Today: Protection

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

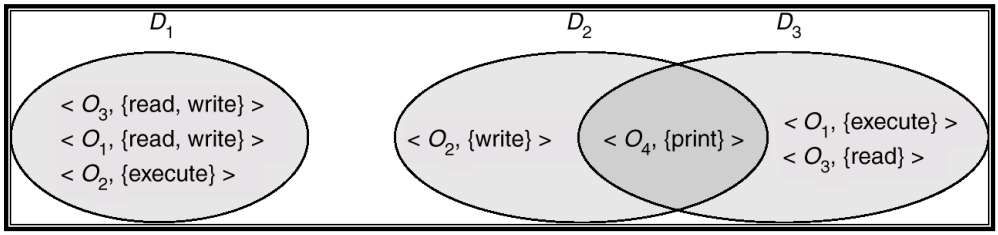
Protection

- 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 = <*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



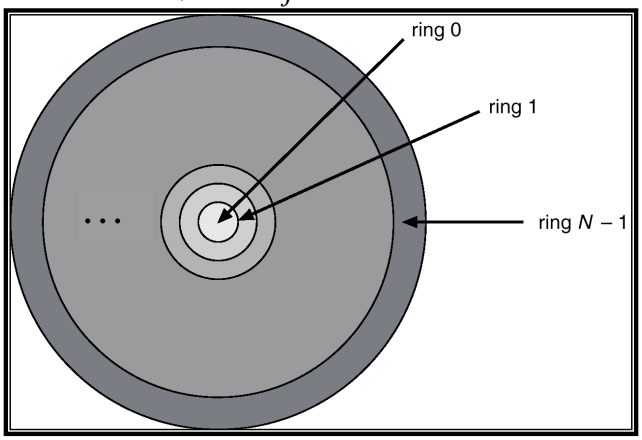
Domain Implementation (UNIX)

- 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)

- Let D_i and D_j be any two domain rings.
- If $j < I \Rightarrow D_i \subseteq D_j$



Multics Rings

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	

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:

- 50% of the exam is on I/O systems and distributed systems
- 50% of the exam is on the rest of the course

Highlights of Process Management

1. What is a context switch? What happens during a context switch? What causes a context switch to occur?
2. What is the difference between a process and a thread?
3. What are FCFS, Round Robin, SJF, and Multilevel Feedback Queue algorithms?
4. What is an I/O bound process? What is a CPU bound process? Is there any reason to treat them differently for scheduling purposes?
5. What is a semaphore? What are the three things a semaphore can be used for?
6. What is a monitor? What is a condition variable?
7. What is busy waiting?
8. What are the four necessary conditions for deadlock to occur?
9. What is the difference between deadlock detection and deadlock prevention?
10. After detecting deadlock, what options are conceivable for recovering from deadlock?

Highlights of Memory and I/O Management

1. What is virtual memory and why do we use it?
2. What is paging, a page?
3. What does the OS store in the page table?
4. What is a TLB? How is one used?
5. What is a page fault, how does the OS know it needs to take one, and what does the OS do when a page fault occurs?
6. Page replacement algorithms: FIFO, MIN, LRU, Second chance. For each understand how they work, advantages and disadvantages.
7. How does the OS communicate with I/O devices?
8. What are I/O buffers used for?
9. What are I/O caches used for? How do they affect reading and writing to I/O devices?
10. What is seek time?
11. What is rotational latency?
12. What is transfer time?
13. Disk scheduling algorithms: FIFO, SSTF, SCAN, C-SCAN. How do they work, advantages and disadvantages.

Memory Management

Topics you should understand:

1. What is virtual memory and why do we use it?
2. Memory allocation strategies:
   - Contiguous allocation (first-fit and best-fit algorithms)
   - Paging
   - Segmentation
   - Paged segmentation

Memory Management (cont.)

For each strategy, understand these concepts:

- Address translation
- Hardware support required
- Coping with fragmentation
- Ability to grow processes
- Ability to share memory with other processes
- Ability to move processes
- Memory protection
- What needs to happen on a context switch to support memory management

File Systems

Topics you should understand:

1. What is a file, a file type?
2. What types of access are typical for files?
3. What does the OS do on a file open, file close?
4. What is a directory?
5. What is a link?
6. What happens if the directory structure is a graph?
7. How does an OS support multiple users of shared files?
8. Strategies for laying files out on disk. Advantages and disadvantages.
   a) Contiguous allocation
   b) Linked
   c) Indexed

I/O Systems

Topics you should understand

- Direct Memory Access
- Polling and Interrupts
- Caching and Buffering

Distributed Systems

1. What is the difference between a distributed system and a parallel system?
2. What advantages do distributed systems have over isolated systems?
3. What advantages do isolated systems have over distributed systems?

Networks

1. What is a LAN?
2. What is a WAN?
3. What are common network topologies? Which are most suitable to WANs? Which to LANs?
4. How do node failures affect the different network topologies?
5. What are the expected communication costs for the different network topologies?
6. What are packets?
7. What is a network protocol stack? What is TCP/IP?

Distributed sharing

1. What is data migration? When would you use it?
2. What is computation migration? When would you use it?
3. What is job migration? When would you use it?

Remote Procedure Call

1. What is RPC?
2. How does RPC differ from normal procedure call?
3. What extra computation is required to do RPC instead of a normal procedure call?
4. Would you ever use RPC to communicate between two processes on the same machine?

Distributed file systems

1. What are location transparent names?
2. What are location independent names?
3. What does it mean to say that a distributed file system has a single (global) namespace?
4. What is a cache?
5. What are the advantages of using a cache in a distributed file system? What are the disadvantages?
6. What are the advantages and disadvantages of write-back and write-through caches?

Protection

1. What is protection and how does it differ from security?
2. What is a domain?
3. What is a domain access matrix? How are these implemented in actual operating systems?
4. How can entries in an access matrix be modified? What is a domain switch and why is it needed?

General Skills

- You should have a good sense of how the pieces fit together and how changes in one part of the OS might impact another.
- You will **not** be asked to read or write Java code.
- You will **not** be asked detailed questions about any specific operating system such as Unix, Windows NT.

Sermons in Computer Science

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