
# UNIT IV
# File Allocation & File Protection

S.Rajarajan

School of Computing

SASTRA

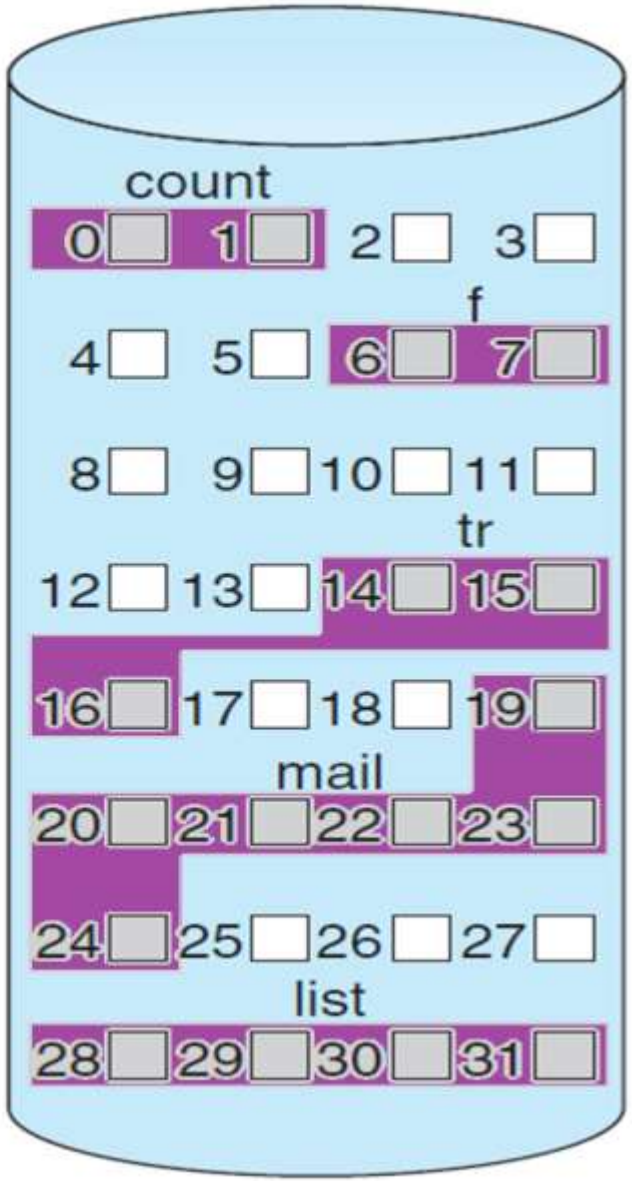
- The allocation methods define how the files are stored in the disk blocks
- There are three main disk space or file allocation methods.
  - Contiguous Allocation
  - Linked Allocation
  - Indexed Allocation
- The main idea behind these methods is to provide:
  - Efficient disk space utilization.
  - Fast access to the file blocks.
- All the three methods have their own advantages and disadvantages

# File Allocation Methods

| | Contiguous | Chained | Indexed | |
|---|---|---|---|---|
| **Preallocation?** | Necessary | Possible | Possible | |
| **Fixed or variable size portions?** | Variable | Fixed blocks | Fixed blocks | Variable |
| **Portion size** | Large | Small | Small | Medium |
| **Allocation frequency** | Once | Low to high | High | Low |
| **Time to allocate** | Medium | Long | Short | Medium |
| **File allocation table size** | One entry | One entry | Large | Medium |

# Contiguous file allocation

- Contiguous allocation requires that each file occupy a set of **contiguous blocks on the disk** like an array
- **Disk addresses** define a **linear ordering** on the disk.
- If the file is **n blocks long** and **starts at** location **b**, then it occupies **blocks b, b + 1, b + 2, ..., b + n – 1**.
- This means that given the **starting block address** and the **length of the file** (in terms of **blocks required**), we can determine the blocks occupied by the file.
- The directory entry for a file with contiguous allocation contains
  - Address of starting block
  - Length of the allocated portion.



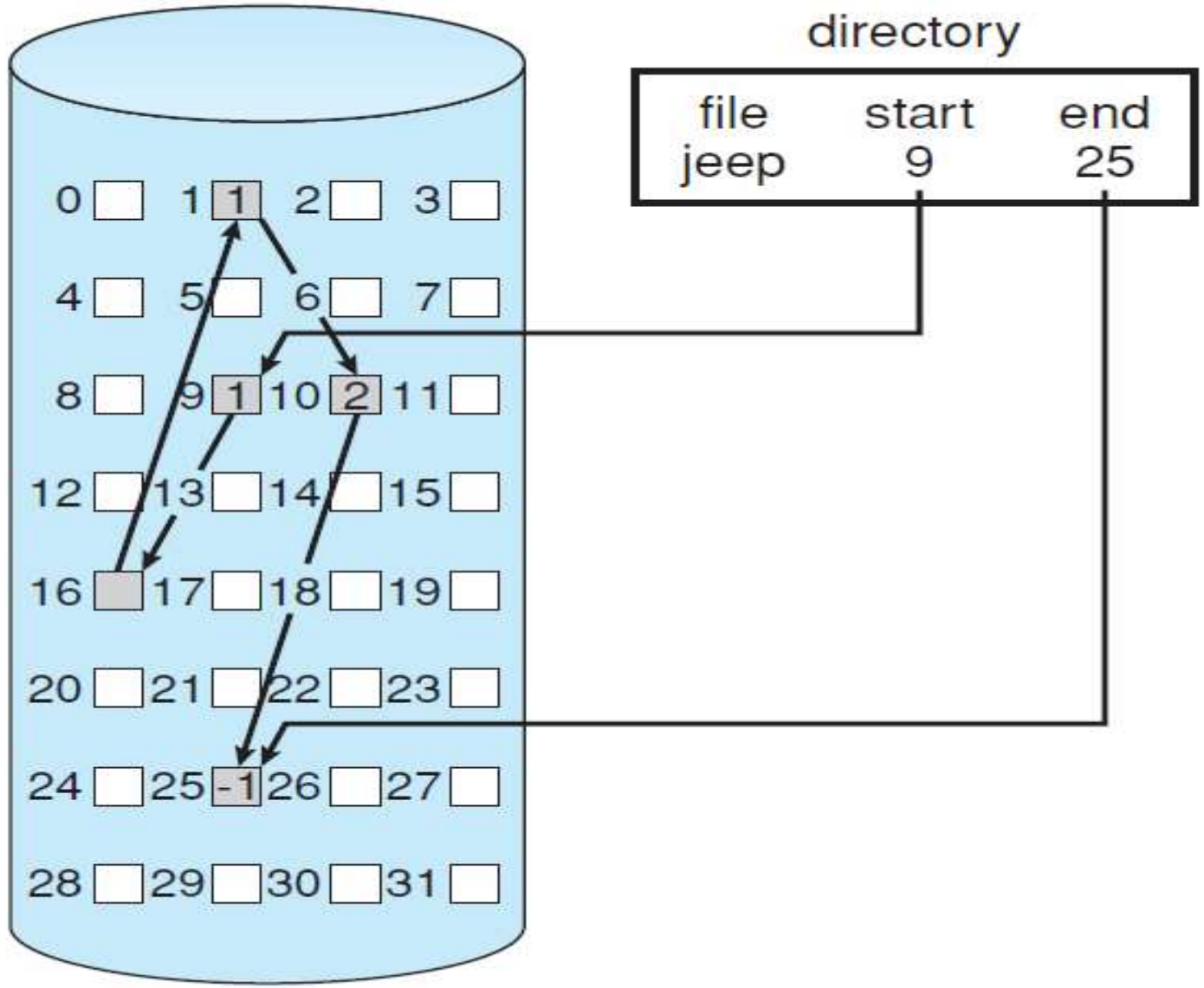

directory

| file | start | length |
|---|---|---|
| count | 0 | 2 |
| tr | 14 | 3 |
| mail | 19 | 6 |
| list | 28 | 4 |
| f | 6 | 2 |

- **Advantages:**
- It Is the best from the point of view of the individual sequential file
- **Multiple blocks can be read** at a time to improve **IO performance** for sequential processing
- Both the **Sequential and Direct Accesses** are supported by this.
- For direct access, the address of the **$k^{th}$ block** of the file which **starts at block b** can easily be obtained as **(b+k).**
- **Disadvantages**
- One difficulty is **finding space** for a new file.
- Increasing file size is difficult because it depends on the availability of contiguous memory
- Both **internal and External fragmentation** will occur.
- It will be necessary to do **compaction**

# Linked (chained) allocation

- In this scheme, each file is a linked list of disk blocks which **need not be** contiguous.
- The disk blocks **can be scattered** anywhere on the disk.
- Each block contains a **pointer to the next block** in the chain
- The directory entry contains a pointer to the starting and the ending file block.
- Better for sequential files

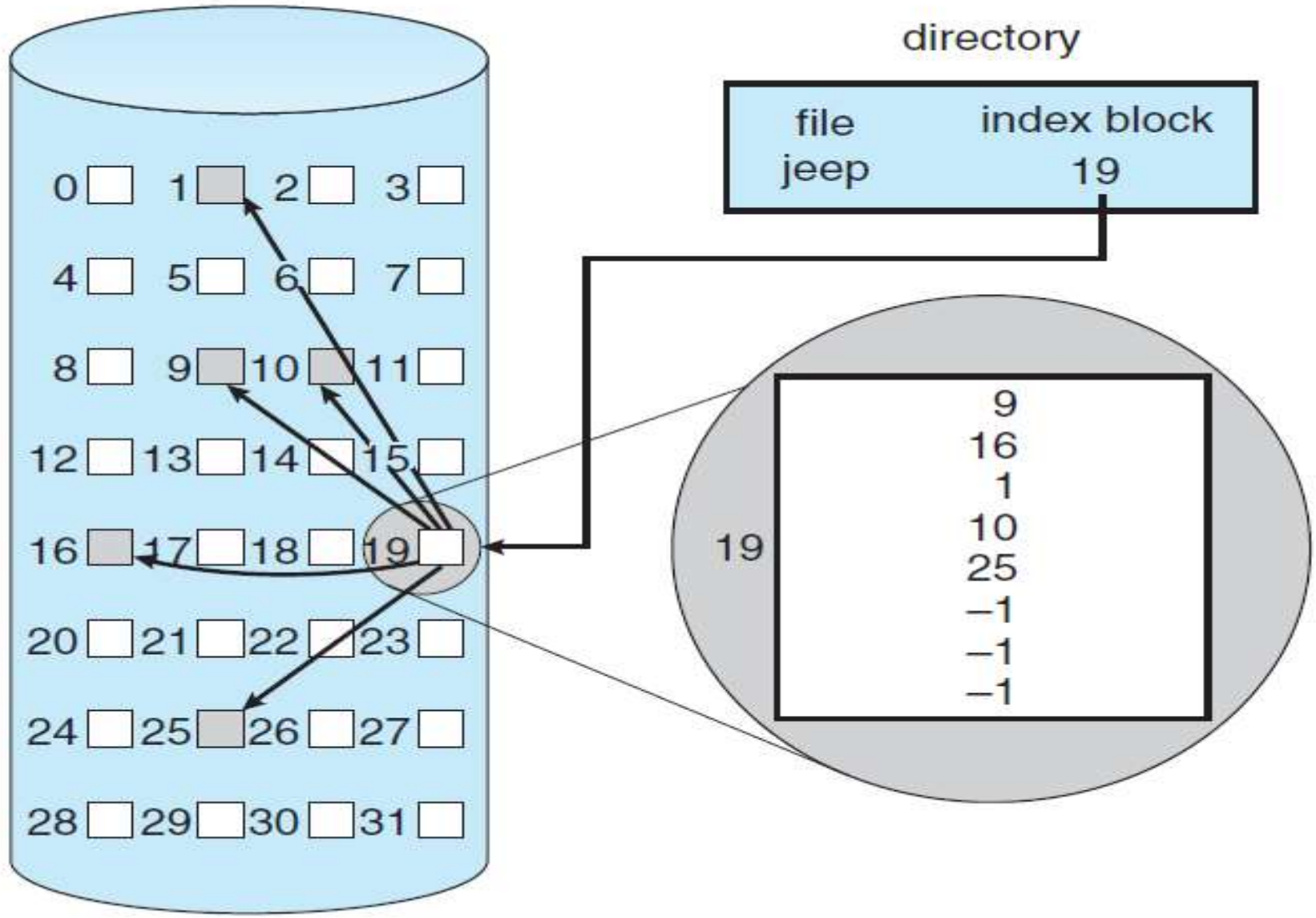

directory
file
start
end
jeep
9
25
0
1
1
2
3
4
5
6
7
8
9
1
10
2
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
-1
26
27
28
29
30
31

- **Advantage**
- File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
- This method does not suffer from **external fragmentation.**
- **Disadvantage**
- To select an individual block of a file requires **tracing through the chain** to the desired block
- **No accommodation** of **principle of locality.** Thus it is necessary to bring in several blocks of a file at a time, then a series of accesses to different parts of the disk are required
- This method does not suffer from external fragmentation.

- Another disadvantage is the **space** required for the **pointers.**
- If a pointer requires **4 bytes** out of a **512-byte block**, then **0.78 percent of the disk** is being used for pointers, rather than for information.
- The usual solution to this problem is to collect **blocks into multiples**, called **clusters,** and to **allocate clusters rather than blocks**.
- For instance, the file system may define a **cluster as four blocks** and operate on the disk only **in cluster units**.
- Pointers then use a **much smaller percentage** of the **file's disk space**.
- Yet another problem of linked allocation is **reliability.**
- A **bug in the operating-system** software or a disk hardware failure might result in picking up the wrong pointer

# Indexed allocation

- In this scheme, a special block known as the **Index block** contains the **pointers to all the blocks** occupied by a file.
- Each **file** has its **own index block**. The **ith entry** in the index block contains the disk address of the **ith file block**
- Typically the file indexes are **not physically stored** as part of the **file allocation table.**
- Rather the file index for a file is **kept in a separate block**, and the entry for the file in the file allocation table **points to that block**.
- Allocation may be based on **Fixed or Variable**
- **File consolidation** may be done from time to time


directory
file
jeep
index block
19
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
19
9
16
1
10
25
−1
−1
−1

- **Advantage**
- This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.
- It overcomes the problem of external fragmentation
- **Disadvantages:**
- The pointer overhead for indexed allocation is greater than linked allocation.
- For **very small files**, say files that expand **only 2-3 blocks**, the indexed allocation would keep **one entire block (index block) for the pointers** which is inefficient in terms of memory utilization. However, in linked allocation we lose the space of only 1 pointer per block.

# Protection

- Three requirements – **confidentiality, integrity, availability**
- When information is stored in a computer system, we want to keep it **safe from physical damage** (the **issue of reliability)** and **improper access** (the **issue of protection**).
- Reliability is generally provided by duplicate **copies of files**.
- File systems can **be damaged by hardware problems** (such as errors in reading or writing), power surges or failures, head crashes, dirt, temperature extremes, and vandalism.
- Files may be **deleted accidentally**.
- **Bugs in the file-system** software can also cause file contents to be lost

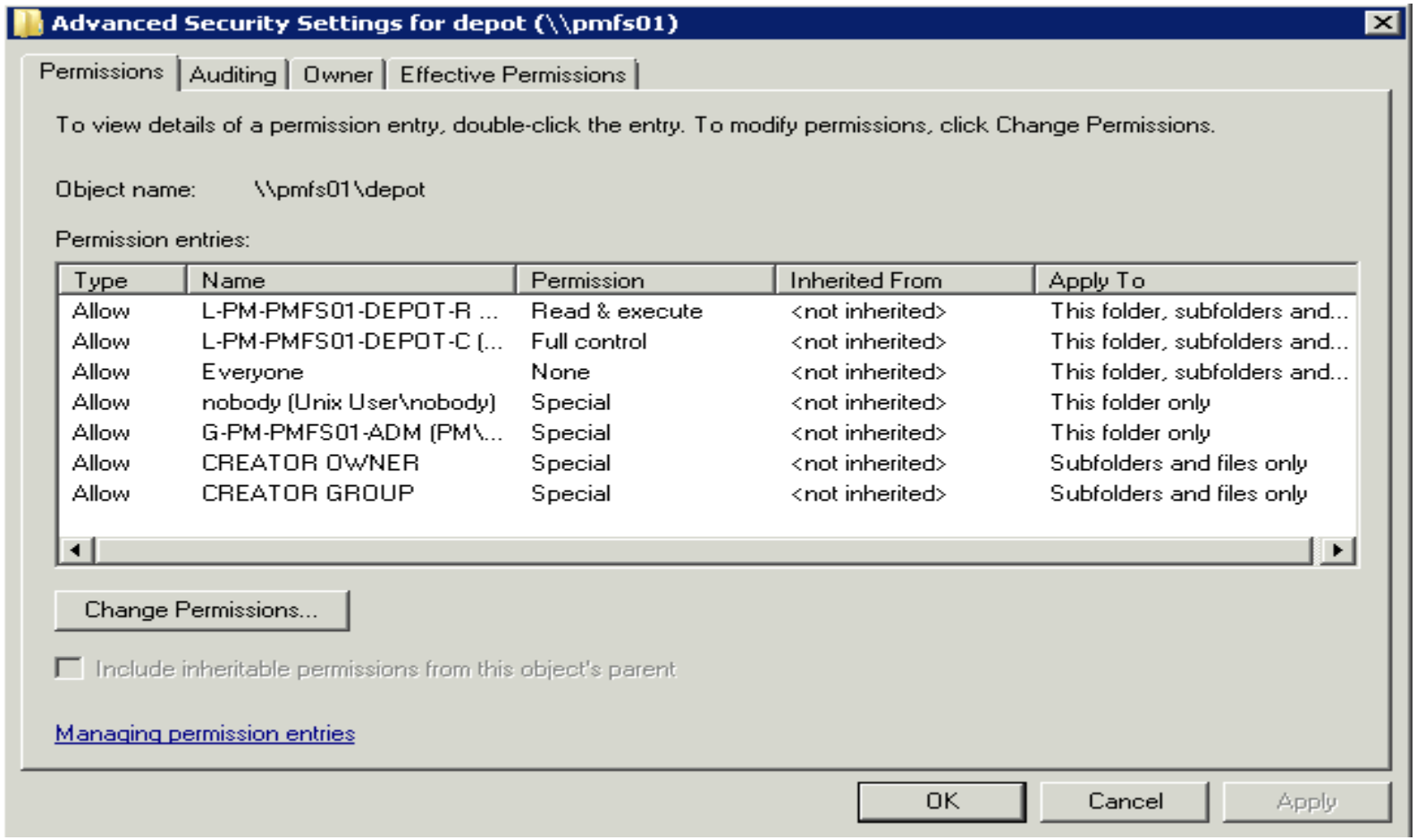
# Types of Access

- The need to protect files is a direct result of the **ability to access files.**
- Systems that **do not permit access** to the files of **other users** do not need protection.
- Thus, we could provide complete protection by **prohibiting access**.
- Alternatively, we could **provide free access** with **no protection.**
- What is needed is **controlled access.**
- Protection mechanisms **provide controlled access** by **limiting the types of file access** that can be made

- Several different types of operations may be controlled:
  - **Read**. Read from the file.
  - **Write**. Write or rewrite the file.
  - **Execute**. Load the file into memory and execute it.
  - **Append**. Write new information at the end of the file.
  - **Delete**. Delete the file and free its space for possible reuse.
  - **List**. List the name and attributes of the file.
- Other operations, such as **renaming**, **copying**, and **editing** the file, may also be **controlled.**

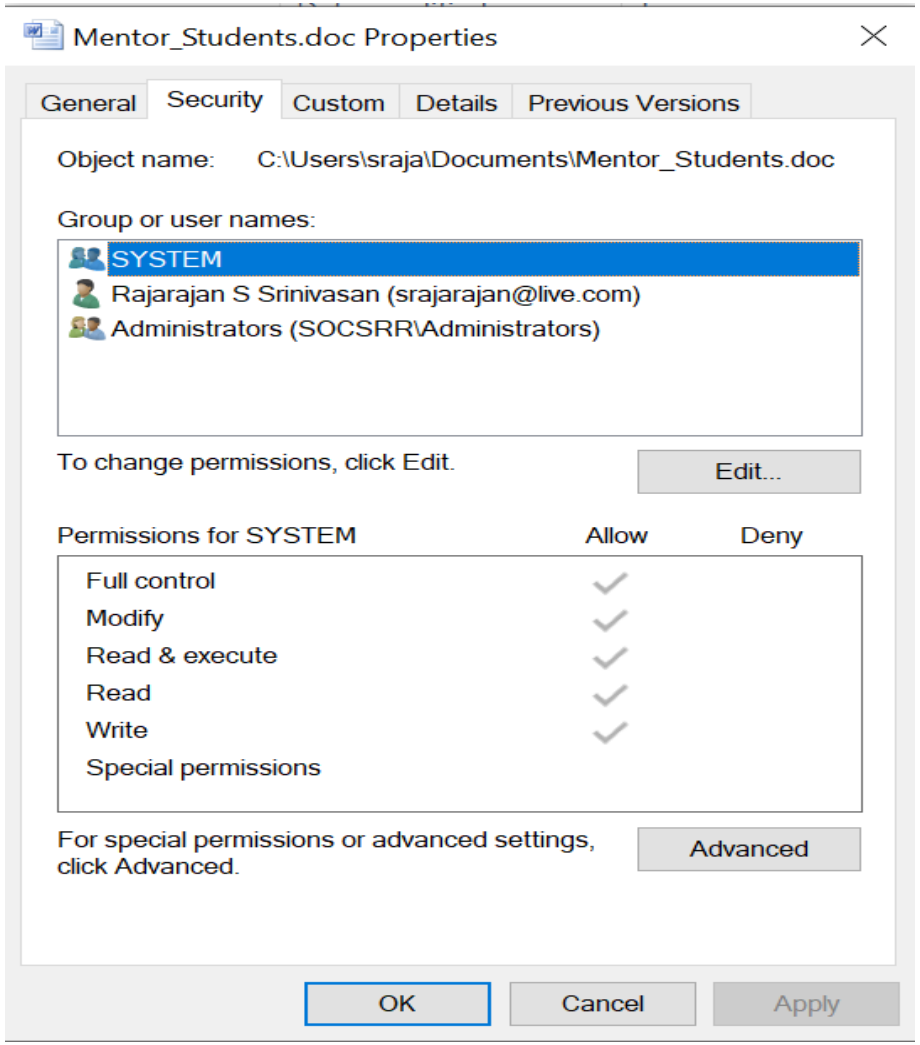
# Access Control

- The most common approach to the protection problem is to **make access dependent** on the **identity of the user**.
- **Different users** may need **different types of access** to a file or directory.
- The most general scheme to implement **identity dependent access** is to associate with each file and directory an **access-control list (ACL) specifying user names** and the **types of access** allowed for each user
- When a user requests access to a particular file, the **operating system checks the access list** associated with that file.
- If that user is listed for the requested access, the **access is allowed**.
- Otherwise, a **protection violation occurs**, and the user job is **denied access** to the file.


Advanced Security Settings for depot (\\pmfs01)
Permissions
Auditing
Owner
Effective Permissions
To view details of a permission entry, double-click the entry. To modify permissions, click Change Permissions.
Object name: \\pmfs01\depot
Permission entries:
Type
Name
Permission
Inherited From
Apply To
Allow L-PM-PMFS01-DEPOT-R ... Read & execute <not inherited> This folder, subfolders and...
Allow L-PM-PMFS01-DEPOT-C (... Full control <not inherited> This folder, subfolders and...
Allow Everyone None <not inherited> This folder, subfolders and...
Allow nobody (Unix User\nobody) Special <not inherited> This folder only
Allow G-PM-PMFS01-ADM (PM\... Special <not inherited> This folder only
Allow CREATOR OWNER Special <not inherited> Subfolders and files only
Allow CREATOR GROUP Special <not inherited> Subfolders and files only
Change Permissions...
Include inheritable permissions from this object's parent
Managing permission entries
OK
Cancel
Apply

- To condense the length of the access-control list, many systems recognize three classifications of users in connection with each file:
  - **Owner**. The user who created the file is the owner.
  - **Group**. A set of users who are sharing the file and need similar access is a group, or work group.
  - **Universe**. All other users in the system constitute the universe.

| | | | | | |
|---|---|---|---|---|---|
| -rw-rw-r-- | 1 pbg | staff | 31200 | Sep 3 08:30 | intro.ps |
| drwx------ | 5 pbg | staff | 512 | Jul 8 09.33 | private/ |
| drwxrwxr-x | 2 pbg | staff | 512 | Jul 8 09:35 | doc/ |
| drwxrwx--- | 2 jwg | student | 512 | Aug 3 14:13 | student-proj/ |
| -rw-r--r-- | 1 pbg | staff | 9423 | Feb 24 2012 | program.c |
| -rwxr-xr-x | 1 pbg | staff | 20471 | Feb 24 2012 | program |
| drwx--x--x | 4 tag | faculty | 512 | Jul 31 10:31 | lib/ |
| drwx------ | 3 pbg | staff | 1024 | Aug 29 06:52 | mail/ |
| drwxrwxrwx | 3 pbg | staff | 512 | Jul 8 09:35 | test/ |


Mentor_Students.doc Properties
General
Security
Custom
Details
Previous Versions
Object name: C:\Users\sraja\Documents\Mentor_Students.doc
Group or user names:
SYSTEM
Rajarajan S Srinivasan (srajarajan@live.com)
Administrators (SOCSRR\Administrators)
To change permissions, click Edit.
Edit...
Permissions for SYSTEM
Allow
Deny
Full control
Modify
Read & execute
Read
Write
Special permissions
For special permissions or advanced settings, click Advanced.
Advanced
OK
Cancel
Apply

# Other Protection Approaches

- Another approach to the protection problem is to associate a **password.**
- The use of passwords has a few disadvantages, however.
- First, the **number of passwords** that a user needs to **remember** may become large, making the scheme impractical.
- Second, if **only one password** is used for all the files, then once it is discovered, all files are accessible.

- In a multilevel directory structure, we need to protect not only individual files but also **collections of files in subdirectories**.
- We need to provide a mechanism for **directory protection.**
- We want to control the **creation and deletion** of **files in a directory**.