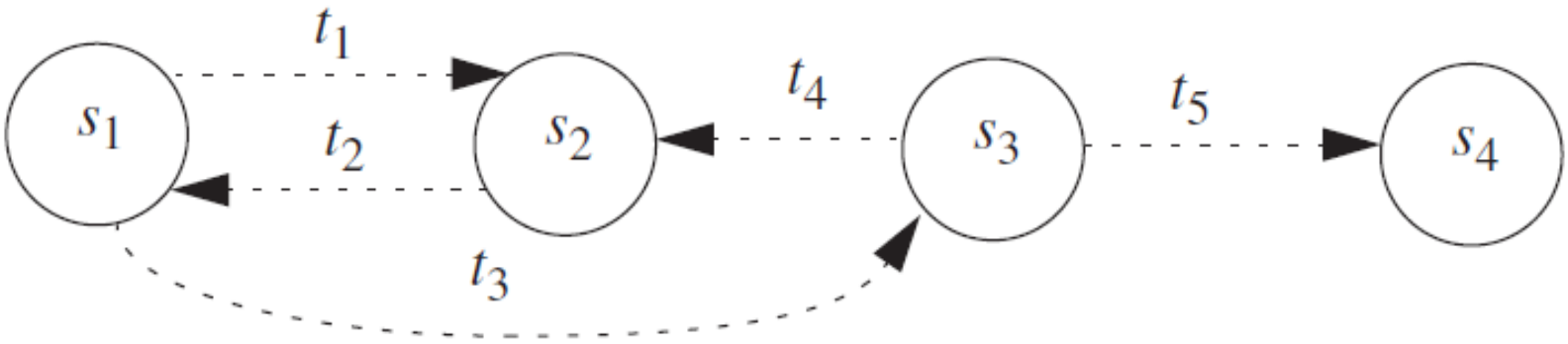
Security Policy

- What is a security policy?
  - Defines what it means for a system to be secure
- Formally: Partition system into
  - Secure (authorized) states
  - Non-secure (unauthorized) states
- Secure system:
  - Starts in authorized state
  - Can’t enter unauthorized state

Secure System - Example

- Is this Finite State Machine Secure?
  - *A* and *B* are authorized states
  - *B* is start state
  - *C* is start state
  - *A*, *B*, and *C* are authorized states

Secure System - Example



- Is this Finite State Machine Secure?
  - *S1* and *S2* are authorized states
  - *S3 and S4 are unauthorized states*

Additional Definitions:

- Breach of security
  - –Transition causing system to enter unauthorized state
- Let *X* be a set of entities, *I* be information.
  - –*I* has **confidentiality** with respect to *X* if no member of *X* can obtain information on *I*
    - All entities not authorized to have such access make up the set $\overline{X}$.
  - –*I* has **integrity** with respect to *X* if all members of *X* trust *I*
  - –*I* has **availability** with respect to *X* if all members of *X* can access *I*
- Security Policy defines all of the above
  - –Now just need to define obtain, trust, access

Confidentiality Policy

- What does —obtain information mean?
- Formally: information flow
  - Transfer of rights
  - Transfer of information without transfer of rights
- Model often depends on trust
  - Parts of system where information could flow
  - Trusted entity must participate to enable flow
- Highly developed in Military/Government

Integrity Policy

- Defines how information can be altered
  - Entities allowed to alter data
  - Conditions under which data can be altered
  - Limits to change of data
- Examples:
  - Purchase over $1000 requires signature
  - Check over $10,000 must be signed by two officers
    - Separation of duties
- Highly developed in commercial world

Availability Policy

- Defines what it means for information to be accessible
  - Time limits (quality of service)
  - Access methods
    - On-line access vs. telephone vs. mail
- Integrity and availability may interrelate
  - Fast old copy vs. slow current version

Access Control Mechanism

- It is typically a software system implementing the access control function
- It is usually part of other systems
- The access control mechanism uses some access control policies to decide whether to grant or deny a subject access to a requested resource
- We will refer to an *access control system* as system comprising an access control mechanism and all information required to take access control decisions (for example, access permissions)

Security Mechanism

- Policy describes what is allowed
- Mechanism enforces (part of) policy
  - The two need not be the same!
- Example Policy: Students should not copy homework
  - Mechanism: Disallow access to files owned by other users
- Does mechanism enforce policy?
  - Is mechanism too strict?

Example

A site’s security policy states that information relating to a particular product is proprietary and is not to leave the control of the company. The company stores its backup tapes in a vault in the town’s bank (this is common practice in case the computer installation is completely destroyed). The company must ensure that only authorized employees have access to the backup tapes even when the tapes are stored off-site; hence, the bank’s controls on access to the vault.

Example

The UNIX operating system, initially developed for a small research group, had mechanisms sufficient to prevent users from accidentally damaging one another's files; for example, the user *ken* could not delete the user *dmr*'s files (unless *dmr* had set the files and the containing directories appropriately). The implied security policy for this friendly environment was "do not delete or corrupt another's files, and any file not protected may be read."

Security Model

- Security Policy: What is/isn't authorized
- Problem: Policy specification often informal
  - Implicit vs. Explicit
  - Ambiguity
- Security Model: Model that represents a particular policy (policies)
  - Model must be explicit, unambiguous

Access Control - basic concepts

- An access control system regulates the operations that can be executed on data and resources to be protected
- Its goal is to control operations executed by subjects in order to prevent actions that could damage data and resources
- Access control is typically provided as part of the operating system and of the database management system (DBMS)

Access Control - basic concepts

- Subjects request actions on objects

  - Alice wants to read file.txt
  - Bob wants to update account balance
  - Process wants to open a socket

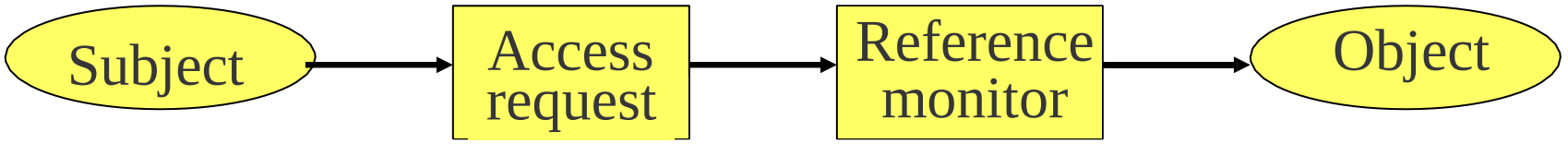
Access Control - basic concepts

- Traditional definition: AC = authentication + authorization

- Authentication = verifying subject identity
  (implies identification of the subject)
- Authorization = checking that the subject has the right to request the action on the subject

- Authorization without identification and authentication?

Access Control - basic concepts



- The very nature of access control suggests that there is an *active* subject *requiring access* to a passive *object* to perform some specific *access operation.*
- A *reference monitor* grants or denies access
- This fundamental and simple notion of access control is due to Lampson

---

B. Lampson. Protection. *ACM Operating System Reviews*, **8**, 1974.

Access Control - basic concepts

- Trusted computing base (TCB) = all system components that need to be trusted to implement access control
  - Includes hardware, firmware, OS, other software
  - Security kernel = implementation of reference monitor in an OS
  - Isolation of subjects from each other is needed to prevent them from circumventing the AC policy

Object

- Anything that holds data, such as relations, directories, interprocess messages, network packets, I/O devices, or physical media
- We often refer to objects, controlled by the access control system, as *protection objects*
- Note that not all resources managed by a system need to be protected

Subject

- An abstraction of any active entity that performs computation in the system
- Subjects can be classified into:
  - *users* -- single individuals connecting to the system
  - *groups* -- sets of users
  - *roles* -- named collections of privileges / functional entities within the organization
  - *processes* -- executing programs on behalf of users
- Relations may exist among the various types of subject

Access Operations - Access Modes

- Operations that a subject can exercise on the protected objects in the system
- Each type of operation corresponds to an *access mode*
- The basic idea is that several different types of operation may be executed on a given type of object; the access control system must be able to control the specific type of operation
- The most simple example of access modes is:
  - read look at the contents of an object
  - write change the contents of an object
- In reality, there is a large variety of access modes
- The access modes supported by an access control mechanism depend on the resources to be protected (read, write, execute, select, insert, update, delete, …)
- Often an access control system uses modes with the same name for different types of object; the same mode can correspond to different operations when applied to different objects

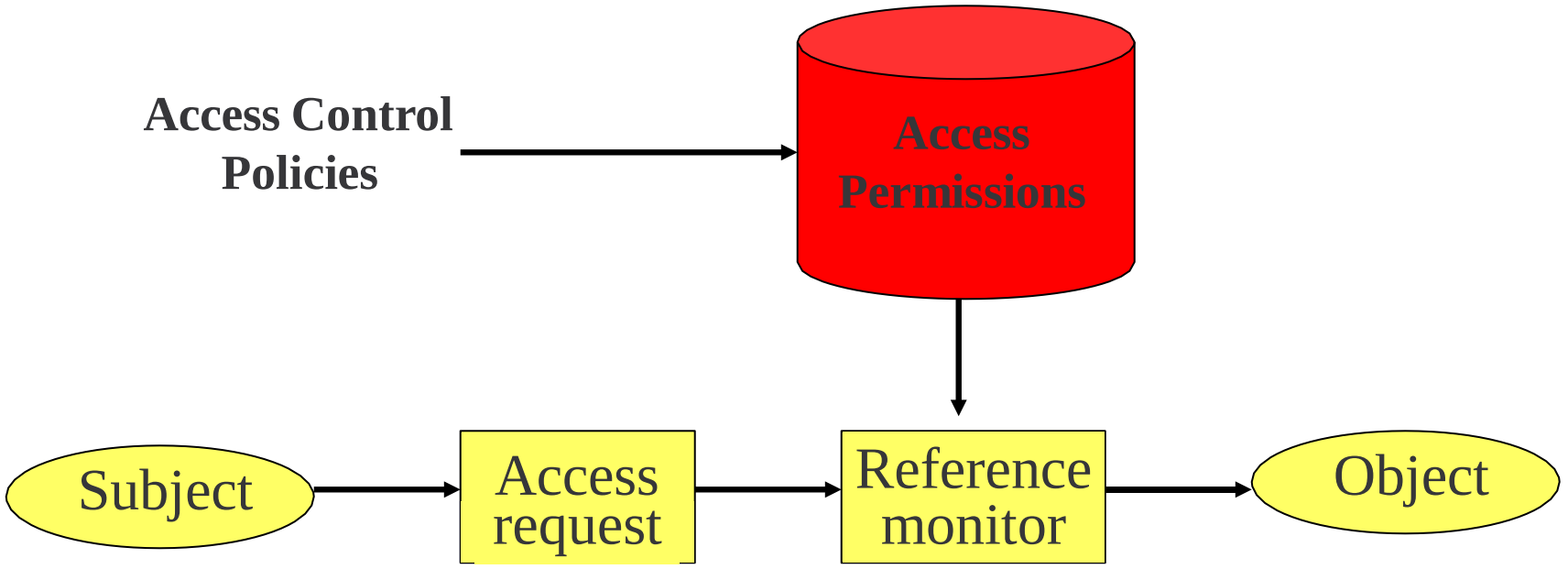
Access Operations - Access Modes
An example

- Unix operating system
  - Access modes defined for files
    - read: reading from a file
    - write: writing to a file
    - execute: executing a (program) file
  - Access models defined for directories
    - read: list a directory contents
    - write: create or rename a file in a directory
    - execute: search a directory

Access Operations
Access Permissions and Attributes

- How does the reference monitor decides whether to give access or not?
- Main approaches:
  - It uses *access permissions*
    - Typical of discretionary access control (DAC) models
  - It uses information (often referred to as *attributes*) concerning subjects and objects
    - Typical of multilevel access control (MAC) models
- More innovative approaches have been developed where access permissions can be also expressed in terms of object and subject attributes and even context parameters

Access Operations
Access Permissions



Access Permissions

- Access permissions, also called *authorizations*, are expressed in terms of subjects, objects, and access modes
- From a conceptual point of view an access permission is a tuple <*s, o, a*> where
  - *s* is a subject
  - *o* is an object
  - *a* is an access mode

  It states that subject *s* has the permission to execute operation *a* on object *o*

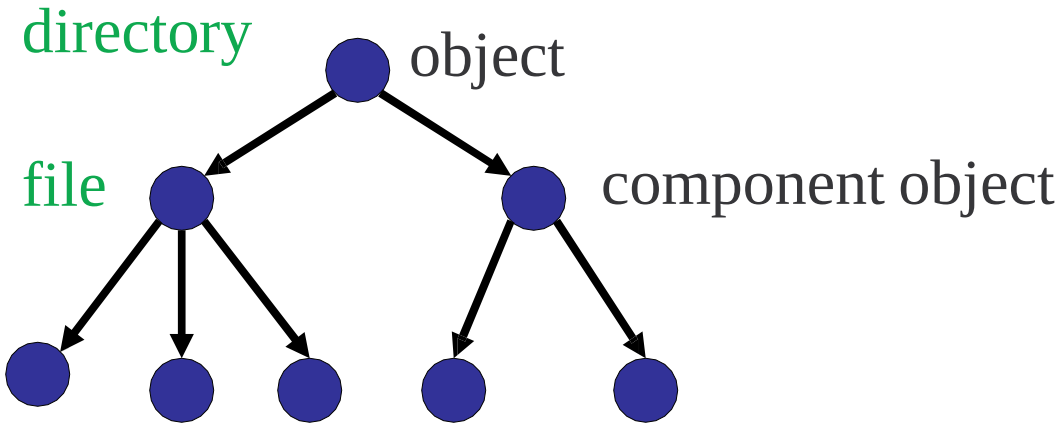
  We also say that *s* has access right *a* on object *o*
- Example:the access permission <Bob, Read, F1> states that Bob has the permission to read file F1

Access Permissions

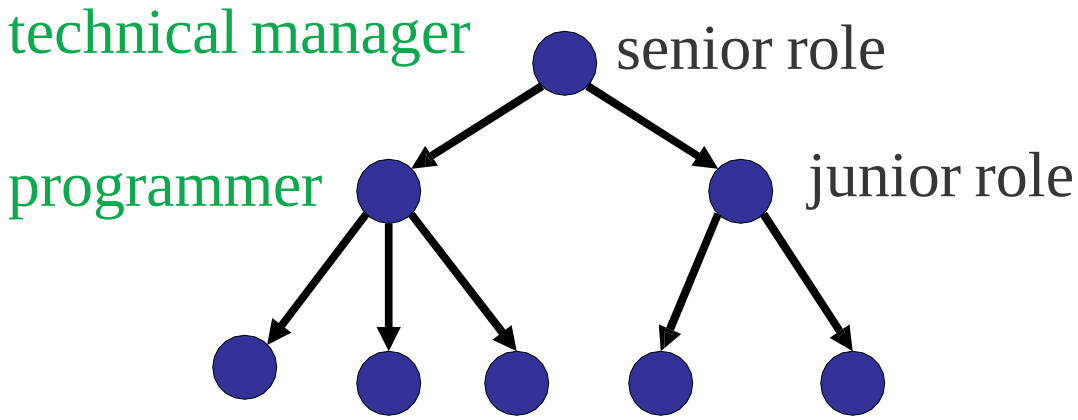
- Subjects, objects, and access modes can be organized into hierarchies
- The semantics of the hierarchy depends on the domain
- The use of hierarchies has two important advantages:
  - It reduces the number of permissions that need to be entered into the access control system, thus reducing administration costs
  - Combined with negative authorizations (to be discussed later on), it supports the specification of exceptions

Object Hierarchy

PART-OF

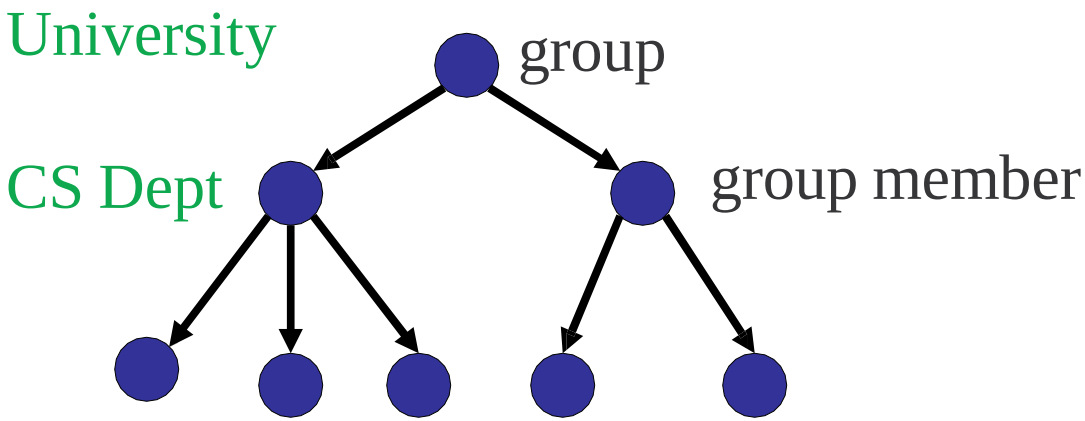


Role Hierarchy



Group Hierarchy

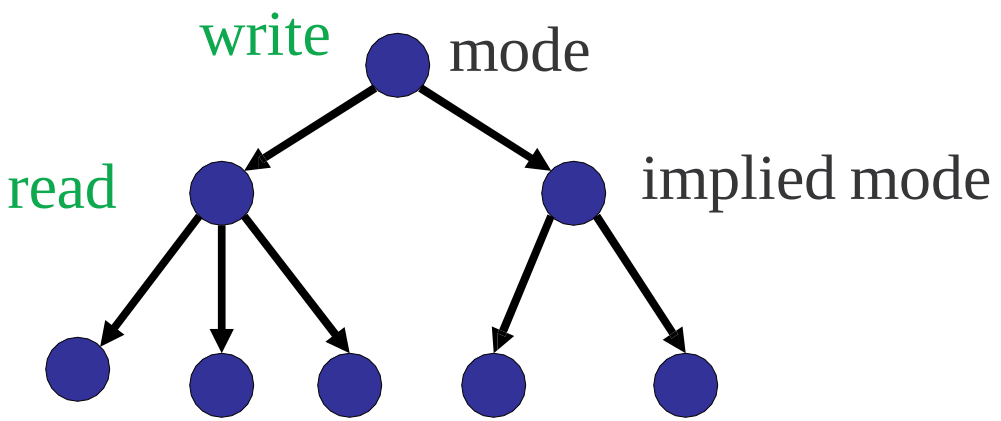
GROUP MEMBERSHIP



Suppose that the group CS department has 200 members and the University group 5000 members; suppose we have the policy that the department calendar can be read to all members of the University and written only by the members of CS; these policies can be encoded into two access permissions of the form:
<University, calendar, Read> <CS Dept, calendar, Write>

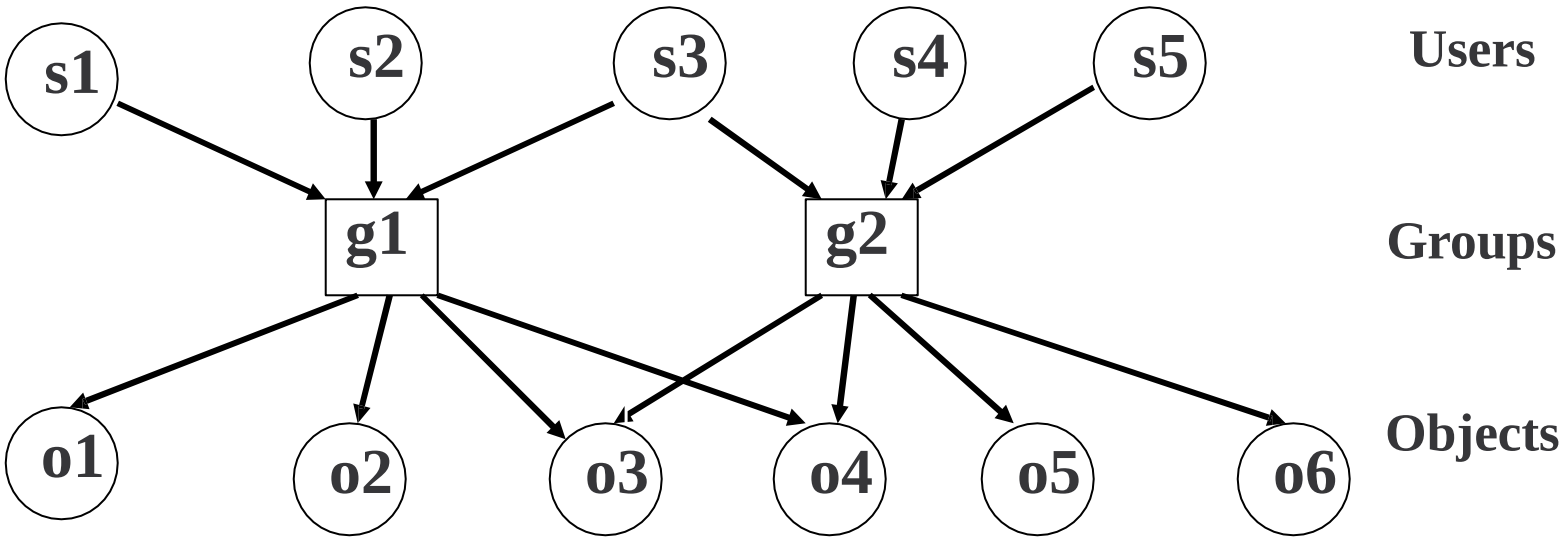
Access Mode Hierarchy

SUBSUMPTION



Groups and Negative Permissions

- Groups can be seen as an intermediate level between users and objects
- An example of an ideal world where all access permissions are mediated by groups



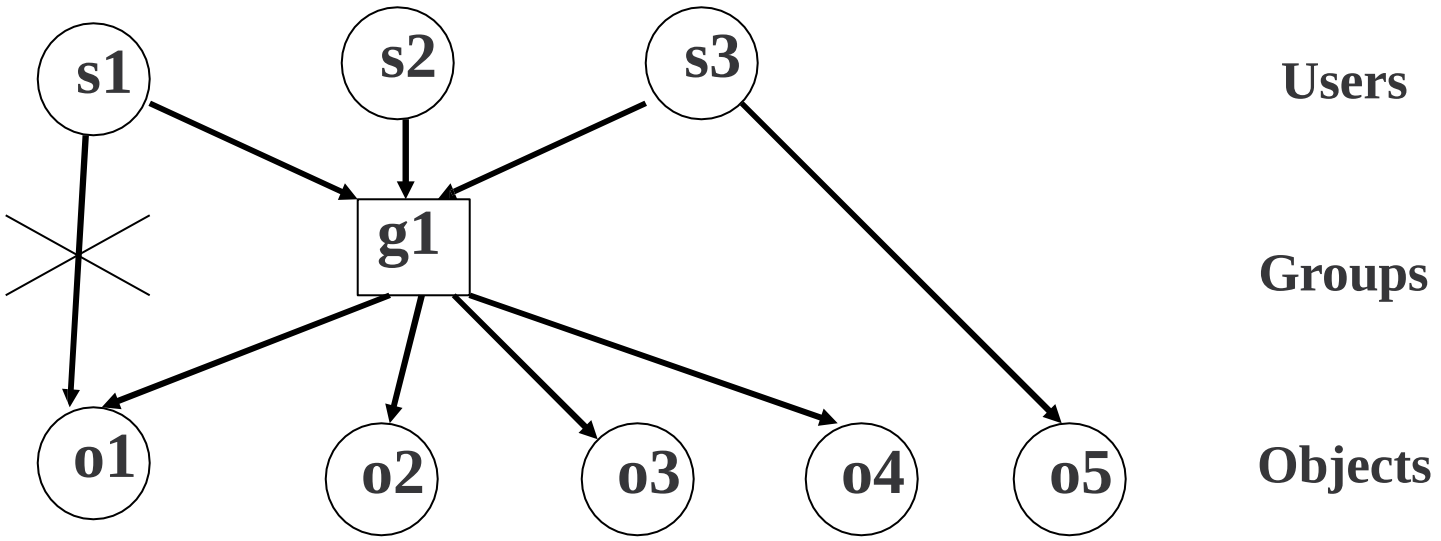
Groups and Negative Permissions

- Often access control policies have special cases where is proves convenient to give some user a permission for an object directly or *deny* a user a permission that it would normally derive from its membership in some group
- A *negative* permission specifies an operation that a subject is not allowed to perform
- Representing negative permissions requires extending our simple tuple model with an additional component:

  <*s, o, a, sign*> where *sign* Î{+, -}

Groups and Negative Permissions

An example in which not all access permissions are mediated through groups



Ownership and Administration

- A key question when dealing with access control is who specifies which subjects can access which objects for which operations
- In the case of permissions, this means specifying which are the subjects that can enter permissions

Ownership and Administration
Two basic options

- *Discretionary* approach
  - the owner of a resource decrees who is allowed to have access
  - But then: who is the owner of a resource?

- *Mandatory* approach
  - a system-wide policy decrees who is allowed to have access

Access Control Structures

The most well known access control structures for DAC models are based on the notion of *Access Control Matrix.* Let:

- *S* be a set of subjects
- *O* be a set of objects
- *A* be a set of access modes

An access control matrix *M* on *S, O,* and *A* is defined as

$$M = (M_{so})_{s \in S,\, o \in O} \text{ with } M_{so} \subset A$$

The entry M_{so} specifies the set of access operations subject *s* can perform on object *o.*

Access Control Structures
Example

	bill.doc	edit.exe	fun.dir
Alice	-	{execute}	{execute, read}
Bill	{read, write}	-	{execute, read, write}

Access Control Structures
Access Control Lists and Capabilities

- Directly implementing access control matrices is quite inefficient, because in most cases these matrices are sparse
- Therefore two main implementations have been developed
  - Access control lists
    - Used in DBMS and Operating Systems
  - Capabilities

Basic Operations in Access Control

- *Grant* permissions
  - Inserting values in the matrix's entries
- *Revoke* permissions
  - Remove values from the matrix's entries
- *Check* permissions
  - Verifying whether the entry related to a subject *s* and an object *o* contains a given access mode

Access Control List

- ACL = list of the access rights associated with an object
- ACLs are a practical way to represent the AC matrix: one column of the matrix is stored for each object
  - file1.txt ACL:
    Alice: { read, write }; Bob: { read };
    Carol: { read, write }; David: { append }.
  - file2.txt ACL:
    Alice: { write }; Bob: { read }.
  - file3.txt ACL:
    David: { open, read, write, close }.

Access Control List

ACL = list of the access rights associated with an object

- ACL examples:
  - Windows and Mac file systems

Capability

- Capability = an access right associated with the subject
- Capabilities are another way to represent the AC matrix: one row is stored for each subject

  – Alice's capabilities:
  file1.txt: { read, write }; file2.txt: { write }.
  – Bob's capabilities:
  file1.txt: { read }; file2.txt: { read }.
  – Carol's capabilities:
  file1.txt: { read, write }.
  – David's capabilities:
  file1.txt: { append }; file3.txt: { open, read, write, close }.

Capability

Capability = an access right associated with the subject

- Examples of capabilities:
  - Mobile app permission

Access Control Models

Two main categories:

- Discretionary Access Control Models (DAC)
  - Definition: If an individual user can set an access control mechanism to allow or deny access to an object, that mechanism is a *discretionary access control (DAC*), also called an *identity-based access control* (*IBAC*).
- Mandatory Access Control Models (MAC)
  - Definition: When a system mechanism controls access to an object and an individual user cannot alter that access, the control is a *mandatory access control* (MAC) [, occasionally called a *rule-based access control*.]

Access Control Models

- Other models:
  - The Chinese Wall Model – it combines elements of DAC and MAC
  - RBAC Model – it is a DAC model; however, it is sometimes considered a policy-neutral model
  - The Biba Model – relevant for integrity
  - The Information-Flow model – generalizes the ideas underlying MAC

DAC

- DAC policies govern the access of subjects to objects on the basis of subjects' identity, objects' identity and permissions
- When an access request is submitted to the system, the access control mechanism verifies whether there is a permission authorizing the access
- Such mechanisms are discretionary in that they allow subjects to grant other subjects authorization to access their objects at their discretion
- Owner determines access rights
- Typically identity-based access control: Owner specifies other users who have access

DAC

- Data owner, usually a user, sets the access rights
- Subjects can share their access rights to others
  - File owner or creator decides who can access it
  - User or process that can read a secret file can also copy and share it
- Typical in commercial and consumer systems
- Commonly implemented with ACLs or capabilities

- Note: There may be a policy against sharing, and access may be audited to detect violations, but the policy is not enforced technically

DAC

- DAC matches our everyday experience:
  - Paper documents can be copied and shared
  - Person with a key can open the door to others; door keys can be shared and copied
  - Telling your friends a secret, hoping that they do not tell it to anyone else, does not prevent them

DAC

- Advantages:
  - Flexibility in terms of policy specification
  - Supported by all OS and DBMS
- Drawbacks:
  - No information flow control (Trojan Horses attacks)