Introduction to File Systems

Hello, and welcome to this introductory lesson on File Systems.

The file system is an important component of the operating system, as it provides many important services to the rest of the system and to the user. The most important ones may be controlled access to structured data, persistence, and naming.

We will start by defining what a file is. We will do this in a very rather abstract fashion because some modern systems, primarily from the UNIX/Linux family, use file handles to access pretty much everything, ranging from devices, to system information, to bona-fide storage files, and others; and they like to call much of what can be accessed through a file handle, a "file".

We will then very briefly define the role of the file system. We will do this in a hurry because we will re-visit file systems by looking at examples.

We will take the opportunity to briefly discuss how files are accessed, which gives us some indication of the potential problems when we look at file system implementations.

We wrap up by looking at how files are organized from a user's perspective. We are so used today to see files a sequences of bytes, a file model that has been perpetuated primarily by UNIX, that we forget that high-performance file systems sometimes offer much more sophisticated file organizations, which are tailored for various different usage models.

At the end of this lesson we will have the basic terminology defined to look more closely at a number of file system implementation aspects, such as file allocation, directory management, and others.

Let's begin.

What is a file?

What is a file?

We are use to an informal use of computing system to see files as documents that appear on our disk, possibly inside some folders.

These persistent storage files are a very small subset of what we call files in general.

More generally, a file can be considered a collection of data elements, which are grouped together so that they can accessed and retrieved together, and possibly modified in a structured fashion.

In some cases, files are persistent and are stored on some form of physical storage device, in many cases non-volatile. We will see later that the file system provides a naming infrastructure, which allows these files to be accessed using an identifier.

In many systems, files represent anything that is a sequence, or a stream of data units. This can be an actual persistent file, or a pipe, or a stream of data from a device driver, or data coming in over a network connection.

<nothing>

[Slide 3]

The file system then is that part of the operating system that manages files and access to them.

This means that the file system creates, destroys, reads, modifies, and moves files.

It provides a naming service that allows users to identify files by providing a name.

The file system controls access to files, thus providing protection of file content by limiting operations on files to authorized users.

Finally, the file system manages how resources are allocated to file, for example, how storage device blocks are allocated to files.

Logical Organization of a file

How are files organized?

In general, a file is organized as a collection of “records”, denoted by R0 to Rn as seen in this example.

By a record, we mean a block of information that is transferred during a logical read or write operation.

Records can be either fixed or variable length.

<nothing>

File Access

This very simple representation of records in a file helps us distinguish between two ways to access data stored in records.

One is to access the records of a file “sequentially”, that is, one record after the other, in the order in which they are organized in the file.

This little program example illustrates sequential access to a file.

After we open the file, we read one record after the other until we reach the end of the file. The file object “f” internally keeps track of the next record to be read.

We start at the beginning of the file. As we read records, we traverse the file until we reach the end.

In so-called random access, on the other hand, we specify which record we want to read, typically by providing an integer denoting the index of the record in the file.

In this little program example, we keep reading literally a random record by specifying as index a random number modulo some number, say the number of records in the file.

Each time we read a record, it is a different one.

Logical Organization of a File

The question is, how are these records organized in a file? We are used to files as sequences of bytes, but some operating systems provide quite a variety of file organizations, for the user to choose, ranging from simple piles, to index sequential files, to indexed files and many others.

In this lesson we will look at a few possible file organizations.

Pile

The simplest organization is just a sequence of identical sized records, for example a sequence of bytes.

This is sometimes called a “pile file” and it is what we are used from UNIX and similar operating systems.

Records are added as we write to the file.

In general, pile files can have variable-length records, which leads to all kinds of complications.

For example, random access now becomes difficult, since we don’t know what the offset of the beginning of the record is. (5)

In order to figure out this offset, we need to know the length of all the preceding records. To do this we need sequentially access all these records.

An additional problem is ...

[Slide 8]

the addition, deletion, or modification of records.

Say that we want to update the third record with one that is longer. If the file is stored on a persistent storage device we need to find a way to replace the old record with the new one without having to shift all the subsequent records over to make the new fit.

One way this can be done is to use a separate log file or transaction file, where we store recent modifications.

The new record is temporarily appended to the log file, which is then folded back into the main file at an opportune moment.

Sequential File

Another organization is the sequential file, which is a sequence of records, which all have the same format.

Often, one of the record fields are used as the key field, and the records are stored in order of the record field.

This organization makes it very efficient to process records in batches, which happens a lot in data base applications.

In this form, it does not support random access using keys to locate records.

Also, how can we efficiently add new records, given that the records are stored in key order?

The solution is, again, to temporarily store the modifications in a log or transfer file and fold the changes into the main file.

Indexed Sequential File

In order to improve support for random access, index sequential files, add special index files, which contain optimized data structures that map keys to offsets of records.

A special overflow file allows for records to be added while maintaining the efficient support for random access provided by the index.

Records are added to the overflow file with a link from the predecessor in the main file.

Indexed File

Indices can be added to all kind of files, even piles.

This greatly increases random access performance.

We can also add multiple indices, for different keys.

Some of the indices can also be partial, meaning that a key does not point to a unique record, but to a whole class of records that match the key.

Such files are generally called indexed files.

Introduction to File Systems Conclusion

We have come to the end of this introductory lesson on File Systems.

We have discussed what a file is, and what some of the roles are of a file system.

We have learned to distinguish two methods of accessing records in a file.

We also have talked about a number of file organizations.

In all but the highest-performance systems, sophisticated file organizations have been largely replaced by simple sequences of bytes, on top of which system and application designers build again sequential organizations and support for indices in user space. The cost of doing this is not negligible.

We hope that you have enjoyed this lesson.

You now have the terminology necessary as we explore together file system implementations.

Thank you for watching.