CS 547: Foundation of Computer Security

S. Tripathy
IIT Patna

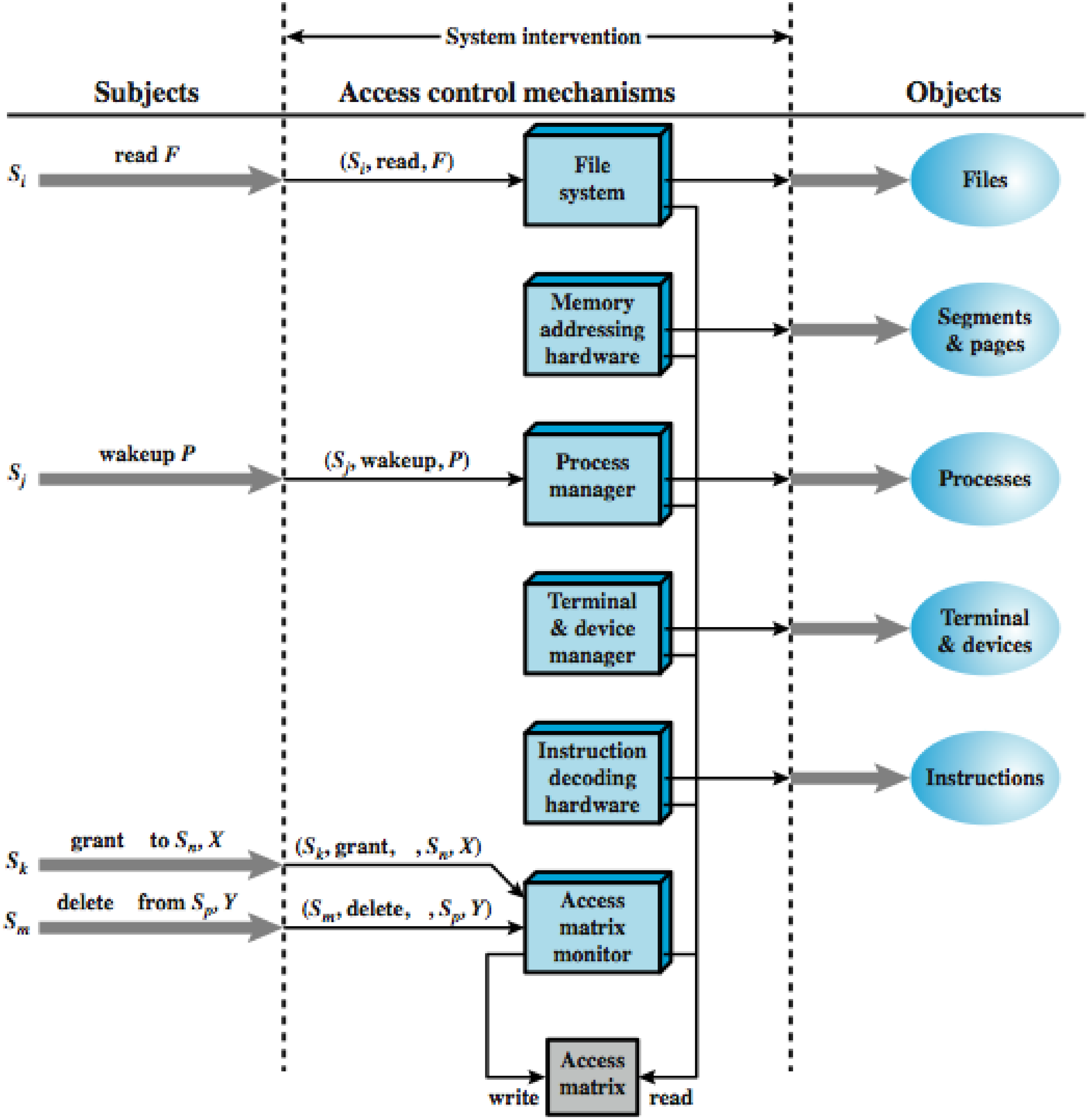
Previous Class

- Access Control
  - Discretionary Access Control

Present class

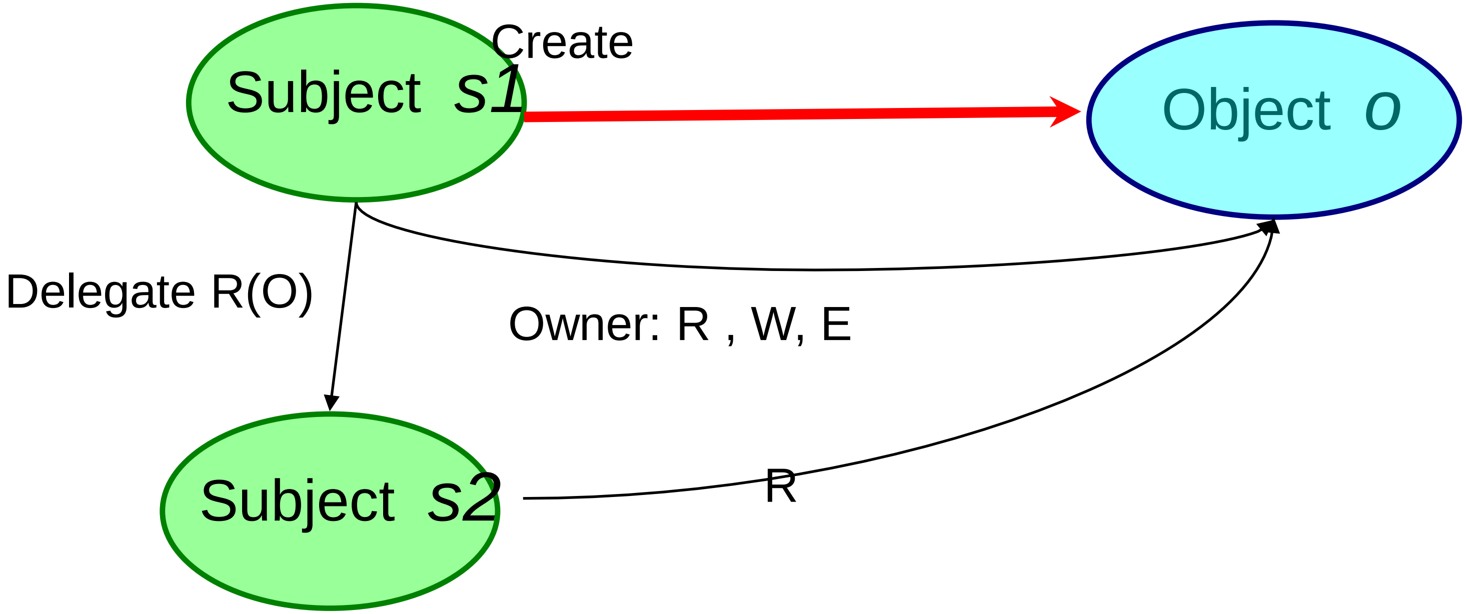
- Access Control
  - Mandatory Access Control
  - Role-Based Access Control

Access Control Function



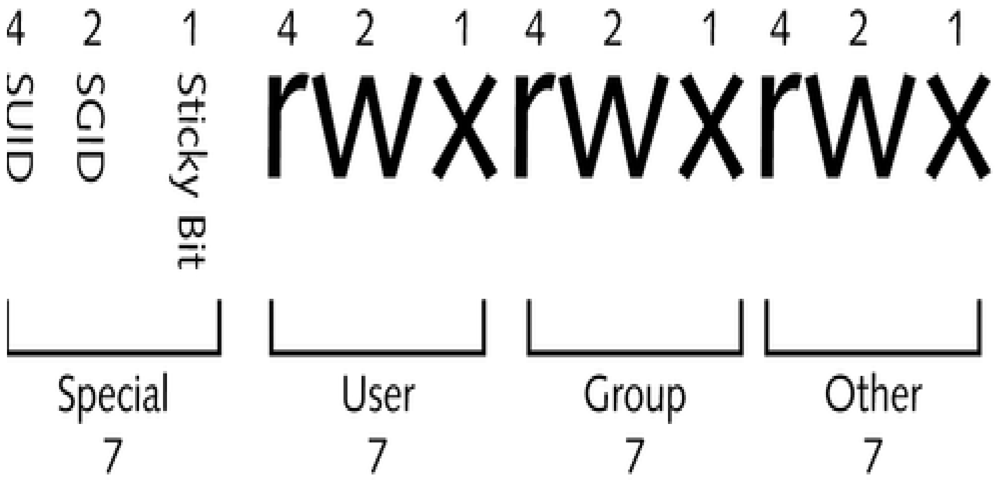
Access Control Models: DAC

- DAC model enforces access control based on user identities, object ownership and permission delegation. The owner of an object may delegate the permission of the object to another user.



Permissions

- SUID (Set User ID)
- SGID(Set Group ID)
- Sticky Bit



SUID bit

- A special permission bit that allows executable files to run using the privileges of the owner of the files rather than the user of the file
- Can be set using commands:

```
chmod  u+s  filelist
chmod  4xxx filelist
```

- Shows up in ls - l in place of the user x bit as an s if the file is executable - (rwsrwxrwx)
- Very dangerous to use

SGID

- A special permission bit that allows executable files to run using the privileges of the owner's group rather than the user's group of the file
- Set using the commands

```
chmod  g+s  filelist
chmod  2xxx  filelist
```

Sticky Bit

- A special bit that can be used as follows:
- For a file/ directory: it sets it up such that only the owner of the directory can delete (or rename) files from the directory, even if other users have write privilege (tmp)
- Can be set using the chmod command using the options:

  ```
  chmod  +t  filelist
  ```
- Shows up in “ls –l” as a t - (rwxrwxrwt)

ACL Commands

- Modern UNIX systems support ACLs
  - FreeBSD, OpenBSD, Linux, Solaris
- FreeBSD
  - `Setfacl` assigns a list of UNIX user IDs and groups
  - any number of users and groups can be associated with a file
  - read, write, 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
- ACLs are read with the getfacl command and set with the setfacl command.

More ACL Command Examples

```
/home/fac/som$getfacl buf.c
# file: buf.c
# owner: som
# group: fac
user::rw-
group::rwx
other::r--

/home/fac/som$setfacl -m group::r buf.c
/home/fac/som$getfacl buf.c
# file: buf.c
# owner: som
# group: fac
user::rw-
group::r--
other::r--
```

Extended ACLs

- ACLs that say more than Unix permissions are extended ACLs
  - Specific users and groups can be named and given permissions via ACLs, which fall under the group class (even for ACLs naming users and not groups)
- .

More ACL Command Examples

```
/home/fac/som$ls -l
total 0
-rw-r--r-- 1 som fac 0 Oct 14 17:57 acltst
/home/fac/som$getfacl acltst
# file: acltst
# owner: som
# group: fac
user::rw-
group::r--
other::r--
```

```
home/fac/som$setfacl -m skparida:rw acltst
/home/fac/som$getfacl acltst
# file: acltst
# owner: som
# group: fac
user::rw-
user:skparida:rw-
group::r--
mask::rw-
```

Access Control Algorithm

- The DACL of a file or folder is a sorted list of ACEs
  - Local ACEs precede inherited ACEs
  - ACEs inherited from folder F precede those inherited from parent of F
  - Among those with same source, Deny ACEs precede Allow ACEs
- Algorithm for granting access request (e.g., read and execute):
  - ACEs in the DACL are examined in order
  - Does the ACE refer to the user or a group containing the user?
  - If so, do any of the accesses in the ACE match those of the request?
  - If so, what type of ACE is it?
    - Deny: return ACCESS_DENIED
    - Allow: grant the specified accesses and if there are no remaining accesses to grant, return ACCESS_ALLOWED
  - If we reach the end of the DACL and there are remaining requested accesses that have not been granted yet, return ACCESS_DENIED

Access Control Lists (ACLs) in UNIX

- when a process requests access to a file system object two steps are performed:
  - *step 1*: selects the most appropriate *ACL*
    - owner, named users, owning / named groups, others
  - *step 2*: checks if the matching entry contains sufficient permissions

DAC Pattern Advantages

advantages:

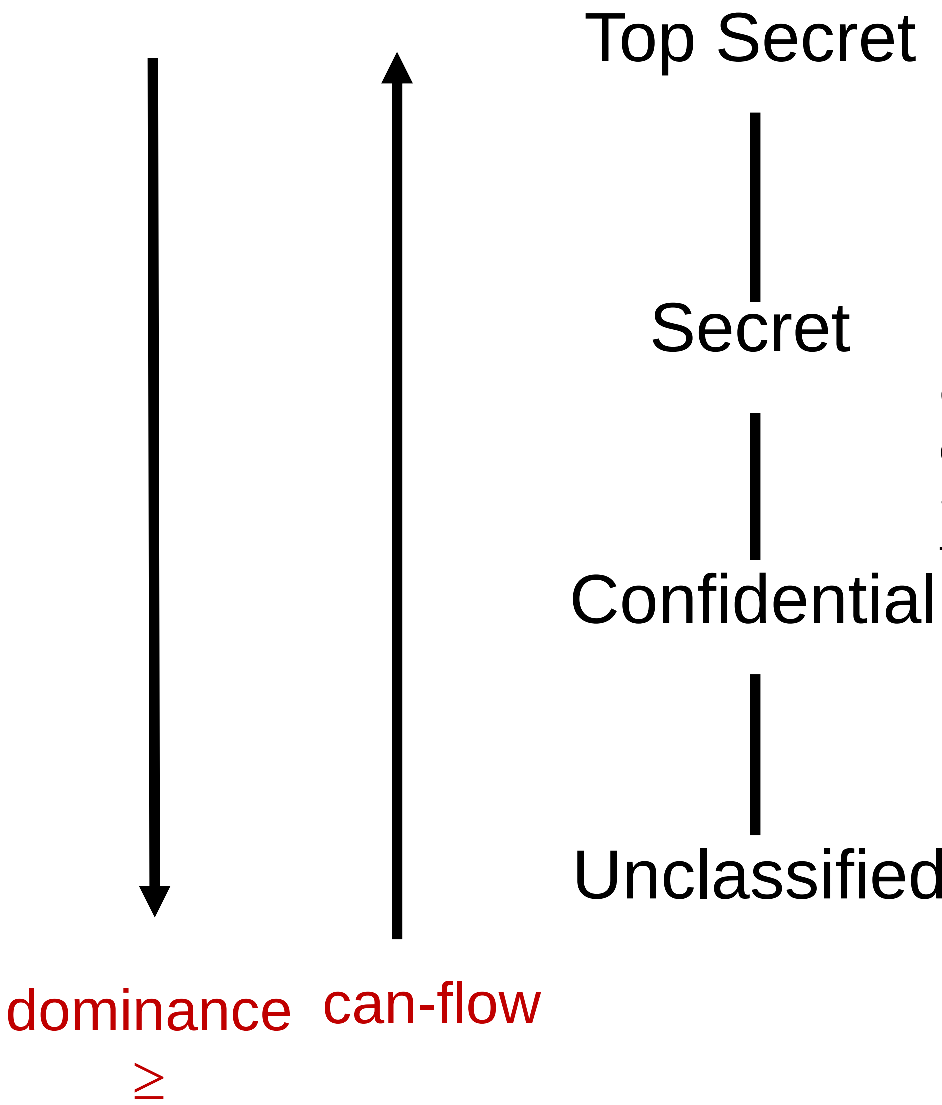
- Users can self manage access privileges.
- The burden of security administrators is significantly reduced, as resource users and administrators jointly manage permission.
- Per-user granularity for individual access decisions as well as coarse-grained access for groups are supported.
- It is easy to change privileges.
- Supporting new privileges is easy.

- Disadvantages?
  - Difficult to enforce a system-wide security policy, e.g.: A user can leak classified documents to a unclassified users.
- Only based user's identity and ownership, Ignoring security relevant info such as
  - User's role, Function of the program, Trustworthiness of the program
- Compromised program can change access to the user's objects
- It is difficult to judge the "reasonable rights" for a user or group.
- Inconsistencies in policies are possible due to individual delegation of permission.

Mandatory Access Control (MAC)

- Defined by three major properties:
  - Administratively-defined security policy
  - Control over all subjects (process) and objects (files, sockets, network interfaces)
  - Decisions based on all security-relevant info
- MAC
  - by assigning security levels to users and objects'
  - Access to an object is granted only if the security levels of the subject and the object satisfy certain constraints.
- The MAC pattern is also known as multilevel security model and lattice-based access control.

Mandatory Access Control (MAC)



Top Secret

Secret

Confidential

Unclassified

dominance
≥

can-flow

Labeling Mechanism is used

Prevent any illegal flow of information through the enforcement of multilevel security

Military Security
Classification involves sensitivity levels, compartments
•Do not let classified information leak to unclassified files

Require a strict classification of subjects and objects in security levels

Drawback of being too rigid

Applicable only to very few environments

Bell-LaPadula Model: Multi-level Security

- Introduced in 1973
- Air Force was concerned with security in time-sharing systems
  - Many OS bugs
  - Accidental misuse
- Main Objective:
  - Enable one to formally show that a computer system can securely process classified information

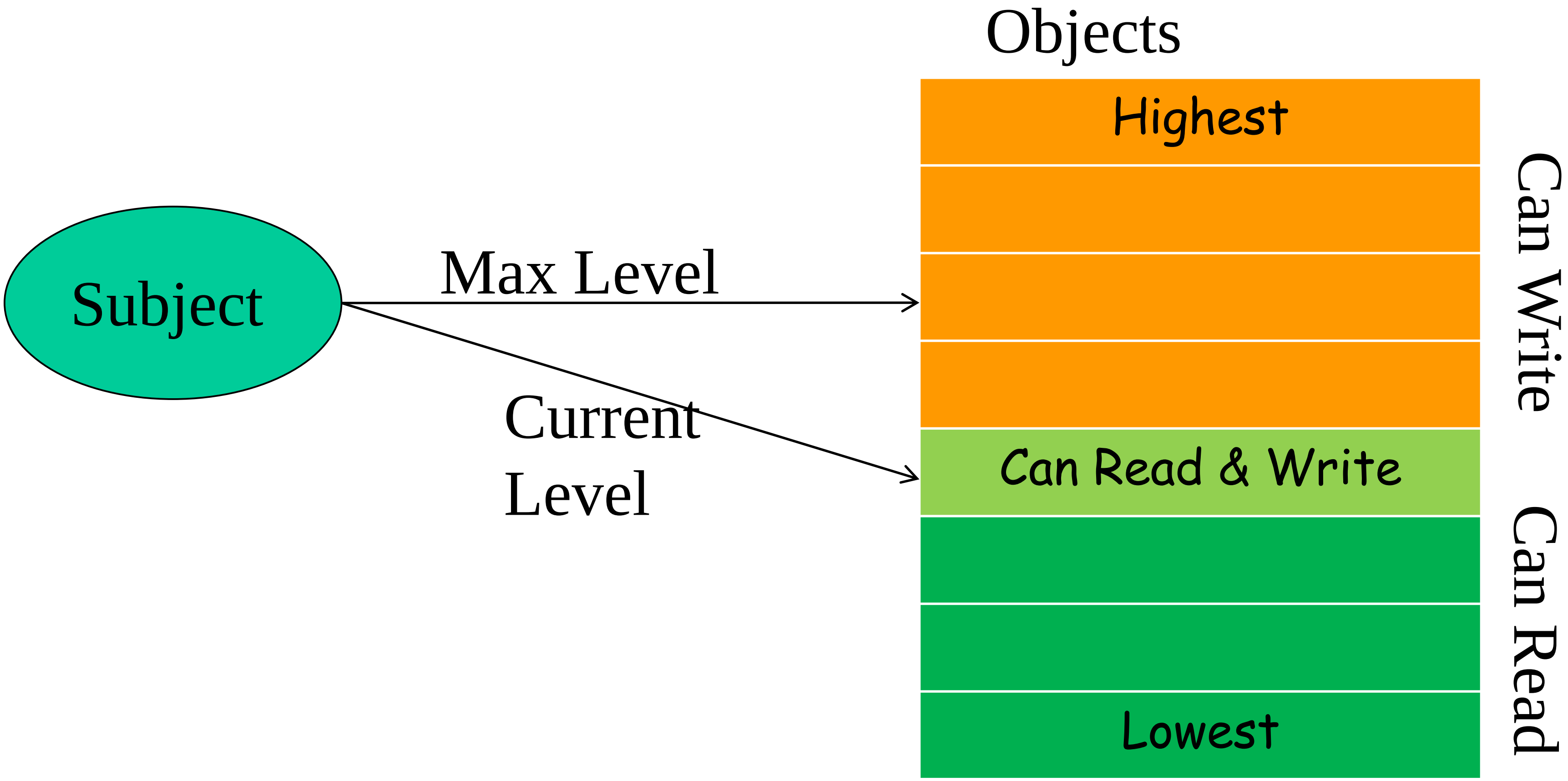
The BLP Security Model

- A computer system is modeled as a state-transition system
  - There is a set of subjects; some are designated as trusted.
  - Each state has objects, an access matrix, and the current access information.
  - There are state transition rules describing how a system can go from one state to another
  - Each subject s has a maximal sec level $L_m(s)$, and a current sec level $L_c(s)$
  - Each object has a classification level

The BLP Security Policy

- A state is secure if it satisfies
  - Simple Security Condition (no read up):
    - S can read O iff $L_m(S) \geq L(O)$
  - The Star Property (no write down): for any S that is not trusted
    - S can read O iff $L_c(S) \geq L(O)$ (no read up)
    - S can write O iff $L_c(S) \leq L(O)$ (no write down)
  - Discretionary-security property
    - every access is allowed by the access matrix
- A system is secure if and only if every reachable state is secure.

Implication of the BLP Policy



- Thanks