

Today: Protection

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



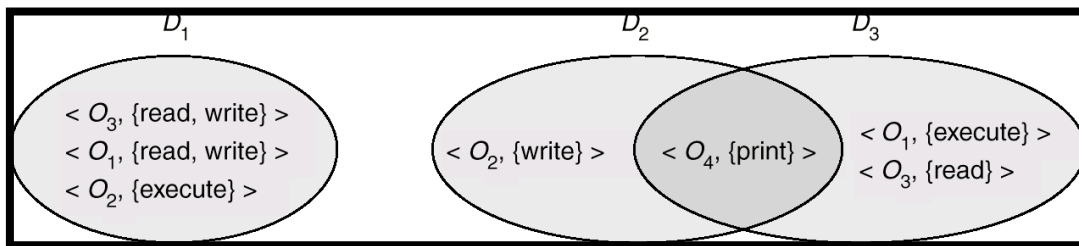
Protection

- 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

- Access-right = $\langle \text{object-name}, \text{rights-set} \rangle$
where *rights-set* is a subset of all valid operations that can be performed on the object.
- Domain = set of access-rights
 - associated with users, user groups and their processes



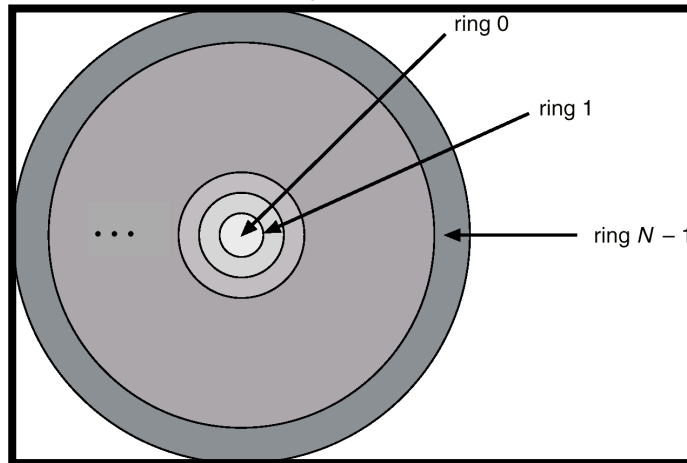
Domain Implementation (UNIX)

- 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)

- 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}



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

- 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*



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.



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)
For each domain, what operations allowed on what objects.
 - Object 1 – Read
 - Object 4 – Read, Write, Execute
 - Object 5 – Read, Write, Delete, Copy



Revocation of Access Rights

- *Access List* – Delete access rights from access list.
 - Simple
 - Immediate
- *Capability List* – Scheme required to locate capability in the system before capability can be revoked.



Capability-Based Systems

- Hydra
 - Fixed set of access rights known to and interpreted by the system.
 - Interpretation of user-defined rights performed solely by user's program; system provides access protection for use of these rights.
- Cambridge CAP System
 - Data capability - provides standard read, write, execute of individual storage segments associated with object.
 - Software capability -interpretation left to the subsystem, through its protected procedures.



Language-Based Protection

- Specification of protection in a programming language allows the high-level description of policies for the allocation and use of resources.
- Language implementation can provide software for protection enforcement when automatic hardware-supported checking is unavailable.
- Interpret protection specifications to generate calls on whatever protection system is provided by the hardware and the operating system.



Protection in Java 2

- Protection is handled by the Java Virtual Machine (JVM)
- A class is assigned a protection domain when it is loaded by the JVM.
- The protection domain indicates what operations the class can (and cannot) perform.
- If a library method is invoked that performs a privileged operation, the stack is inspected to ensure the operation can be performed by the library.



Course Wrap-up and Review

Final Exam covers:

- More emphasis on File & I/O systems and distributed systems
- Final is comprehensive



Course Overview

- Processes & Threads
- Memory
- I/O, file systems
- Networking, distributed systems

Hardware abstraction	Example OS Services	User abstraction
Processor	Process management, Scheduling, Traps, protection, accounting, synchronization	Process
Memory	Management, Protection, virtual memory	Address spaces
I/O devices	Concurrency with CPU, Interrupt handling	Terminal, mouse, printer, system calls
File System	File management, Persistence	Files
Distributed systems	Networking, security, distributed file system	Remote procedure calls, network file system



Sermons in Computer Science

- Simplicity
- Performance
- Programming as Craft
- Information is Property
- Stay Broad

- Ack: Tom Anderson, U. Washington and M. Dahlin U. Texas

