# Access Control

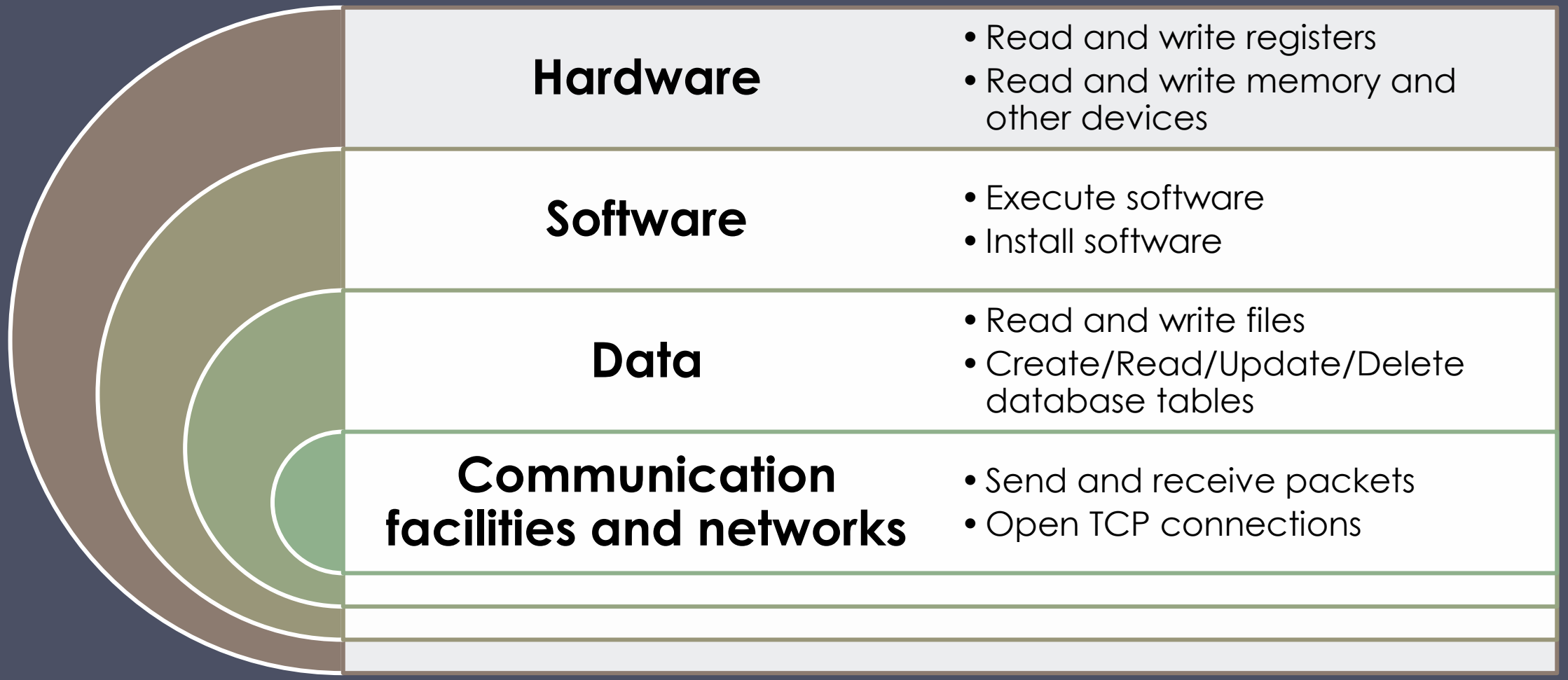
Introduction to Computer Security

Naercio Magaia and Imran Khan

# Contents

- Access Control Matrix
  - Access Control List
  - Capability List
- Discretionary (and Mandatory) Access Control
- Example Unix Access Control
- Role Based Access Control
- Attribute Based Access Control

# Assets and what can be done with them



| | |
|---|---|
| **Hardware** | • Read and write registers<br>• Read and write memory and other devices |
| **Software** | • Execute software<br>• Install software |
| **Data** | • Read and write files<br>• Create/Read/Update/Delete database tables |
| **Communication facilities and networks** | • Send and receive packets<br>• Open TCP connections |

# Access Control Definition

The Internet Security Glossary RFC 4949 defines access control as:

"a process by which use of *computer assets and* system resources **is regulated according to a security policy** and **is permitted only by authorized entities** (users, programs, processes, or other systems) according to that policy"

# Access Control Principles

- In a broad sense, all of computer security is concerned with access control

- RFC 4949 defines computer security as:

"measures that **implement and assure security services** in a computer system, particularly those that assure access control service"

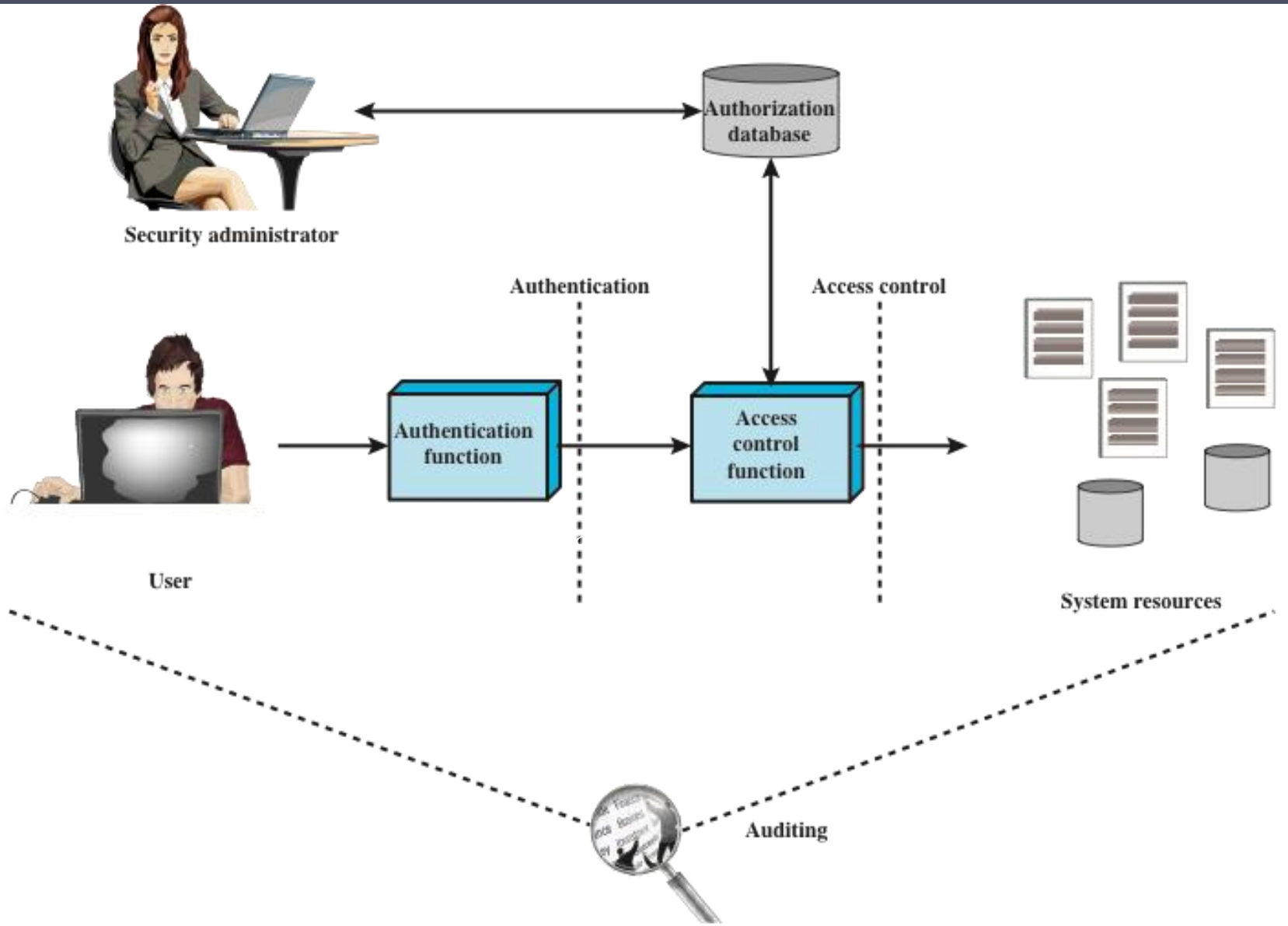



**Figure 4.1 Relationship Among Access Control and Other Security Functions**

*Source*: Based on [SAND94].

# Access Control Policies

- Discretionary access control (DAC)
  - Controls access based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do
- Mandatory access control (MAC)
  - Controls access based on comparing security labels with security clearances
- Role-based access control (RBAC)
  - Controls access based on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles
- Attribute-based access control (ABAC)
  - Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions

# Basic Elements of Access Control

## Subject

- An entity capable of accessing objects
- Three classes
  - Owner
  - Group
  - World

## Object

- A resource to which access is controlled
- Entity used to contain and/or receive information

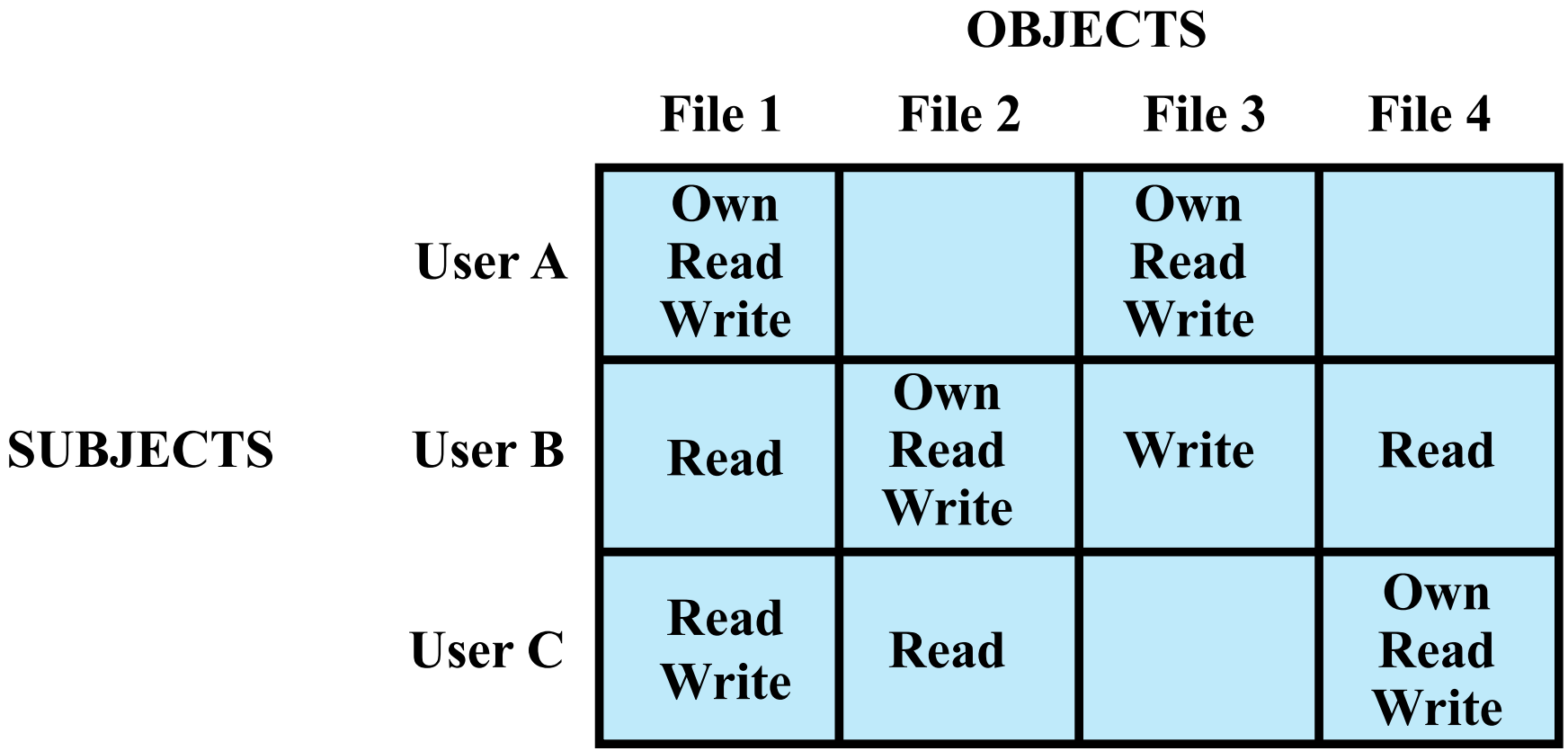
## Access right

- Describes the way in which a subject may access an object
- Could include:
  - Read, Write, Execute
  - Create, Delete
  - Search

# Discretionary Access Control (DAC)

- A scheme in which an entity may be granted access rights that permit the entity, by its own volition, to **enable another entity** to access some resource
- Often provided using an **access control matrix**
  - One dimension consists of identified subjects that may attempt data access to the resources
  - The other dimension lists the objects that may be accessed
- Each entry in the matrix indicates the access rights of a particular subject for a particular object

# Access Control Structures

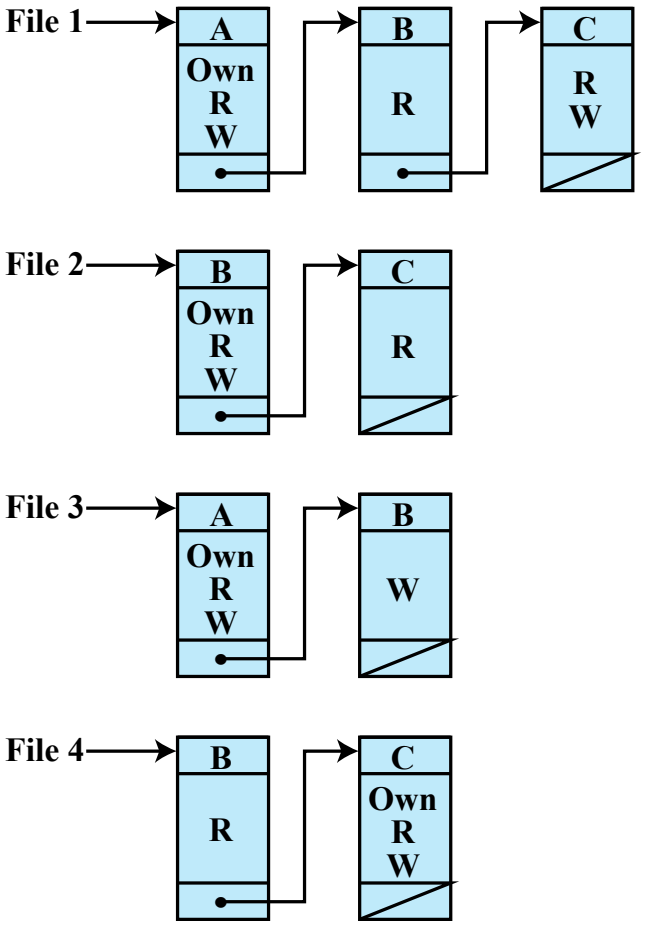


| | | OBJECTS | | | |
|---|---|---|---|---|---|
| | | File 1 | File 2 | File 3 | File 4 |
| SUBJECTS | User A | Own<br>Read<br>Write | | Own<br>Read<br>Write | |
| | User B | Read | Own<br>Read<br>Write | Write | Read |
| | User C | Read<br>Write | Read | | Own<br>Read<br>Write |

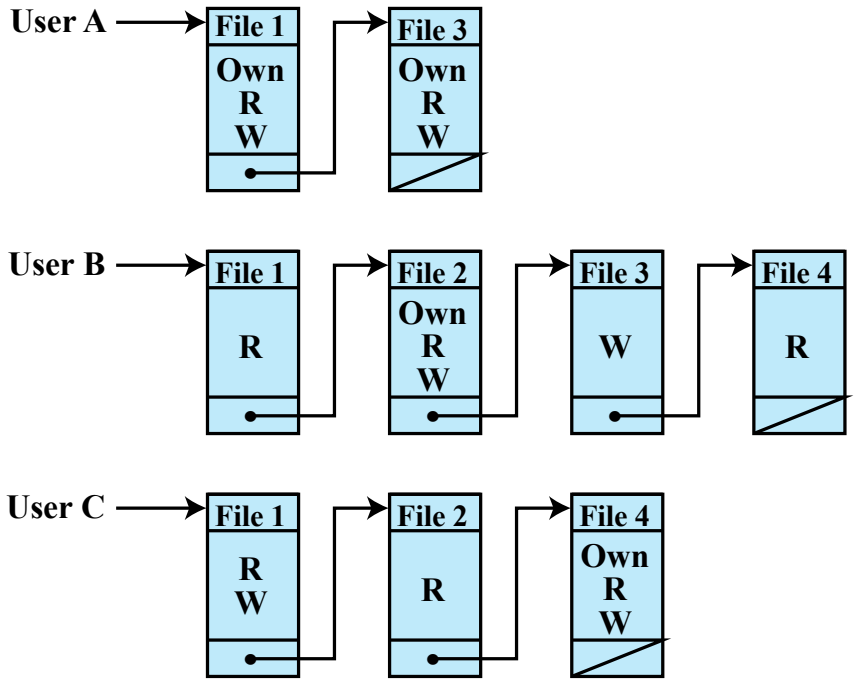
(a) Access matrix

This is a sparse matrix, so represent in software as a list

# Example of Access Control Structures




(b) Access control lists for files of part (a)




(c) Capability lists for files of part (a)

# UNIX File Access Control

## UNIX files are administered using inodes (index nodes)

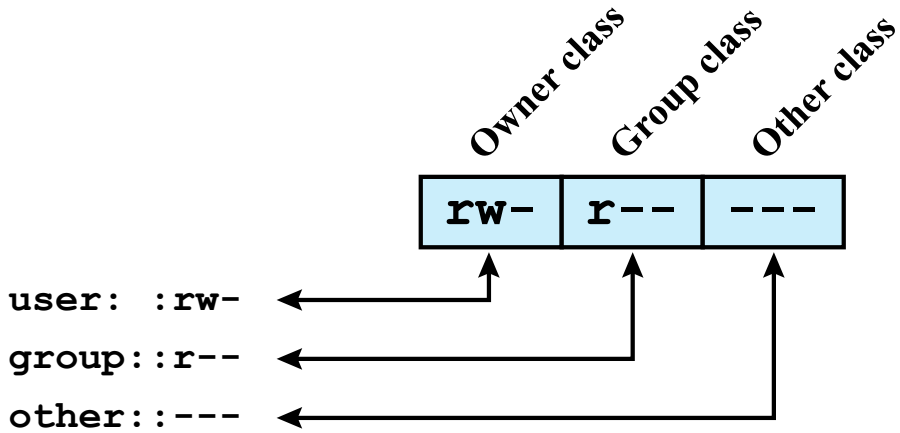
- Control structures with key information needed for a particular file
- Several file names may be associated with a single inode
- An active inode is associated with exactly one file
- File attributes, permissions and control information are sorted in the inode
- On the disk there is an **inode table**, or inode list, that contains the inodes of all the files in the file system
- When a file is opened its inode is brought into main memory and stored in a memory resident inode table

## Directories are structured in a hierarchical tree

- May contain files and/or other directories
- Contains file names plus pointers to associated inodes

# UNIX File Access Control

- Unique user identification number (user ID)
- Member of a primary group identified by a group ID
- Belongs to a specific group
- 12 protection bits
  - Specify read, write, and execute permission for the owner of the file, members of the group and all other users
- The owner ID, group ID, and protection bits are part of the file's inode




(a) Traditional UNIX approach (minimal access control list)

**Figure 4.5 UNIX File Access Control**

# Traditional UNIX File Access Control

- "Set user ID"(SetUID) and "Set group ID"(SetGID)
  - System temporarily uses rights of the file owner/group in addition to the real user's rights when making access control decisions
  - Enables privileged programs **to access files/resources not generally accessible**
- Sticky bit
  - When applied to a directory it specifies that **only the owner** of any file in the directory can rename, move, or delete that file
- Superuser
  - Is exempt from usual access control restrictions
  - Has system-wide access

# Access Control Lists (ACLs) in UNIX

**Modern UNIX systems support ACLs**

- FreeBSD, OpenBSD, Linux, Solaris

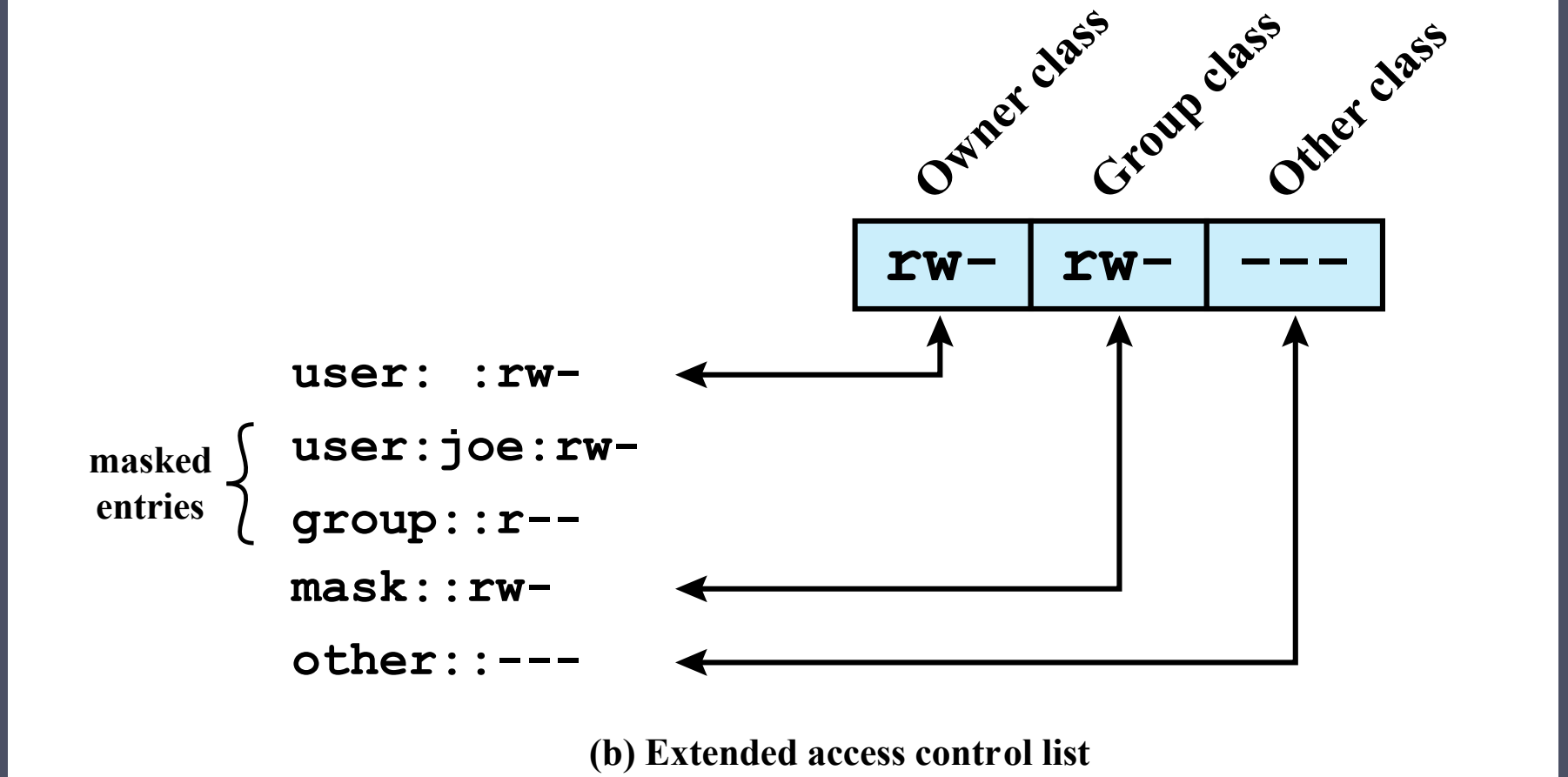
**FreeBSD**

- `setfacl` command assigns a list of UNIX user IDs and groups to a file
- Any number of users and groups can be associated with a file
- Each with read, write, and execute protection bits
- A file does not need to have an ACL
- Includes an additional protection bit that indicates whether the file has an extended ACL

**When a process requests access to a file system object two steps are performed:**

- Step 1 selects the most appropriate ACL
- Step 2 checks if the matching entry contains sufficient permissions

# Unix File Access Control

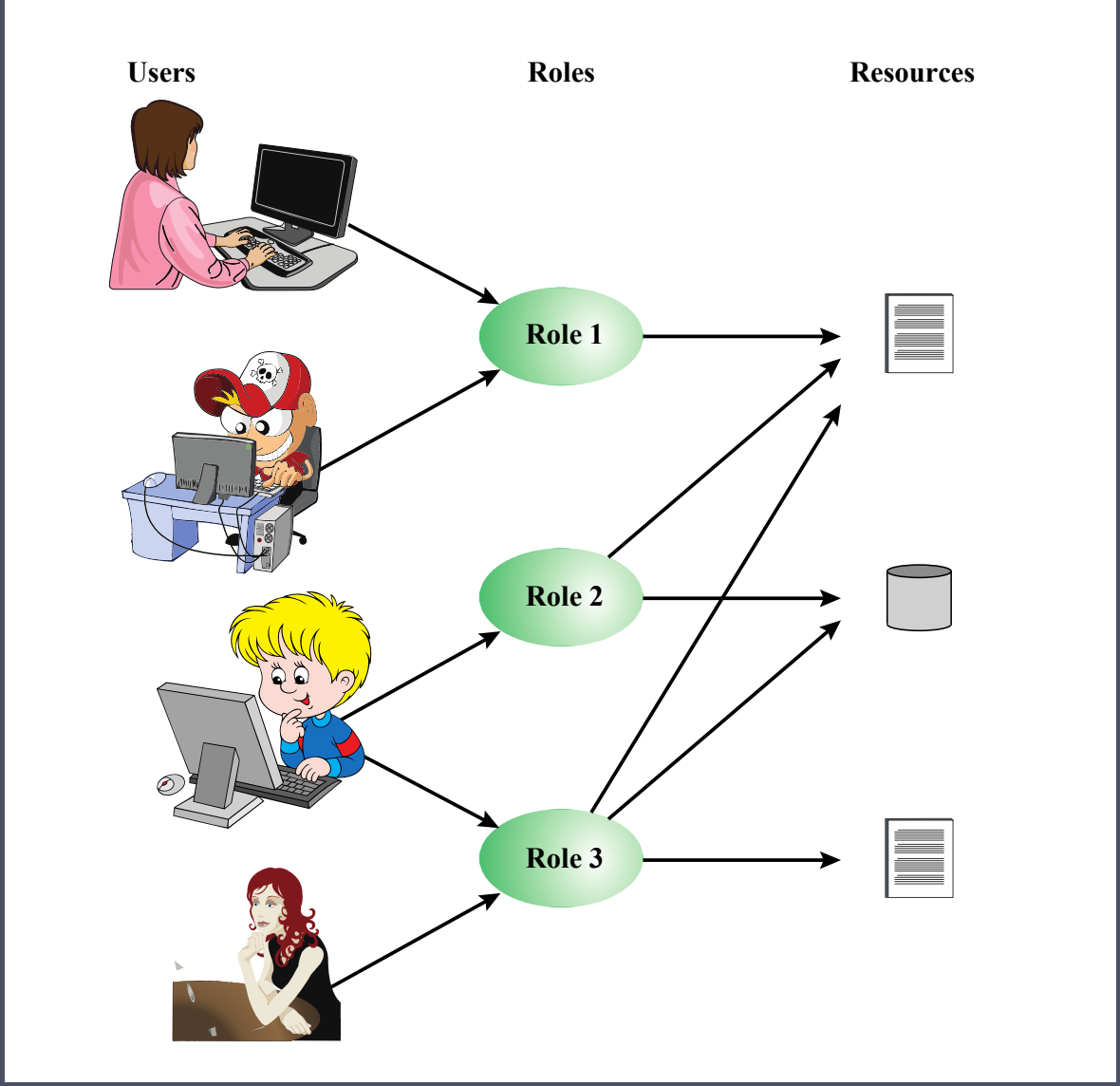



(b) Extended access control list

# Role Based Access Control

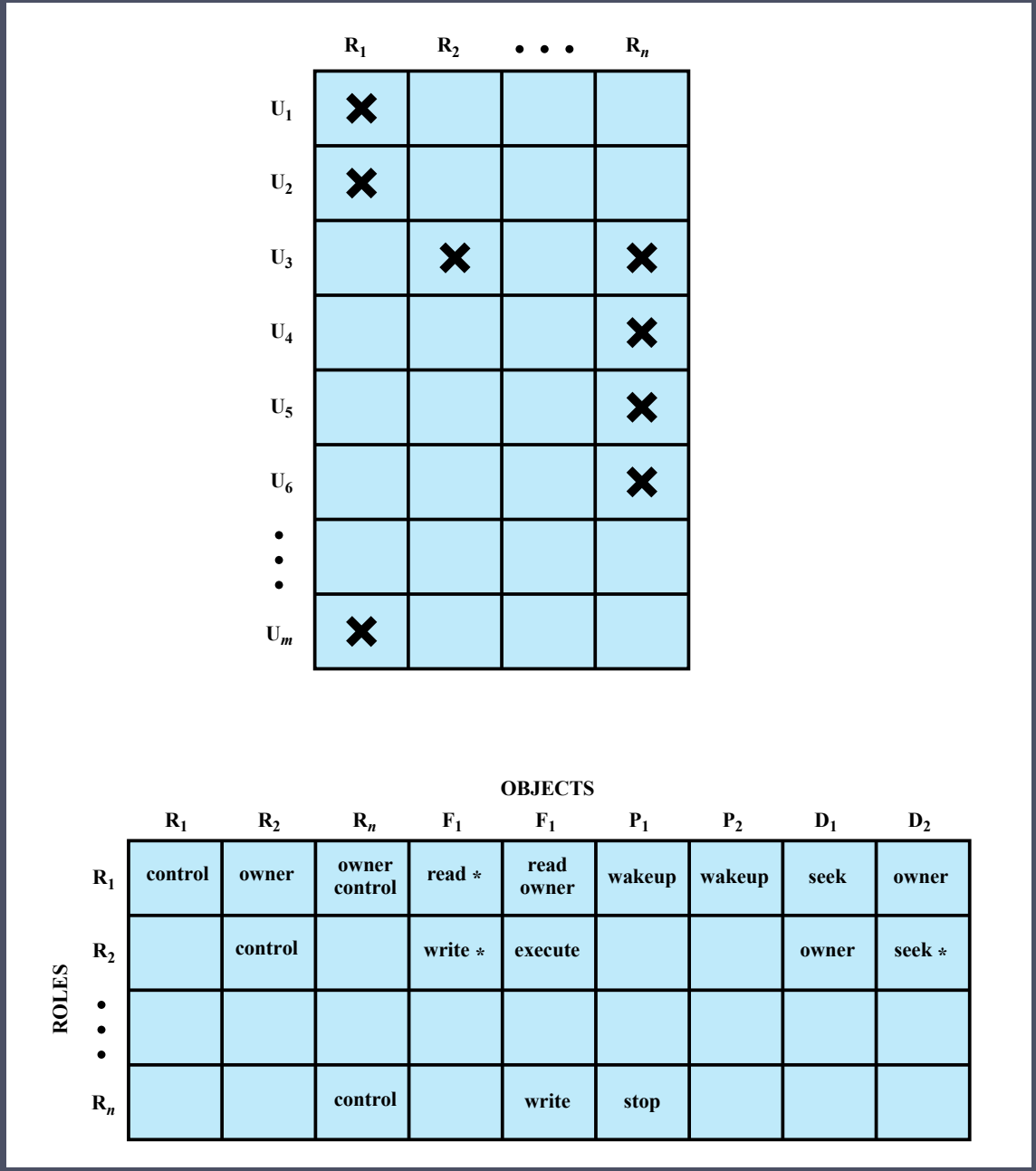
- Real computer systems are used by organizations with defined job descriptions
- We can use the job descriptions to define ***roles*** within the computer system
- We can then define the security policy around **what a role needs to do to deliver** on the job

# Users, Roles, and Resources

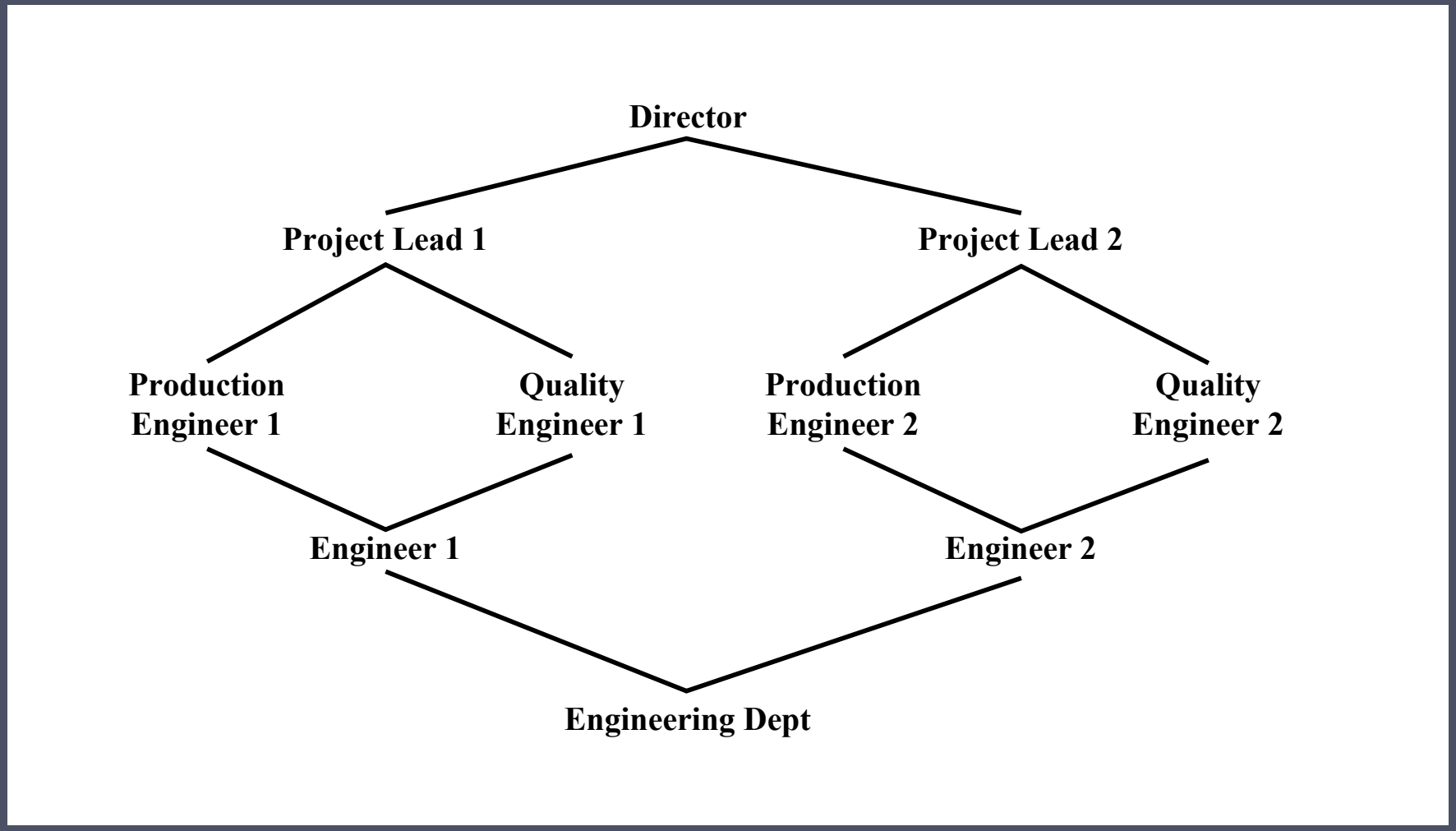


# Roles and the Access Control Matrix

- Map users into their roles
  - A given user may have multiple roles
- Use the Access Control Matrix to define what each role can do
- The matrix can then be represented on the system as either an Access Control List or a Capability List



# Example of Role Hierarchy



# Constraints - RBAC

- Provide a means of adapting RBAC to the specifics of administrative and security policies of an organization
- A defined relationship among roles or a condition related to roles
- Types:

| Mutually exclusive roles | Cardinality | Prerequisite roles |
| --- | --- | --- |
| • A user can only be assigned to one role in the set (either during a session or statically)<br>• Any permission (access right) can be granted to only one role in the set | • Setting a maximum number with respect to roles | • Dictates that a user can only be assigned to a particular role if it is already assigned to some other specified role |

# Attribute-Based Access Control (ABAC)

Can define authorizations that express conditions on properties of both the resource and the subject

Strength is its flexibility and expressive power

Main obstacle to its adoption in real systems has been **concern about the performance** impact of evaluating predicates on both resource and user properties for each access

Web services have been pioneering technologies through the introduction of the eXtensible Access Control Markup Language (XACML)

There is considerable interest in applying the model to cloud services

# ABAC Model: Attributes

## Subject attributes

- A subject is an **active entity** that causes information to flow among objects or changes the system state
- Attributes define the **identity and characteristics** of the subject

## Object attributes

- An object (or resource) is a **passive** information system-related **entity** containing or receiving information
- Objects have attributes that can be leverages to make access control decisions

## Environment attributes

- Describe the operational, technical, and even situational environment or **context** in which the information access occurs
- These attributes have so far been largely ignored in most access control policies

# ABAC

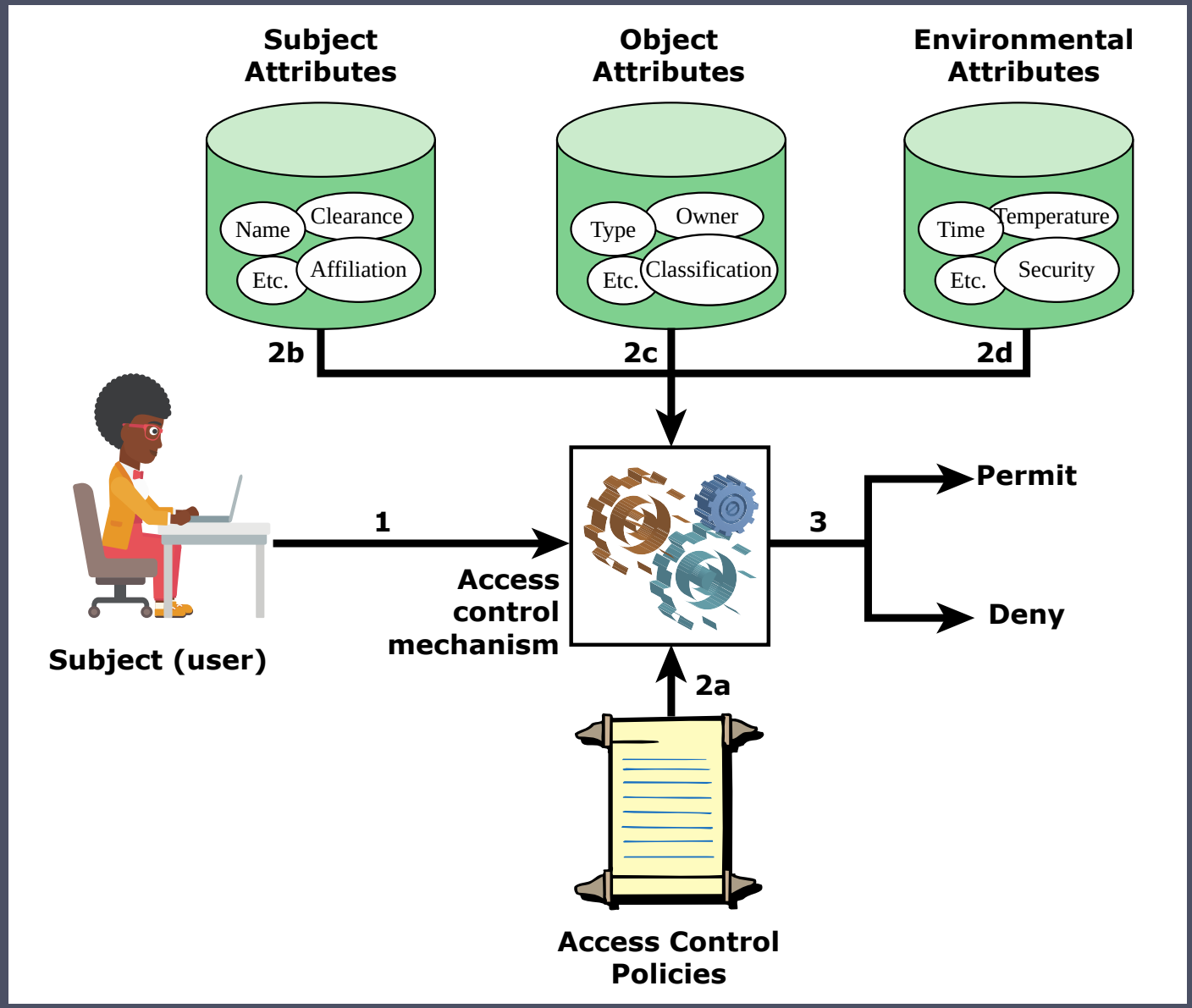
Distinguishable because it controls access to objects by **evaluating rules against the attributes of entities, operations, and the environment** relevant to a request

Relies upon the **evaluation of attributes of the subject, attributes of the object, and a formal relationship** or access control rule defining the allowable operations for subject-object attribute combinations in a given environment

Systems are **capable of enforcing** DAC, RBAC, and MAC concepts

Allows an **unlimited number of attributes to be combined** to satisfy any access control rule

# ABAC Scenario



# ABAC Policies

A **policy** is a set of rules and relationships that govern **allowable behavior** within an organization, based on the **privileges of subjects** and **how resources or objects are to be protected** under which environment conditions

Typically written from the **perspective of the object** that needs protecting and the **privileges available to subjects**

**Privileges** represent the authorized behavior of a subject and are defined by an authority and embodied in a policy

Other terms commonly used instead of privileges are rights, authorizations, and entitlements

# Summary

- Access Control Matrix
  - Access Control List
  - Capability List
- Discretionary Access Control
- Example Unix Access Control
- Role Based Access Control
- Attribute Based Access Control
  - Use attributes of subject, object and environment to provide very expressive security policies
  - Appropriate in complex web services, using eXtended Access Control Markup Language