1. Describe with a neat sketch about the various directory structure.

A directory is a container that is used to contain folders and files. It organizes files and folders in a hierarchical manner. [There are several logical structures of a directory, these are given below1](https://www.geeksforgeeks.org/structures-of-directory-in-operating-system/)[2](https://www.javatpoint.com/os-directory-structure)[3](https://en.wikipedia.org/wiki/Directory_structure):

**Single-level directory**

The single-level directory is the simplest directory structure. In it, all files are contained in the same directory which makes it easy to support and understand. A single level directory has a significant limitation, however, when the number of files increases or when the system has more than one user. Since all the files are in the same directory, they must have a unique name. If two users call their dataset test, then the unique name rule violated.

| **Advantages** | **Disadvantages** |
| --- | --- |
| Easy to implement | Name collision |
| Fast searching for small directories | Difficult to group files by type or owner |
| Simple operations on files |  |

**Two-level directory**

As we have seen, a single level directory often leads to confusion of files names among different users. The solution to this problem is to create a separate directory for each user. In the two-level directory structure, each user has their own user files directory (UFD). The UFDs have similar structures, but each lists only the files of a single user. The system’s master file directory (MFD) is searched whenever a new user logs in.

| **Advantages** | **Disadvantages** |
| --- | --- |
| No name collision | Fixed number of users |
| Easy to find files for a user | No grouping within a user’s directory |
| Better security and privacy |  |

**Tree-structured directory**

The tree-structured directory allows users to create their own subdirectories and to organize their files accordingly. A tree is a data structure that consists of nodes connected by edges. A tree has one root node and can have zero or more child nodes. Each child node can have zero or more child nodes, and so on. The tree-structured directory has one root directory, and every file in the system has a unique path name.

| **Advantages** | **Disadvantages** |
| --- | --- |
| No name collision | Path names can be long |
| Flexible grouping of files | Moving files and directories can be complex |
| Hierarchical organization of files |  |

**Acyclic-graph directory**

The acyclic-graph directory allows directories to share subdirectories and files. This is useful when two different users or groups want to access the same file without making copies of it. The acyclic-graph directory is similar to the tree-structured directory, except that it allows links or pointers to other directories or files. A link is essentially a pointer to another file or subdirectory.

| **Advantages** | **Disadvantages** |
| --- | --- |
| No duplication of files | Difficult to maintain consistency |
| Efficient use of disk space | Possible aliasing problems |
| Multiple access paths for files |  |

Here is a neat sketch of the various directory structures:

Single-level:

+-----------------+

| MFD |

+-----------------+

| File1 |

| File2 |

| File3 |

+-----------------+

Two-level:

+-----------------+

| MFD |

+-----------------+

| UFD1 -> +-----> +-----------------+

| UFD2 -> +-----> | UFD2 |

+-----------------+ +-----------------+

| File4 |

| File5 |

+-----------------+

Tree-structured:

+-----------------+

| Root |

+-----------------+

/ \

/ \

/ \

+---+ +---+

| A | | B |

+---+ +---+

/ \ / \

+---+ +---+

| C | | D |

+---+ +---+

Acyclic-graph:

+-----------------+

| Root |

+-----------------+

/ \

/ \

/ \

+---+ +---+

| A +---->+ B +<----+

+---+ +---+ |

/ \ / \ |

+---+ +---+ v

| C +---->+ D +---->+ E +

+---+ +---+ +---+

 2.   Discuss about the various file access methods and draw neat sketch explain about the:

a) Directory structure b) File sharing

File access methods define how data is accessed and modified within a file. There are different file access methods with their own set of strengths and limitations. [The four primary file access methods are1](https://www.javatpoint.com/os-file-access-methods)[2](https://www.tutorialspoint.com/file-access-method)[3](https://www.electronicsmind.com/file-access-methods-in-operating-system/):

**Sequential access method**

In this method, information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editor and compiler usually access the file in this fashion. Read and write operations are performed sequentially and the file pointer is automatically advanced to the next record.

| **Advantages** | **Disadvantages** |
| --- | --- |
| Simple to implement | Slow access to specific records |
| Suitable for applications that require sequential processing | Difficult to insert or delete records |
| Efficient for large files | Wasted space for variable-length records |

**Direct access method**

In this method, information can be accessed directly without reading through the entire file. Each record has a unique address or key that can be used to locate it. The file pointer can be moved to any position within the file. This method allows for random access and faster retrieval of records.

| **Advantages** | **Disadvantages** |
| --- | --- |
| Fast access to specific records | Complex to implement |
| Easy to insert or delete records | Requires extra space for storing keys or addresses |
| Suitable for applications that require random processing | Prone to data corruption or fragmentation |

**Indexed access method**

In this method, information can be accessed directly using an index that maps keys or addresses to records. The index is a separate file that contains pointers to the actual records. The index can be searched quickly to find the location of a record. This method combines the benefits of sequential and direct access methods.

| **Advantages** | **Disadvantages** |
| --- | --- |
| Fast access to specific records | Requires extra space and time for maintaining index |
| Easy to insert or delete records | Index can become outdated or corrupted |
| Suitable for applications that require both sequential and random processing | Complex to implement |

**Indexed sequential access method**

In this method, information can be accessed sequentially or directly using an index that maps keys or addresses to records. The index is a part of the file that contains pointers to the actual records. The index is sorted in ascending order of keys or addresses. This method improves the performance of sequential access method by using an index.

| **Advantages** | **Disadvantages** |
| --- | --- |
| Fast access to specific records | Requires extra space for storing index |
| Easy to insert or delete records | Index can become outdated or corrupted |
| Suitable for applications that require both sequential and random processing | Complex to implement |

Here is a neat sketch of the various file access methods:

Sequential access:

+-----------------+

| File |

+-----------------+

| Record1 +-----> File pointer

| Record2 |

| Record3 |

+-----------------+

Direct access:

+-----------------+

| File |

+-----------------+

| Record1 +<----- Key1

| Record2 +<----- Key2

| Record3 +<----- Key3

+-----------------+

Indexed access:

+-----------------+ +-----------------+

| Index | | File |

+-----------------+ +-----------------+

| Key1 -> +---->| Record1 |

| Key2 -> +---->| Record2 |

| Key3 -> +---->| Record3 |

+-----------------+ +-----------------+

Indexed sequential access:

+-----------------+

| File |

+-----------------+

| Index +-----> Sorted keys/addresses

+-----------------+

/ \

/ \

/ \

+---+ +---+

| A +---->+ B +----> Records

+---+ +---+