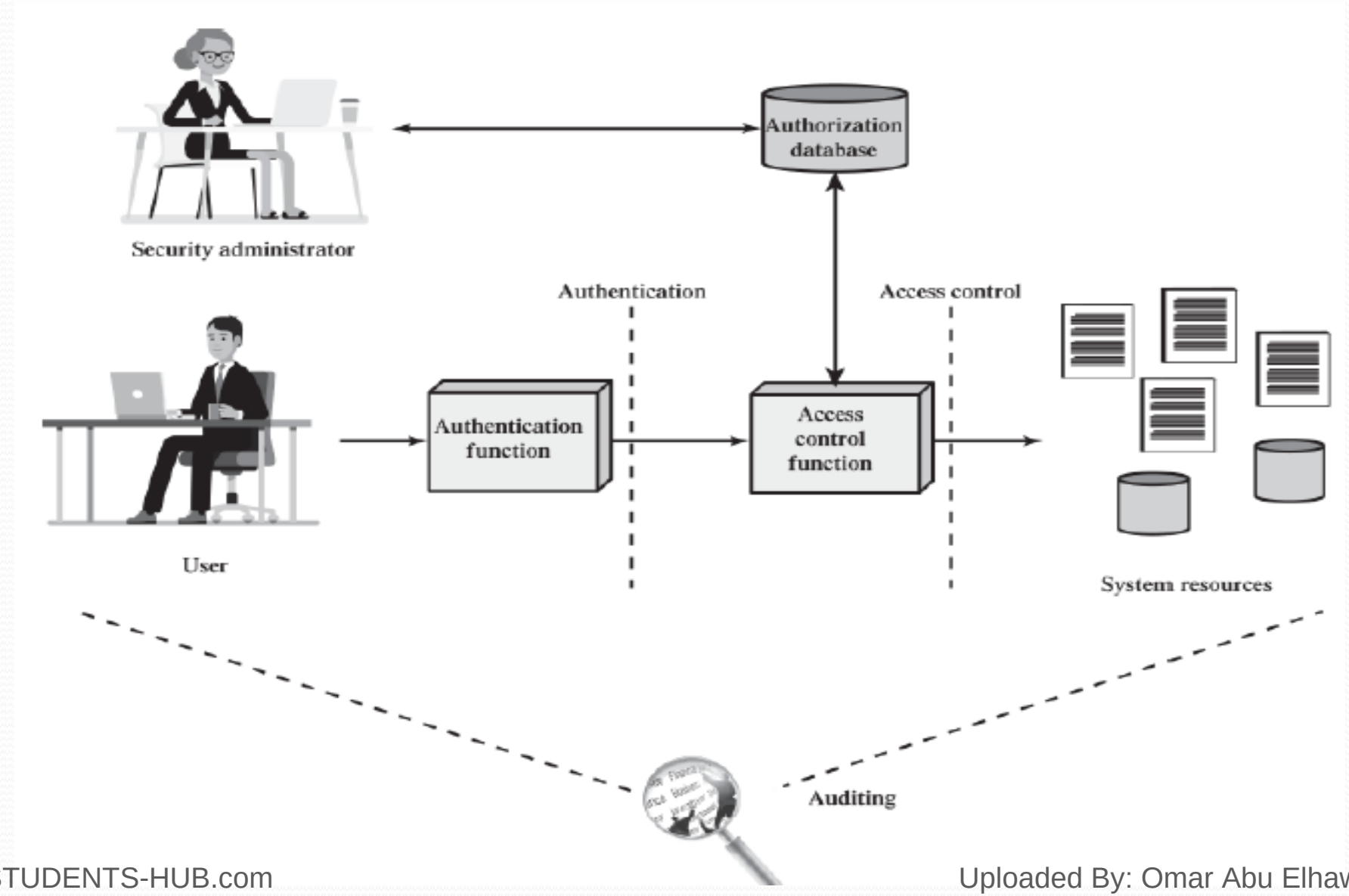
Access Control

Dr. Asem Kitana

Access Control

- The prevention of unauthorized use of a system resource, including the prevention of use of a resource in an unauthorized manner.
- System resources, such as applications, operating systems, firewalls, routers, files, and databases.
- The main purpose is to control *who* can do *what* on a system.
- Using a system does NOT mean someone can do what he/she likes (there are restrictions).

Access Control and Security Functions



Access Control and Security Functions

- The previous figure shows a relationship among access control and other security functions, such as:
  - Authentication: verification that the identity of a user or other entity are valid.
  - Authorization: granting of a right or permission to a system entity to access a resource.
  - Auditing: independent review of system records and activities in order to test for adequacy of system control, ensure compliance to policy, detect breaches and recommend changes.

Access control policies

- Access control policies are generally grouped into the following categories:
  - Discretionary access control (DAC).
  - Mandatory access control (MAC).
  - Role-based access control (RBAC).
  - Attribute-based access control (ABAC).

Access control policies

- Discretionary access control (DAC): use identity of requestor and access rules (that determine what requestor is allowed to do) to control access. Entities may allow other entities to access resources. (i.e. based on the identity of the requestor and access rules).
- Mandatory access control (MAC): based on comparing security labels with security clearances to determine access. Entities cannot grant access to resources to other entities

Access control policies

- Role-based access control (RBAC): Based on roles of users in a system, and rules for roles are used to control access.
- Attribute-based access control (ABAC): based on the attributes of the user, the resources and the current environment

Elements of Access Control System

- Subject: entity capable of access resources
  - Subject could be a process, user, or application.
  - Classes of subject, e.g. Owner, Group, World
- Object: resource to which access is controlled
  - e.g. files, directories, records, or programs.
- Access right: describes way in which a subject may access an object
  - e.g. read (view), write (add/modify), execute, delete, create, search

Elements of Access Control System

There are three basic classes of subject, with different access rights for each class:

- **Owner:** This may be the creator of a resource, such as a file. For system resources, ownership may belong to a system administrator. For project resources, a project administrator may be assigned ownership.
- **Group:** A named group of users may also be granted access rights. In most schemes, a user may belong to multiple groups.
- **World:** The least amount of access is granted to users who are able to access the system, but are not included in the categories owner and group for this resource.

Requirements of Access Control System

- Reliable input: a mechanism to authenticate
- Fine and coarse specifications: regulate access at varying levels, detailed vs. general (e.g., an attribute at specific time or entire DB)
- Least privilege: minimal authorization to do a specific task.
- Separation of duty: divide tasks among different individuals
- Open and closed policies: closed policy (you can access specified resources and anything else is denied), open policy (you can access all the resources except those prohibited).
- Administrative policies: who can add, delete, modify rules

Discretionary Access Control (DAC)

Discretionary Access Control

- DAC: an entity may be granted access rights that permit the entity, if they choose so, to enable another entity to access a resource.
- In DAC, an entity that can change who can access a resource could be any entity and not necessarily to be the security admin.
- DAC is a common access control scheme in operating systems and database management systems.
- Often implemented using an access matrix
  - lists subjects in one dimension (rows)
  - lists objects in the other dimension (columns)
  - each entry (cell) specifies access rights of the specified subject to that object
- Can decompose by either row or column

Discretionary Access Control

- Access Matrix specifies access rights of subjects on objects (a.k.a. access control matrix).
- In practice, access matrix is sparse, so it is implemented as either:
  - Access Control Lists (ACL): For each object, list subjects and their access rights (decomposed by column).
  - Capability Lists: For each subject, list objects and the rights the subject have on that object (decomposed by row).
- Alternative implementation:
  - Authorization Tables: Listing subject, access mode and object; easily implemented in database.

Example of DAC Access Matrix

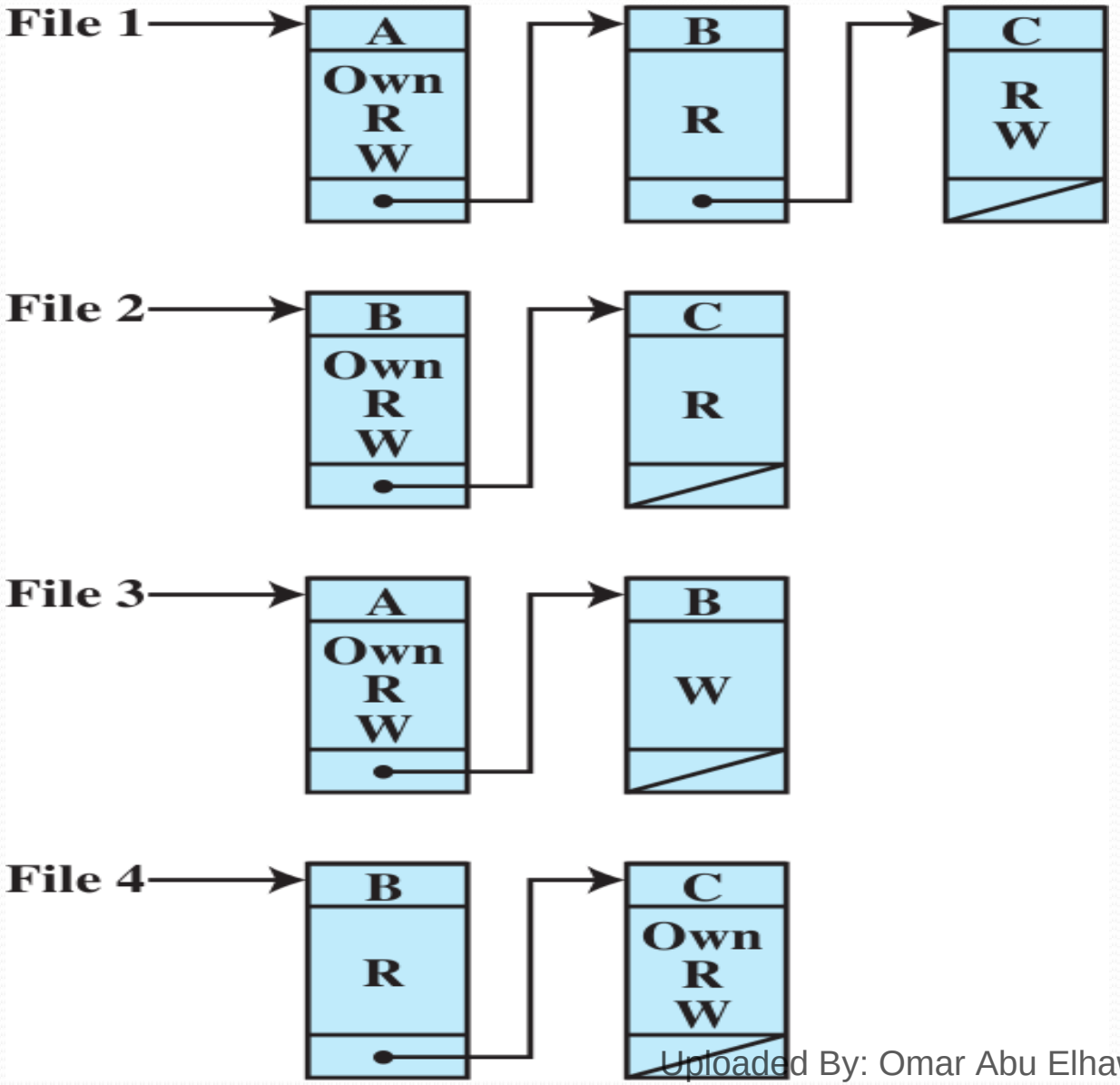
- This access control matrix could be deployed in an OS.
- In reality, this matrix could have thousands of objects and hundreds of subjects. In addition to many empty cells.

SUBJECTS	OBJECTS: File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write

(a) Access matrix

Example of Access Control List

- One list for each object.
- ACL more efficient than access matrix.

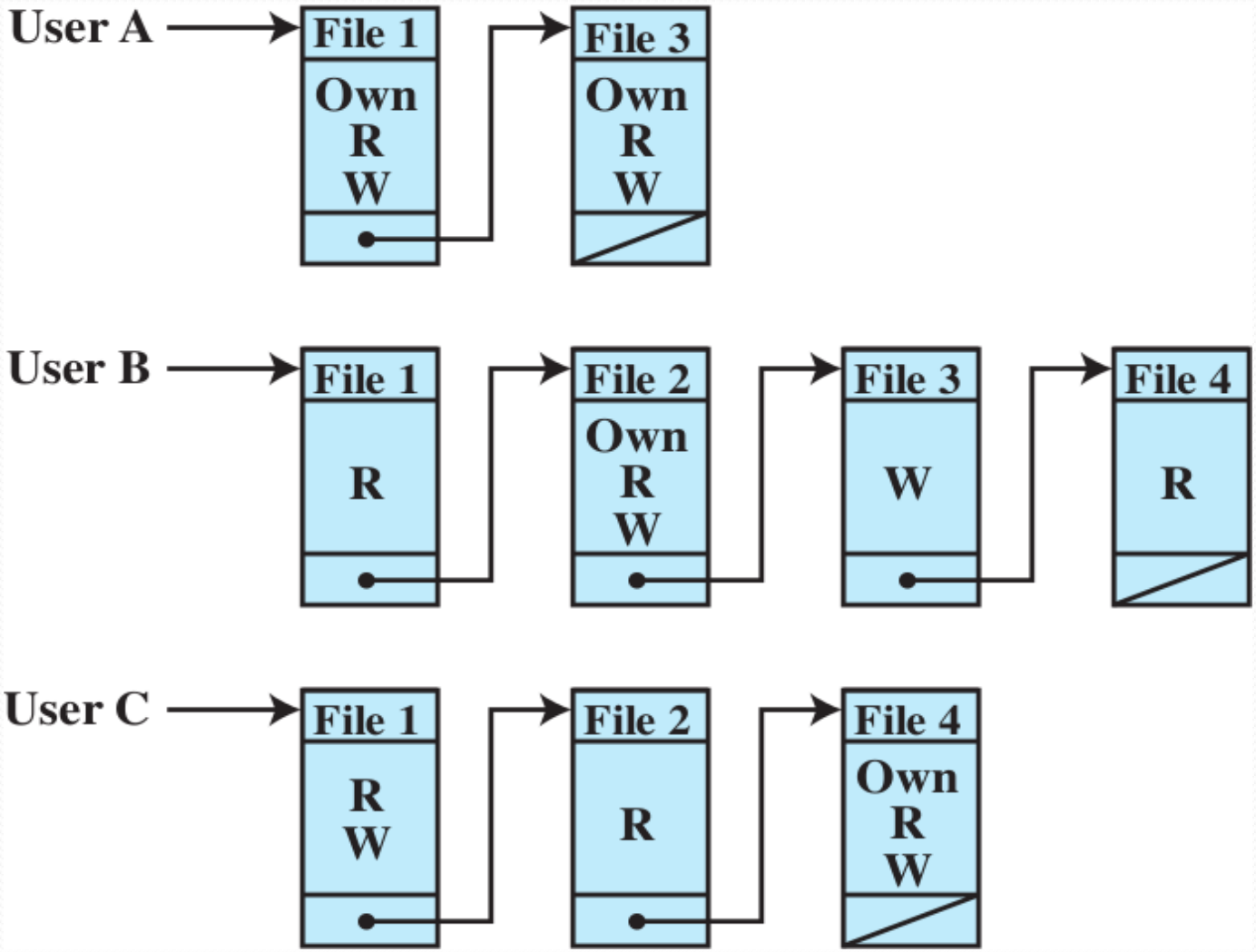


ACL example

- **ACL(File1) = { (UserA,{own, read, write}), (UserB,{read}), (UserC, {read, write}) }**
- **ACL(File2) = { (UserB,{o, r, w}), (UserC,{r}) }**
- **ACL(File3) = { (UserA,{o, r, w}), (UserB,{w}) }**
- **ACL(File4) = { (UserB,{r}), (UserC,{o, r, w}) }**

Example of Capability List

- One list for each subject.



Capability List example

- **Cap(UserA) = { (File1,{own, read, write}), (File3,{own, read, write}) }**
- **Cap(UserB) = { (File1,{r}), (File2,{o, r, w}, (File3,{w}), (File4,{r}) }**
- **Cap(UserC) = { (File1,{r, w}), (File2,{r}, (File4,{o, r, w}) }**

Example of Authorization Table

Subject	Access Mode	Object
A	Own	File 1
A	Read	File 1
A	Write	File 1
A	Own	File 3
A	Read	File 3
A	Write	File 3
B	Read	File 1
B	Own	File 2
B	Read	File 2
B	Write	File 2
B	Write	File 3
B	Read	File 4
C	Read	File 1
C	Write	File 1
C	Read	File 2
C	Own	File 4
C	Read	File 4
C	Write	File 4

Graham-Denning Model

Represents a general model for DAC, that extends the universe of objects to include processes, devices, memory locations, subjects.

- **Processes:** Access rights include the ability to delete a process, stop (block), and wake up a process.
- **Devices:** Access rights include the ability to read/write the device, to control its operation (e.g., a disk seek), and to block/unblock the device for use.
- **Memory locations or regions:** Access rights include the ability to read/write certain regions of memory that are protected such that the default is to disallow access.
- **Subjects:** Access rights with respect to a subject have to do with the ability to grant or delete access rights of that subject to other objects.

Example of Graham-Denning Model

SUBJECTS	OBJECTS: Subjects S_1	S_2	S_3	Files F_1	F_2	Processes P_1	P_2	Disk drives D_1	D_2
S_1	control	owner	owner control	read*	read owner	wakeup	wakeup	seek	owner
S_2		control		write*	execute			owner	seek*
S_3			control		write	stop			

* = copy flag set

In this access control matrix *A*, each entry *A*[*S*, *X*] contains strings, called access attributes, that specify the access rights of subject *S* to object *X*. For example: S1 may read file F1, because 'read' appears in A[S1, F1].

Graham-Denning Model

- Each subject/object has an owner
- Each subject has a controller (which may be itself)
- An access right may be transferable or nontransferable
- Copy flag (*): means a subject can transfer this specific right, with or without copy flag, to another subject.

Graham-Denning Model Mechanism

- A subject issues a request of type α for object *X*.
- The request causes the system (the operating system) to generate a message of the form (S0, α, X) to the controller for *X*.
- The controller examines the access matrix *A* to determine if α is in A[S0, X]. If so, the access is allowed; if not, the access is denied and a protection violation occurs. The violation should trigger a warning and appropriate action.