File Systems

2024 Semester 2 COMPSCI 340: Operating Systems
Talia Xu

Lecture 1
1.0.0

I’m Talia, your instructor

I am a lecturer in the Computer Systems Group.

I am about to receive my PhD from TU Delft.

I previously went to University of Toronto for BASc and MASc.

I'm Talia, your instructor

I worked at a few industry places, mostly as an intern

- Canada: IBM, Arista Networks, Amazon
- US: Reach Power (startup)
- UK: Nokia Bell Labs

My research interest is

- Using light for communication and sensing
- Novel sensing systems for health / mental well-beings
- Generally involvesome hardware or embedded systems

I'm Talia, your instructor

Office: 303s-592

Office hours: Mondays & Fridays 1 pm - 2 pm

Email: talia.xu@auckland.ac.nz

Please provide feedback

Let me know what you like, dislike, or want to see more of

I’m open to suggestions!

These books complement lectures

"Operating Systems: Three Easy Pieces"
by Remzi Arpaci-Dusseau and Andrea Arpaci-Dusseau

"Modern Operating Systems"
by Andrew S. Tanenbaum

Test: You are only responsible for what have been covered in class.

Why should we care about OS?

Distinguished Lecture by Amin Vahdat

- **End of Moore's Law:** The traditional approach of relying on exponential hardware improvement is no longer sustainable.
- **Shifting Demands:** The growing demand for machine learning and data processing requires a different approach to computing infrastructure.
- **System-Level Optimization:** Optimizing the entire system, rather than individual components, can unlock greater gains in capacity and capability.

File System: How to manage a persistent device?

What are the APIs?

What are the important aspects of the implementation?

File system: How to manage a persistent device?

You have a **1 MB file** named foo stored in the directory /bar

- What is the **full path name** of the file?

How are files organized on the disk?

- What **data structures** are used to store the data of the file?
- The **metadata**?
- The **location**?

How do you access data on the disk?

- When you open/write/read the file, what **function calls** are evoked?
- If we write another 1MB data to /bar/foo, where does this data go?
- Which structures on the disks are read from and written to?

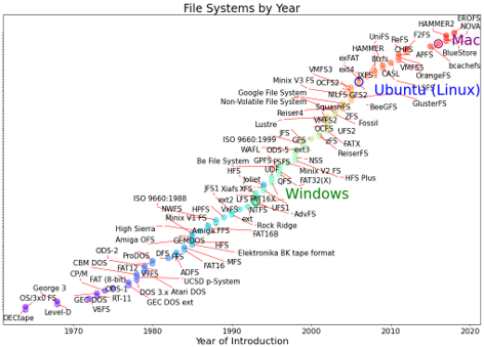
File system is an abstract concept

A File system describes a way of organizing and storing files on a storage device.

What are some requirements of a file system?

- **Availability:** Ensure data can be accessed and used reliably.
- **Permanence:** Store data permanently (or an approximation of it).
- **Concurrent Access:** Make data sharable with other programs or users

File system implementations are design choices



Some popular file system implementations:

- Unix (Linux and Mac)
- Windows
- Specialized system (Mainframe)

What is a file?

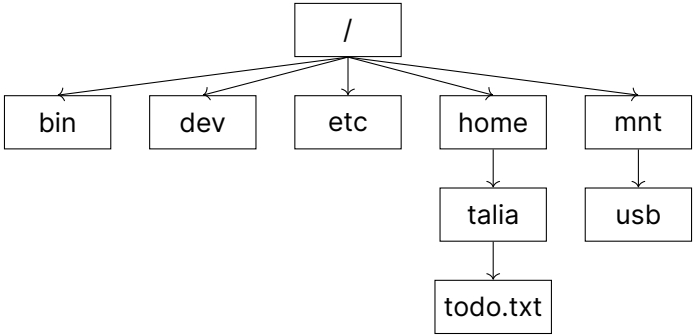
Textbook: A named collection of related information that is recorded on secondary storage.

For most users, the basic unit of interaction with a file system is a file.

- Is a directory a file?
- Is a process a file?
- Is a device a file?

Usual layout of a POSIX Filesystem

POSIX is a set of standards that provides a common interface for operating systems.



Working Directory: /home/talia

- What is the absolute and relative path to **todo.txt**? To **usb**?

POSIX Filesystem

todo.txt Relative: ./todo.txt
todo.txt Absolute: /home/jon/todo.txt
usb Relative: ../../mnt/usb
usb Absolute: /mnt/usb

Special symbols:
- . — Current directory
- .. — Parent directory
- ~ — User's home directory ($HOME)

Relative paths are calculated from current working directory ($PWD)

You Can Access Files Sequentially or Randomly

Sequential access

- Each read advances the position inside the file
- Writes are appended and the position set to the end afterwards

Random access

- Records can be read/written to the file in any order
- A specific position is required for each operation

POSIX Filesystem

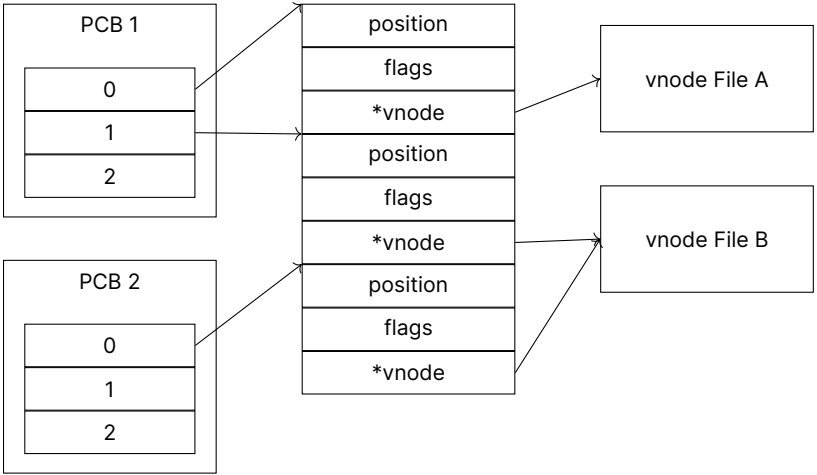
```
int open(const char *pathname, int flags, mode_t mode);

// flags can specify which operations: O_RDWR,O_WRONLY, O_RDWR
// also: O_APPEND moves the position to the end of the file initially

off_t lseek(int fd, off_t offset, int whence);

// lseek changes the position to the offset
// whence can be one of: SEEK_SET, SEEK_CUR, SEEK_END
//   set makes the offset absolute, cur and end are both relative
```

File Tables Are Stored in the Process Control Block (PCB)



Each Process Contains a File Table in its PCB

A **Fie Descriptor** is an **index** in the table

Each item points to a system-wide *global open file* ***(GOF)*** *table*

The GOF table holds information about the **seek position** and **flags**

- It also points to a *VNode* (supports read/write/etc)

A **vnode** (virtual mode) holds information about the file

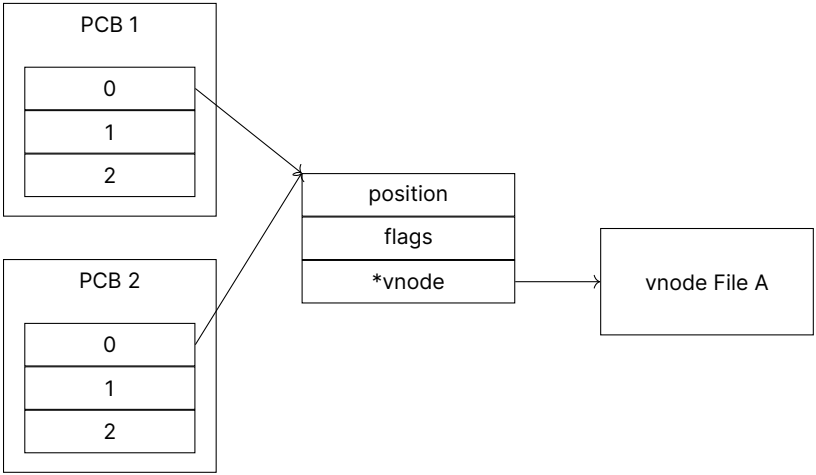
- vnodes can represent regular files, pipes, network sockets, etc.

What happens in a fork?

PCB is copied – local open file table gets inherited

Both PCBs point to the same Global Open Table entry

Both Processes Point to the Same GOF Entry

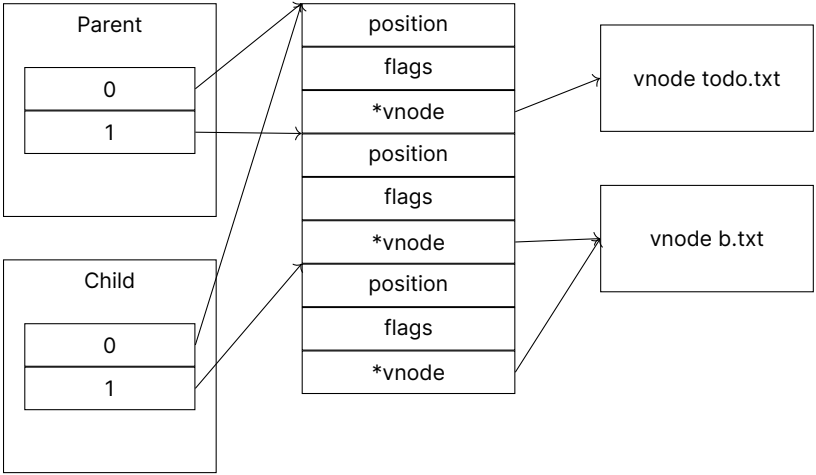


How many LOF and GOF Entries Exist? What is the Relationship?

```
open("todo.txt", O_RDONLY);
fork();
open("b.txt", O_RDONLY);
```

Assume there are no previously opened files (not even the standard ones)

There are 2 LOF Entries Each, and 3 GOF Entries



What happens in Fork?

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    FILE *file;
    file = fopen("test.txt", "w");

    // Fork the process
    pid_t pid = fork();

    fputs("hello\n", file);
    fputs("hello again\n", file);

    return 0;
}
```

What happens in Fork?

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    FILE *file;
    file = fopen("test.txt", "w");

    fputs("hello\n", file);
    fputs("hello again\n", file);

    // Fork the process
    pid_t pid = fork();

    return 0;
}
```

What happens in Fork?

```
int main() {
    FILE *file;
    file = fopen("test.txt", "r");

    pid_t pid = fork();

    if (pid == 0) {
        char line[256];
        if (fgets(line, sizeof(line), file) != NULL) {
            printf("Child read: %s", line);
        }
    } else {
        char line[256];
        if (fgets(line, sizeof(line), file) != NULL) {
            printf("Parent read: %s", line);
        }
    }
}
```

Reading

Textbook

- 13.1 File Concept
- 13.2 Access Methods