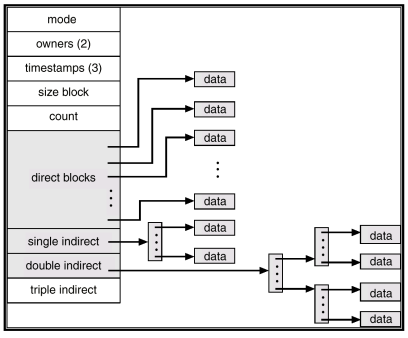
- Access Modes
  - read, write, execute
  - 1 bit for each mode
- Three different user classes

			RWX
a) **owner**	7	⇒	1 1 1
b) **group**	6	⇒	RWX 1 1 0
c) **others**	1	⇒	RWX 0 0 1

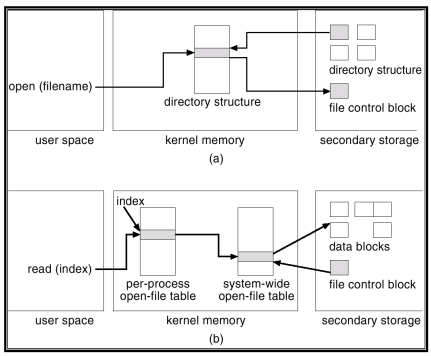
UNIX Directory Structure



UNIX I-node



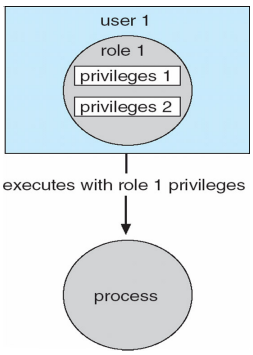
File Access in Unix



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



Questions?