# Study Set for Lecture 11: File System Interface

**Due** Nov 9 at 11:59pm **Points** 10 **Questions** 13
**Available** Nov 3 at 12pm - Dec 8 at 8pm about 1 month **Time Limit** None
**Allowed Attempts** Unlimited

## Instructions

Review lecture notes from **lect11_File_System_Interface.pdf**. Please download the accompanying code from **lect11code.zip**.

Then answer the questions from this study set and submit them to gain access to the further part of the course.

Take the Quiz Again

## Attempt History

| | Attempt | Time | Score |
|---|---|---|---|
| LATEST | Attempt 1 | 1,373 minutes | 10 out of 10 |

Correct answers are hidden.

Score for this attempt: **10** out of 10
Submitted Nov 6 at 11:26am
This attempt took 1,373 minutes.

### Question 1

**0 / 0 pts**

What kind of data are necessary for maintaining open files in operating systems. Where is the data kept?

Your Answer:

Information about files is kept in the directory structure; maintained by the OS on the disk. In order toopen a file, the OS needs to know:

□file position: pointer to last read/write position.

□file-open count: number of times a file has been opened.

□disk location of the file: cached in memory, so it doesn't need to be read from file all the time.

□access rights: per-process access mode information

## Question 2

0 / 0 pts

Describe the ways for an operating system to tell the type of a file?

Your Answer:

An OS can use an extension, internal meta-data, a magic number stored at the beginning of the file, or areference to creating the application to determine the type of file.

## Question 3

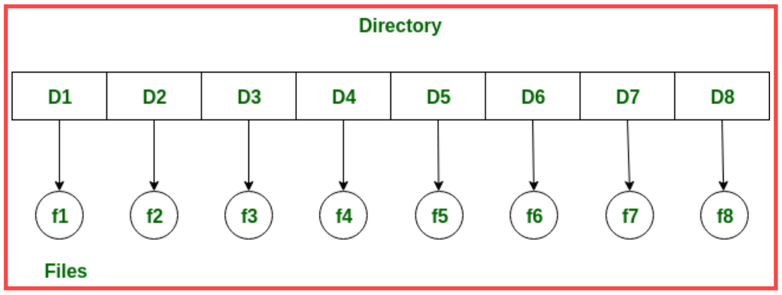
0 / 0 pts

Describe with details all types of directory structures that were discussed in the lecture.

Your Answer:

Single-Level Directory

● A single directory for all users



● Many problems with:

- searching

-- plenty of files

- naming

-- no duplicates

● grouping

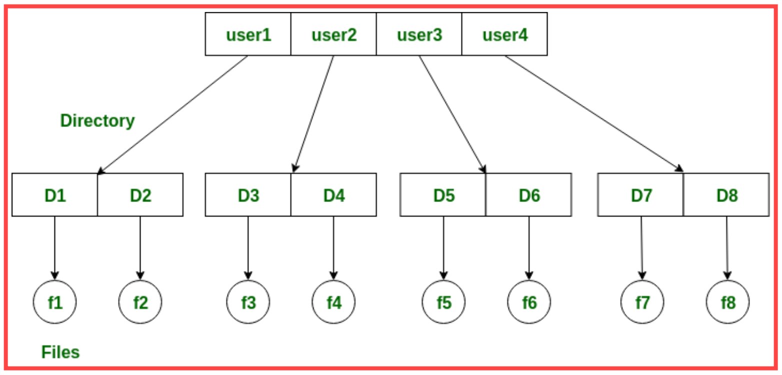
-- no support other as by name

● security

-- shared space

Two-Level Directory Structure

● Separate directory for each user



● Use a path name rather than just a file name

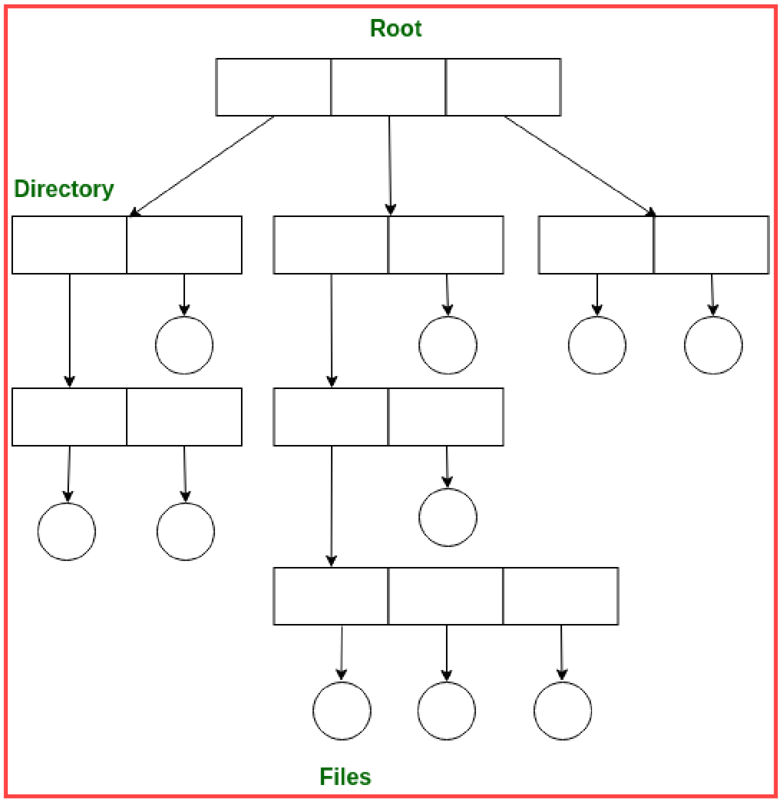
● Can have the same file name for different user

● Efficient searching per user

● ...but, still not enough flexibility

- e.g., no grouping capability

**Tree-structured directory –**
Once we have seen a two-level directory as a tree of height 2, the natural generalization is to extend the directory structure to a tree of arbitrary height.
This generalization allows the user to create there own subdirectories and to organize on their files accordingly.



A tree structure is the most common directory structure. The tree has a root directory, and every file in the system have a unique path.

**Advantages:**

- Very generalize, since full path name can be given.
- Very scalable, the probability of name collision is less.
- Searching becomes very easy, we can use both absolute path as well as relative.

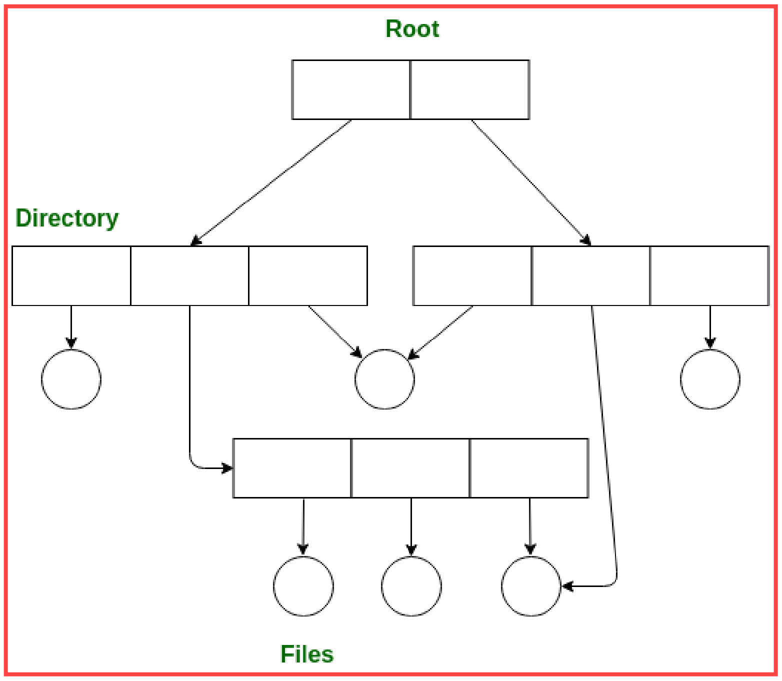
**Disadvantages:**

- Every file does not fit into the hierarchical model, files may be saved into multiple directories.
- We can not share files.
- It is inefficient, because accessing a file may go under multiple directories.

**Acyclic graph directory –**
An acyclic graph is a graph with no cycle and allows to share subdirectories and files. The same file or subdirectories may be in two different directories. It is a natural generalization of the tree-structured directory.

It is used in the situation like when two programmers are working on a joint project and they need to access files. The associated files are stored in a subdirectory, separating them from other projects and files of other programmers, since they are working on a joint project so they want the subdirectories to be into their own directories. The common subdirectories should be shared. So here we use Acyclic directories.

It is the point to note that shared file is not the same as copy file . If any programmer makes some changes in the subdirectory it will reflect in both subdirectories.



**Advantages:**

- We can share files.
- Searching is easy due to different-different paths.

**Disadvantages:**

- We share the files via linking, in case of deleting it may create the problem,
- If the link is softlink then after deleting the file we left with a dangling pointer.
- In case of hardlink, to delete a file we have to delete all the reference associated with it.

## Question 4

0 / 0 pts

Allowing a general graph to represent a directory structure has a number of pitfalls. Describe them along with potential solutions.

Your Answer:

he pitfall with allowing a general graph to represent a directory structure is the introduction of cycles, which can lead to infinite traversal of a directory. If a directory contains a link to itself or even another directory which leads to a cycle, then infinite traversal is a possibility. There is a way to guarantee no cycles though, if you make it so that files are the only things allowed to be linked, then that eliminates the possibility of cycles. Every time a new link is added use a cycle detection algorithm to determine whether it is ok. Unfortunately, this doesn't fix the problem with dangling pointers, which can be fixed using a reference count like acyclic graphs

## Question 5

**0 / 0 pts**

Thoroughly explain a difference between Unix soft links and hard links to files.

Your Answer:

## What is Soft Link And Hard Link In Linux?

A **symbolic** or **soft link** is an actual link to the original file, whereas a **hard link** is a mirror copy of the original file. If you delete the original file, the soft link has no value, because it points to a non-existent file. But in the case of hard link, it is entirely opposite. Even if you delete the original file, the hard link will still has the data of the original file. Because hard link acts as a mirror copy of the original file.

In a nutshell, a soft link

- can cross the file system,
- allows you to link between directories,

- has different inode number and file permissions than original file,
- permissions will not be updated,
- has only the path of the original file, not the contents.

A hard Link

- can't cross the file system boundaries (i.e. A hardlink can only work on the same filesystem),
- can't link directories,
- has the same inode number and permissions of original file,
- permissions will be updated if we change the permissions of source file,
- has the actual contents of original file, so that you still can view the contents, even if the original file moved or removed.
-

## Question 6

**0 / 0 pts**

Following the code distributed with the lecture notes and discussed during the lecture, write a program that recursively traverses the directory hierarchy of a file system and finds the largest file in the whole file system.

Will your program work if a general graph is used for the directory structure? How could the problem be addressed to solve the potential cycle problem?

Your Answer:

Once you obtain information about the files, and getting information from the current folder loop. The best way to do it would be a recursive call.

You would put the recursive call if you have an entry that is a directory. You know you are in a directory because DT_DIR: iis called in the loop, then you would call recursively the same function. This makes it guaranteed that I go through the whole hierarchy.

General Graph Directory

● We allow for any graph to represent a directory

● Problem with possibility of introducing cycles

● How can we guarantee no cycles?

- allow only links to files, and not directories

-- files are terminal nodes in the graph (still problem with dangling pointers)

- every time a new link is added use a cycle detection algorithm to determine whether it is OK

-- expensive!

To fix the cycle problem, we either need to remember the path, and check if the path is repeating itself. Or an algorithm that detects cycles in the whole file systems. Neither of them would be good solutions.... Its usually pushed on the user not to have cycle problems.

## Question 7

**0 / 0 pts**

1. Show Unix commands that change access modes to protect the following files:
   - file named "A" (the owner can do anything, your group can only read, nobody else can see)
   - file named "B" (everybody can execute, but only the owner can write)
   - file named "C" (everybody has total access to this non-executable)
2. Show a Unix command that changes the access mode to protect directory named "D", so only you can open it.

Your Answer:

A: chmod u+rwx A

chmod g+r A

chmod o-rwx A

B: chmod a+x B

chmod u+w B

C: chmod a+rw B

2.

chmod u+rw D

chmod g-rwx D

chmod o-rwx D

## Question 8

0 / 0 pts

Describe how to mount and unmount file systems on Unix. How to list the mounted file systems.

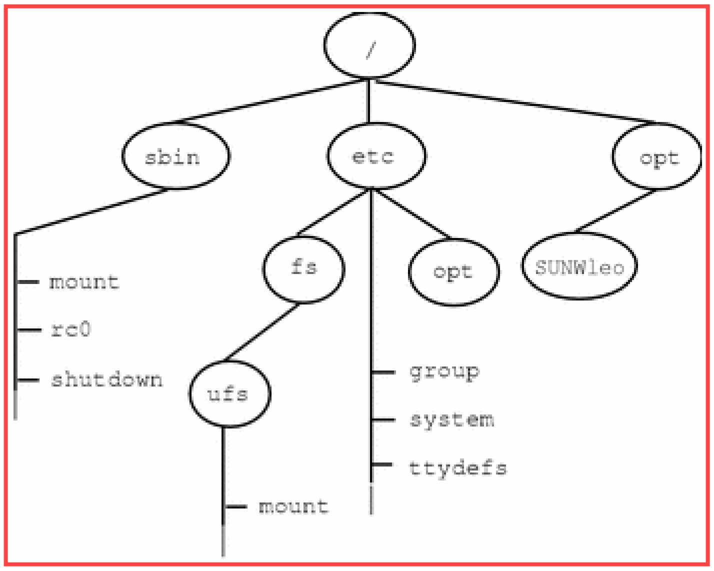
Hint: explore mount, umount, and df Unix commands.

Your Answer:

Before you can access the files on a file system, you need to mount the file system. Mounting a file system attaches that file system to a directory (**mount point**) and makes it available to the system. The root (`/`) file system is always mounted. Any other file system can be connected or disconnected from the root (`/`) file system.

When you mount a file system, any files or directories in the underlying mount point directory are unavailable as long as the file system is mounted. These files are not permanently affected by the mounting process, and they become available again when the file system is unmounted. However, mount directories are typically empty, because you usually do not want to obscure existing files.

For example, the figure below shows a local file system, starting with a root (`/`) file system and subdirectories `sbin`, `etc`, and `opt`.



## Question 9

**0 / 0 pts**

What is the difference between the following commands? Which one always requires super user access?

```
> chmod
> chgrp
> chown
```

Your Answer:

chmod changes the permissions of a file directory, chgrp changes group ownership, and chown changes files owner and group. chowm always needs super user access since you might not own the file

## Question 10

0 / 0 pts

A user executes the following commands in bash in an empty directory:

```
> umask 77
> touch test_file_A
> umask 27
> touch test_file_B
> umask 2
> touch test_file_C
```

What will be the system response to the following command:

```
> ls -l
```

Explain how did you derive the answer by describing the results of each command.

Your Answer:

unmask 77- determines default permissions for creating new files
touch test_file_A - creates test_file_A
unmask 27- determines default permissions for creating new files.
touch test_file_B - creates test_file_B
unmask 2- determines default permissions for creating new files
touh test_file_C - creates test_file_C

ls -l
this will display permissions and status

## Question 11

0 / 0 pts

Explain the meaning and ranslate the following access modes to their letter counterparts:

```
777
600
400
644
666
```

Your Answer:

777= rwx rwx rwx
600= rw- --- ---
400= -w- --- ---
644= rw- -w- -w-
666= rw- rw- rw-

-binary

## Question 12

0 / 0 pts

Explain the meaning and translate the following access modes to their octal counterparts:

```
rwx------
rwxr--r--
r--r--r--
rw-rw-rw-
rwxrwxrwx
rw--w--w-
```

Your Answer:

1. The owner has access to read,write, and execute, while the group and world has none
2. owner has access to read write and execute, group has read and world has read
3. everyone has read
4. everyone has read and write
5. everyone has access to read write and execute

6. owner has access to read and write. group and world has access to write

Jnanswered

## Question 13

10 / 10 pts

I have submitted answers to all questions in this study set.

- ○ True
- ○ False

Quiz Score: **10** out of 10