Access Control Matrix

A popular implementation of access control policy.

- One dimension consists of identified subjects that may attempt 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

Objects

Subjects	File 1	File 2	File 3	File 4
User A	Read Write Execute		Read Write Execute	
User B	Read	Read Write Execute	Write	Read
User C	Read Write	Read		Read Write Execute

Update Access Control Matrix

Possible changes over Access Control Matrix

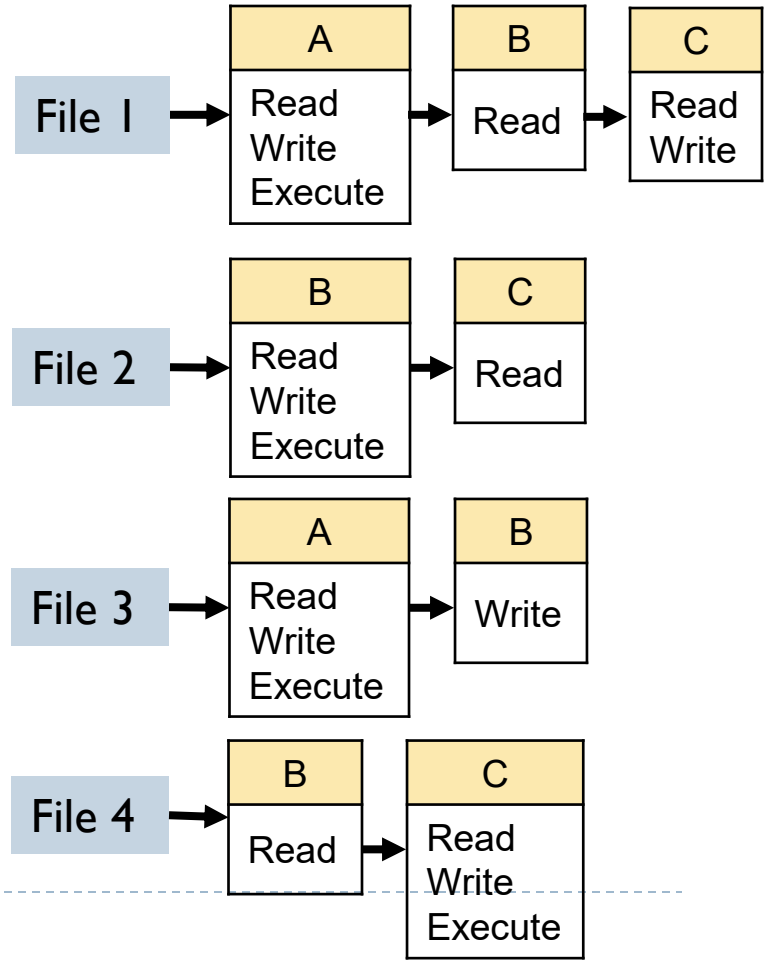
- Create subject s: add a new row *s*. This is typically done by the system administrator.
- Create object o: create a new column *o*. This is typically done by the system administrator.
- Grant permission r for subject s over object o: enter *r* to entry $M_{s,o}$. This is typically done by the resource owner or system administrator.
- Revoke permission r for subject s over object o: delete *r* from entry $M_{s,o}$. This is typically done by the resource owner or system administrator.
- Destroy subject s: delete the row *s*. This is typically done by the system administrator.
- Destroy object o: deletes the column o. This is typically done by the system administrator.

Access Control List (ACL)

In practice, an access control matrix is usually sparse and can be implemented by decomposition in one of two ways

Decomposition by columns

- For each object, ACL lists users their permitted access rights.
- ACL is convenient when determining which subjects have which access to a particular resource.

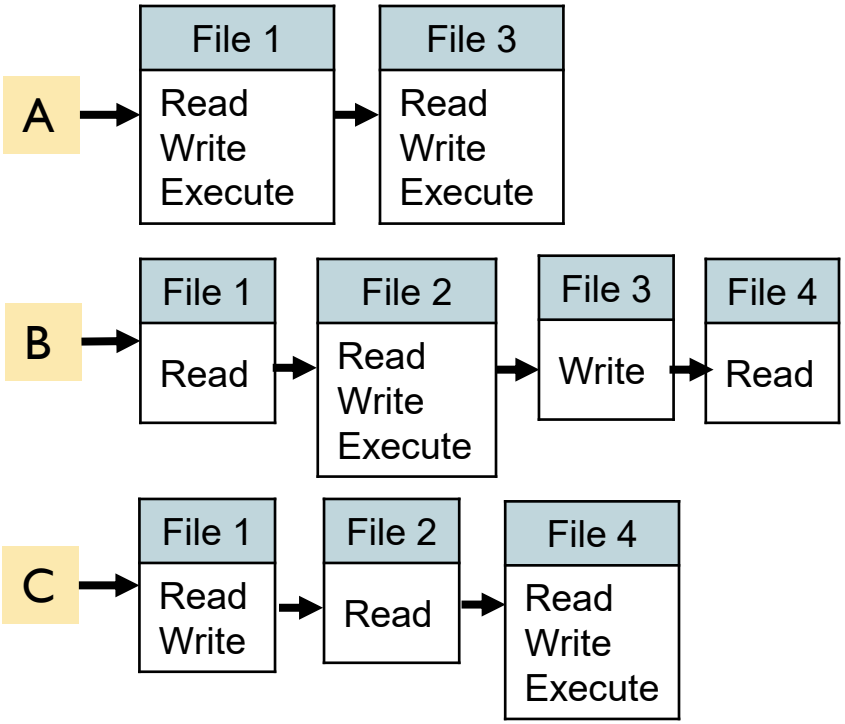


Capability List (C-List)

In practice, an access control matrix is usually sparse and can be implemented by decomposition in one of two ways

Decomposition by rows

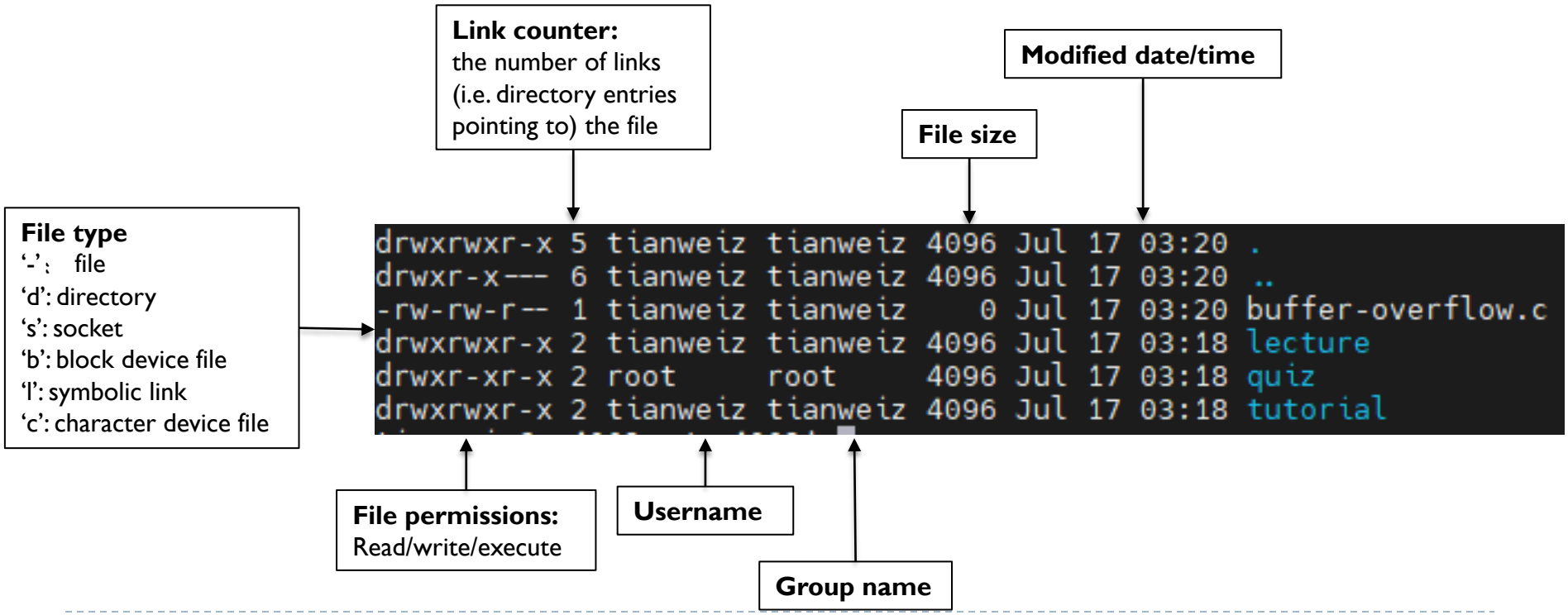
- C-list specifies authorized objects and operations for a particular user.
- C-List is convenient when determining the access rights available to a specific user.



Example: Resource Management in Unix OS

Files, directories, memory devices, I/O devices are uniformly treated as resources

- These resources are the objects of access control.
- Each resource has a single user owner and group owner



Permission Representation

Three permissions with three subjects

- Read, Write, Execute
- Owner, Group, Other
- Examples:
  - rw-r--r--: read and write access for owner, read access for group and other.
  - rwx------: read, write, and execute access for owner, no rights to group and other.

Octal Representation

- rw-r--r--: 110 100 100: 644
- rwx------: 111 000 000: 700

Adjust permission:

- Users can change the permissions:
  - chmod 754 filename
  - chmod u+wrx,g+rx,g-w,o+r,o-wx filename
- root can change the ownerships:
  - chown user:group filename

Controlled Invocation

Superuser privilege is required to execute certain OS functions

- Example: password changing
  - User passwords are stored in the file /etc/shadow
  - This file is owned by the root superuser. A normal user has no access to it
  - When a normal user wants to change his password with the program passwd, this program needs to give him additional permissions to write to /etc/shadow

SUID: a special permission flag for a program

- If SUID is enabled, then user who executes this progam will inherit the permissions of the program's owner.
- A normal user executing passwd can get additional root permission to write the new password to /etc/shadow

The execute permission of the owner is given as s instead of x

```
root@cx4062:~# ls -al /usr/bin/passwd
-rwsr-xr-x 1 root root 59976 Mar 14 08:59 /usr/bin/passwd
```