# OPERATING SYSTEMS

## I/O Management, System Protection and Security

**Arya S S**

Department of Computer Science

# OPERATING SYSTEMS

## System Protection- Access Matrix

**Arya S S**

Department of Computer Science

## Slides Credits for all PPTs of this course

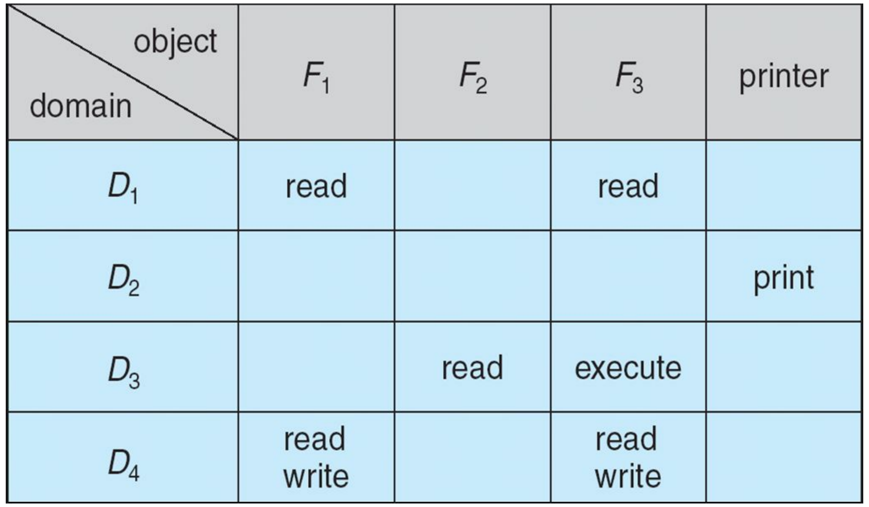
- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the following resources and persons:

1. Slides of Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne - 9th edition 2013 and some slides from 10th edition 2018
2. Some conceptual text and diagram from Operating Systems - Internals and Design Principles, William Stallings, 9th edition 2018
3. Some presentation transcripts from A. Frank – P. Weisberg
4. Some conceptual text from Operating Systems: Three Easy Pieces, Remzi Arpaci-Dusseau, Andrea Arpaci Dusseau

OPERATING SYSTEMS

## Access Matrix

- View protection abstractly 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$_j$



| object domain | $F_1$ | $F_2$ | $F_3$ | printer |
|---|---|---|---|---|
| $D_1$ | read | | read | |
| $D_2$ | | | | print |
| $D_3$ | | read | execute | |
| $D_4$ | read write | | read write | |

## 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
- User who creates object can define access column for that object
- 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 $O_j$ (denoted by "*")*
    - *control – $D_i$ can modify $D_j$ access rights*
    - *transfer – switch from domain $D_i$ to $D_j$*
  - *Copy* and *Owner* applicable to an object
  - *Control* applicable to domain object

## Use of Access Matrix (Cont.)

- Access matrix design separates mechanism from policy
  - Mechanism
    - Operating system provides access-matrix + rules
    - It 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
- But doesn’t solve the general confinement problem i.e. preventing a process from taking disallowed actions
  - Ex: In a client/server situation, preventing a server from leaking information that the client considers confidential

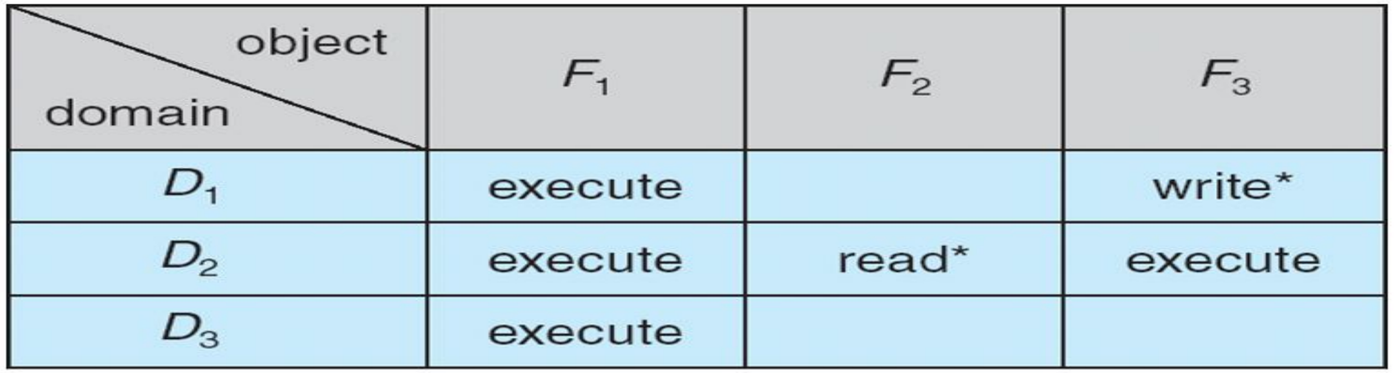
## Access Matrix with Domains as Objects

Processes should be able to switch from one domain to another.
Switching from domain *Di* to domain *Dj* is allowed if and only if the access right switch ∈ access(*i*, *j*).

| object / domain | $F_1$ | $F_2$ | $F_3$ | laser printer | $D_1$ | $D_2$ | $D_3$ | $D_4$ |
|---|---|---|---|---|---|---|---|---|
| $D_1$ | read | | read | | | switch | | |
| $D_2$ | | | | print | | | switch | switch |
| $D_3$ | | read | execute | | | | | |
| $D_4$ | read write | | read write | | switch | | | |

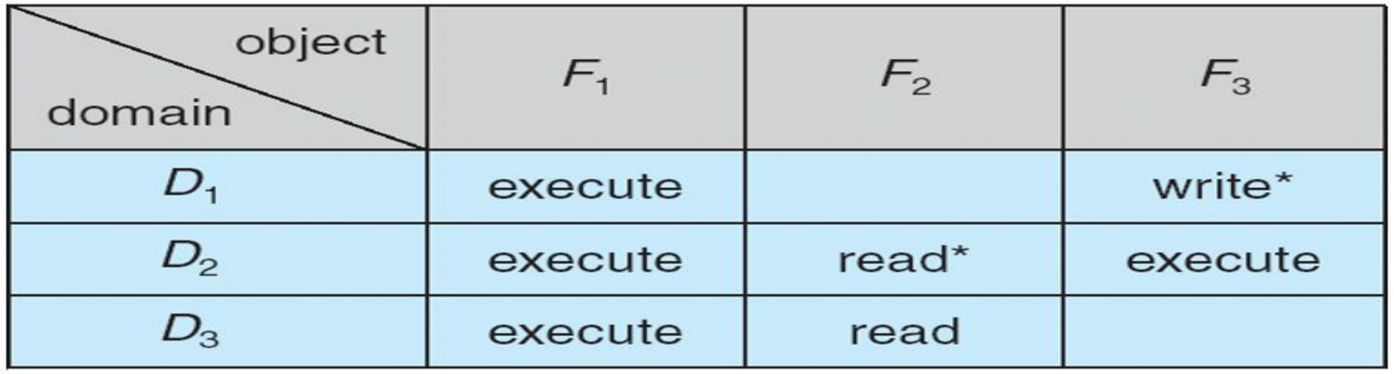
## Access Matrix with Copy Rights

- The ability to copy an access right from one domain (or row) of the access matrix to another is denoted by an asterisk (*) appended to the access right.



| object / domain | $F_1$ | $F_2$ | $F_3$ |
|---|---|---|---|
| $D_1$ | execute | | write* |
| $D_2$ | execute | read* | execute |
| $D_3$ | execute | | |

(a)



| object / domain | $F_1$ | $F_2$ | $F_3$ |
|---|---|---|---|
| $D_1$ | execute | | write* |
| $D_2$ | execute | read* | execute |
| $D_3$ | execute | read | |

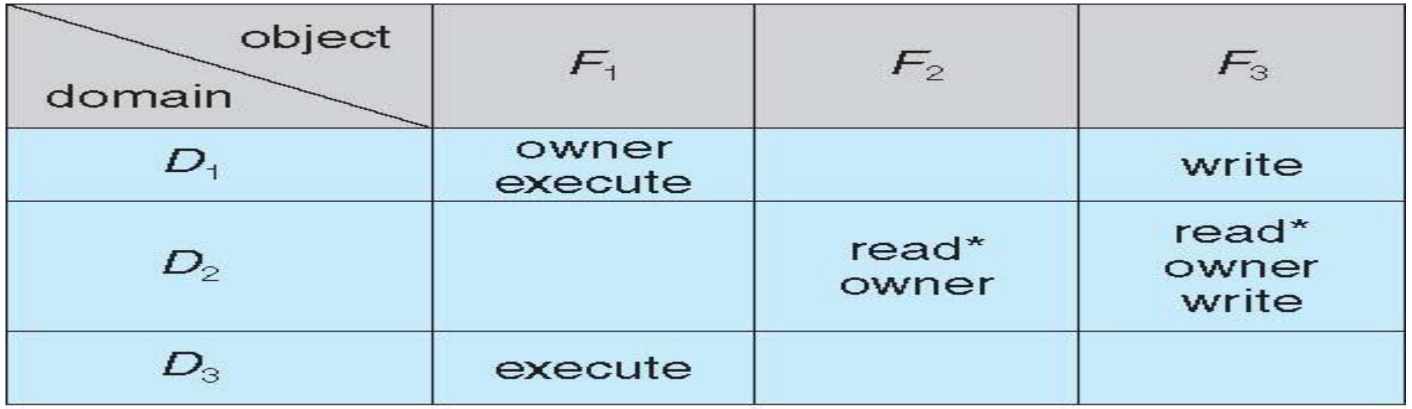
(b)

## Access Matrix with Copy Rights

The copy scheme has 3 variants:

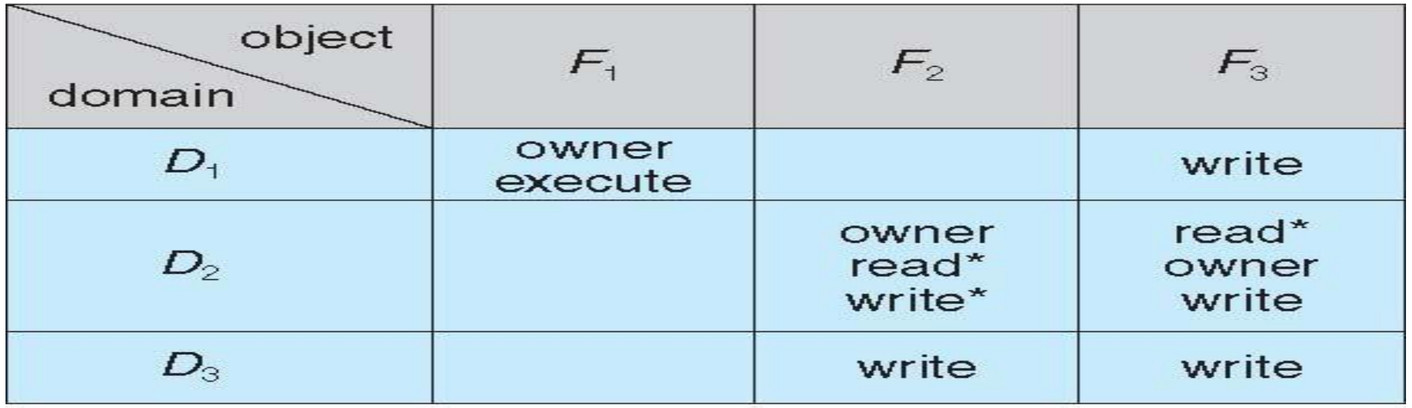
1. A right is copied from access(i,j) to access(k,j) is not limited: This action is called **copy**.
    - When the right Read* is copied from access(i,j) to access(k,j), the Read* is created.
    - So, a process executing in Dk can further copy the right Read*.
2. Propagation of the copy right may be limited: This action is called **limited copy**.
    - When the right Read* is copied from access(i,j) to access(k,j), only the Read (not Read*) is created.
    - So, a process executing in Dk cannot further copy the right Read.
3. A right is copied from access(i,j) to access(k,j); it is then removed from access(i,j).
    - This action is called a **transfer of a right**, rather than a copy

## Access Matrix with Owner Rights



| object / domain | $F_1$ | $F_2$ | $F_3$ |
|---|---|---|---|
| $D_1$ | owner execute | | write |
| $D_2$ | | read* owner | read* owner write |
| $D_3$ | execute | | |

(a)



| object / domain | $F_1$ | $F_2$ | $F_3$ |
|---|---|---|---|
| $D_1$ | owner execute | | write |
| $D_2$ | | owner read* write* | read* owner write |
| $D_3$ | | write | write |

(b)

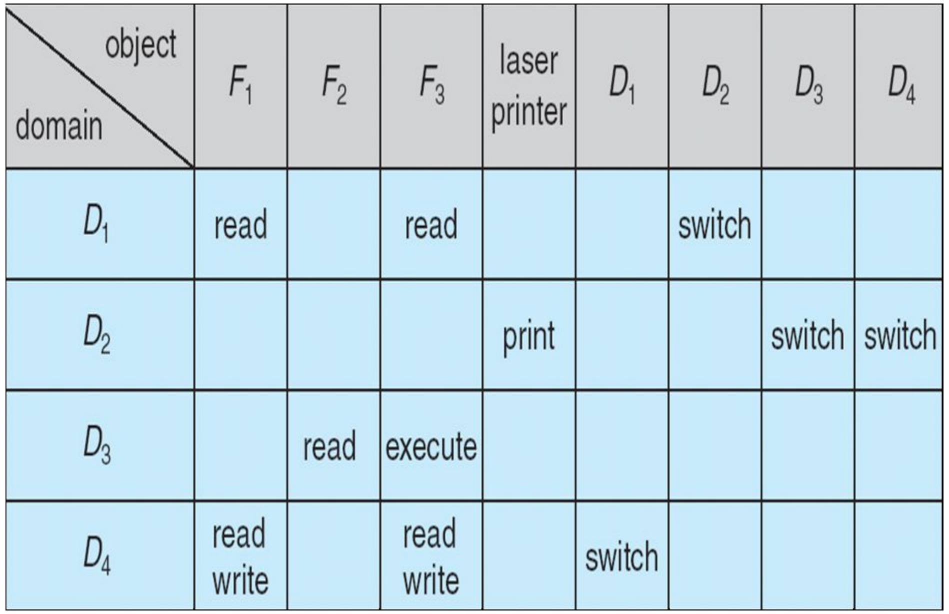
- **Owner** right controls addition of new rights and removal of some rights.
- Domain D1 is the owner of F1 and can add /delete any valid right in column F1
- Owner rights allow a process to change the entries in a column

## Access Matrix with Control Rights

- The copy and owner rights allow a process to change the entries in a column.
- A mechanism is now needed to change the entries in a row.
- The control right is applicable only to domain objects (rows).
- If access(i,j) includes the control right, then a process executing in Di can remove any access right from row j.
- If we include the *control* right in access(D2, D4), then, a process executing in domain D2 could modify domain D4.

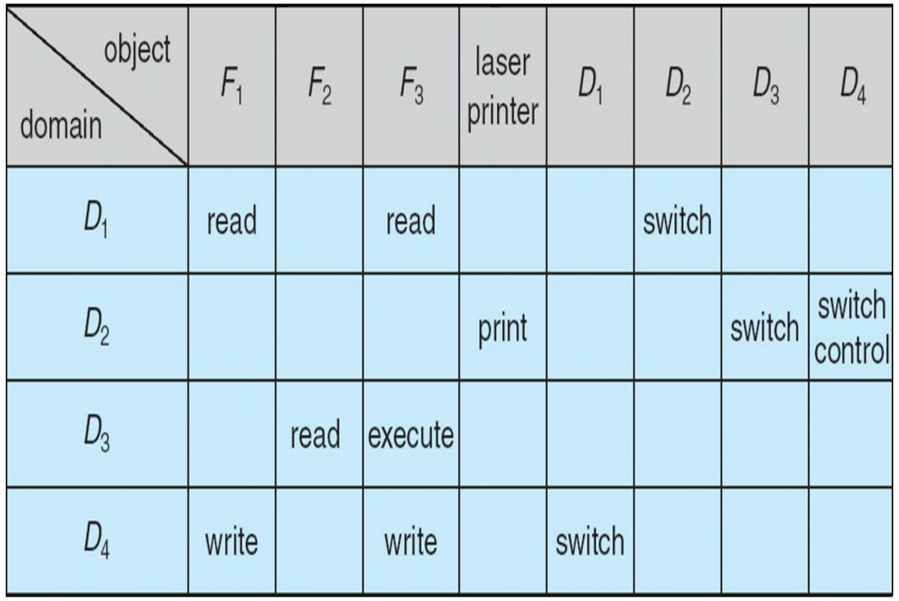
OPERATING SYSTEMS

## Modified Access Matrix with Domains as Objects



| object / domain | $F_1$ | $F_2$ | $F_3$ | laser printer | $D_1$ | $D_2$ | $D_3$ | $D_4$ |
|---|---|---|---|---|---|---|---|---|
| $D_1$ | read | | read | | | switch | | |
| $D_2$ | | | | print | | | switch | switch |
| $D_3$ | | read | execute | | | | | |
| $D_4$ | read write | | read write | | switch | | | |

Fig A



| object / domain | $F_1$ | $F_2$ | $F_3$ | laser printer | $D_1$ | $D_2$ | $D_3$ | $D_4$ |
|---|---|---|---|---|---|---|---|---|
| $D_1$ | read | | read | | | switch | | |
| $D_2$ | | | | print | | | switch | switch control |
| $D_3$ | | read | execute | | | | | |
| $D_4$ | write | | write | | switch | | | |

Fig B: Modified access matrix of fig A

Add **control** right in access(D2, D4).
Then, a process executing in D2 (row) could modify D4 (row).

## THANK YOU

**Arya S S**
Department of Computer Science Engineering

**aryadeep@pes.edu**