# Protection

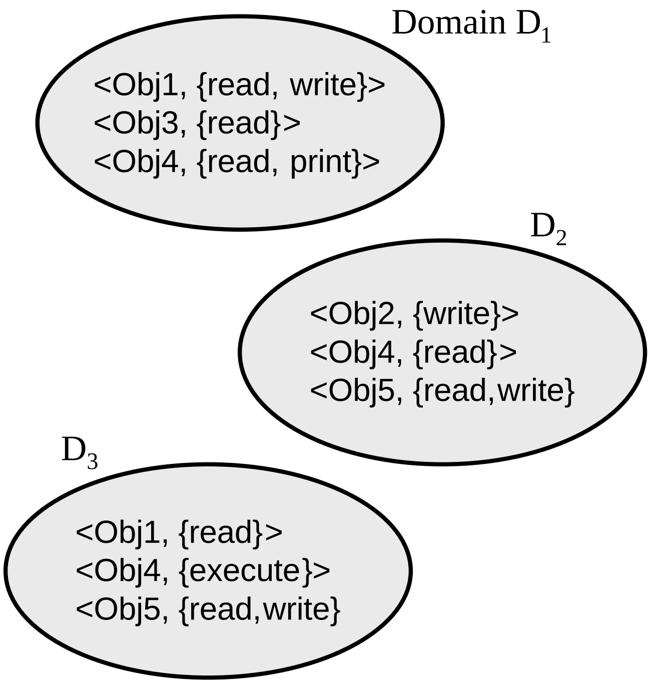
- What is protection (vs. security)?
- Protection domains
- Describing protection: access matrix
- Implementing protection
  - » Implementing an access matrix
  - » Using capabilities for protection
  - » Granting & revoking access rights
- Implementing protection in computer languages

# What is Protection?

- Computer system consists of a collection of hardware and software objects
  - » Each object has a unique name
  - » Each object may be accessed through a set of operations, potentially different for different objects
- Problems
  - » Which operations are allowed on various objects?
  - » Who may perform the operations?
- Protection defines the relationship between things that may perform operations (e.g., processes) and objects that may have operations done on them
- Security enforces the policies that protection defines

# Protection Domains

- Access right is <object-name, set-of-rights>
  - Set-of-rights is a subset of all valid operations that can be performed on the object
  - Identical rights may occur in different domains
- A protection domain is a collection of access rights for one or more objects



Domain $D_1$

<Obj1, {read, write}>
<Obj3, {read}>
<Obj4, {read, print}>

$D_2$

<Obj2, {write}>
<Obj4, {read}>
<Obj5, {read,write}

$D_3$

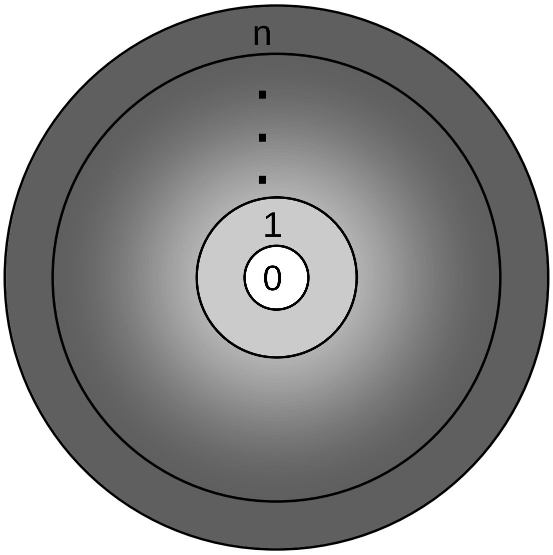
<Obj1, {read}>
<Obj4, {execute}>
<Obj5, {read,write}

# Domain Examples

- Simple personal computers: 2 domains
  - User: ability to use only user instructions
  - Supervisor: can pretty much do anything
- Unix: many domains
  - Domain determined by user ID
    - Process inherits domain of user that created it
    - Process can inherit the domain of the *program's* owner if the setuid bit is set on the program
  - Root user has all rights for all objects
  - Domain switch done through OS
    - First, switch to supervisor domain via trap instruction
    - When OS is done, switch back to user domain using a "return from interrupt" instruction
    - OS switches domain if new process is different from old one

# Domains in MULTICS

- System composed of protection “rings”
- Higher ring number == less protection
  - » OS in low-numbered rings
  - » User programs in high-numbered rings
  - » Transfer to lower-numbered ring by “gates” at specific points
- For rings *j*, *i*: if *j* < *i*, $D_j$ is a superset of $D_i$



# Access Matrix

- Rows of matrix are domains
- Columns are domains & objects
  - » Domains can have rights over other domains
  - » Domains are themselves objects
- Entries are operations permitted on the object within that domain

| Objects ⇨<br>Domains ⇩ | $F_1$ | $F_2$ | $F_3$ | $D_1$ |
|---|---|---|---|---|
| $D_1$ | RX | RW | | M |
| $D_2$ | RP | | W | |
| $D_3$ | | X | R | M |

R = read
W = write
X = execute
P = print
M = modify

# Using an Access Matrix

- When an process in $D_i$ tries to do an operation on object $O_j$
  - » Look up row and column in the access matrix
  - » Check to see that the operation is listed there
  - » If not allowed, don't permit the operation
  - » Process may be in more than one domain...
- Access matrix can be updated dynamically
  - » Rights in matrix can include
    - – Ownership of objects
    - – Granting and revoking other rights
    - – Switching from one domain to another
  - » Allows lots of flexibility

# Mechanism vs. Policy

- Access matrix separates mechanism from policy
  - » Single mechanism (access matrix) supports many policies
  - » Policies need not be hard-coded into the OS
- Mechanism
  - » Provision of methods to specify access rules
    - – Access matrix
    - – Rules for manipulating access matrix
    - – Enforcement of access to and modification of access matrix
  - » Enforcement of rules contained in matrix
- Policy
  - » Decisions about which operations are allowed in which domains
    - – Which domains have rights over an object?
    - – Which rights are possible for any object?
  - » Policy is set by the user(s) of the computer system

# Implementing an Access Matrix

- Access matrix is usually a sparse matrix
  - » Most entries are empty
  - » Rows or columns can be represented as lists
- Access control list (ACL): one per object
  - » Each ACL lists permitted domains & operations for an object
  - » Example: Object1 => {D1:RW}, {D2,RX}, {D5:RWXP}
  - » Domains with no rights are omitted from the list
- Capability list: one per domain
  - » Lists the objects and permitted operations on those objects
  - » Functions like a key ring
  - » Example: Domain1 => {F1:RW}, {F4:X}, {F7:RWD}
  - » Objects with no rights are omitted from the list

# Securing the Access Matrix

- OS must ensure that the access matrix isn't modified (or even accessed) in an unauthorized way
- Access control lists
  - » Reading or modifying the ACL is a system call
  - » OS makes sure the desired operation is allowed
- Capability lists
  - » Can be handled the same way as ACLs: reading and modification done by OS
  - » Can be handed to processes and verified cryptographically later on (more on this next week)
  - » May be better for widely distributed systems where capabilities can't be centrally checked

# Revoking Access Rights

- Granting rights is easy: rights on objects can include the ability to give additional rights to domains
  - » Simply add an entry to the access matrix
  - » Domain can do something it couldn't do before
- Revoking rights is more difficult
- Access control lists
  - » Remove access rights from the list
  - » Simple & immediate: rights are checked at object
- Capability lists
  - » Must be able to find the capability to delete it
  - » May not be difficult if OS keeps capability lists: search process list for the keys, and delete them
  - » Can be very difficult if process holds the list: must invalidate keys at the object (equivalent to changing the lock)

# Access Control List Systems

- Unix file system
  - » Access list for each file has exactly three domains on it
    - – User (owner)
    - – Group
    - – Others
  - » Rights include read, write, execute: interpreted differently for directories and files
- AFS
  - » Access lists only apply to directories: files inherit rights from the directory they're in
  - » Access list may have many entries on it with possible rights:
    - – read, write, lock (for files in the directory)
    - – lookup, insert, delete (for the directories themselves),
    - – administer (ability to add or remove rights from the ACL)

# Capability-Based Systems

- Hydra
  - » Fixed set of access rights the system knows about & uses
  - » User programs can define rights and have them maintained by the OS: interpretation left up to user programs
- Cambridge CAP
  - » Data capabilities include read, write, execute: managed by OS
  - » Software capabilities: definition and interpretation left to subsystem through protected procedures
- Kerberos (combines ACL & capabilities)
  - » Users get one or more tickets (capabilities)
  - » Capabilities identify user to system and allow lookup on ACL for permissions

# Language-Based Protection

- Protection specified in high-level language
  - » Enforced by the compiler or interpreter
  - » Policies can be specified for user objects
- Language can provide protection when hardware or OS doesn't support it
  - » Example: Java (especially sandbox)
  - » Example: SafeTcl
  - » Programs written in these languages can't cause problems because the language and interpreter won't let them
- Language can convert protections specified in the language into those handled by the OS