UE16CS305: Introduction to Operating System Laboratory

IOS Lab Project - Simple File System

Goals: The primary goal for this lab project is for you to have practical knowledge of building a functional file system for Linux. Doing so will familiarize you with the concepts of Linux file systems, file I/O, device interactions and system calls.

Also, this would help you to work in a team setting, understand the requirements and design a system to meet these requirements.

Problem Statement: To build a simple file system. The details are as follows:

The project can be decomposed into three phases in terms of development:

1. PHASE I: System call implementation
2. PHASE II: File System Abstractions
3. PHASE III: Secondary Storage

PHASE I: System call implementation

This requires you to implement system calls for file operations that can be issued by Linux. For the system call components, you can use a software facility call FUSE. FUSE (File System in User Space) https://en.wikipedia.org/wiki/Filesystem_in_Userspace is available for most Linux systems. It provides a way to intercept file system calls issued by Linux programs and to redirect the program flow into a handler (daemon running as a user-level process) function. It also helps to mount the file system. Now if a programmer (via any Linux program) makes a file system call, FUSE will invoke a routine in a daemon process (http://man7.org/linux/man-pages/man3/daemon.3.html) or a handler that you have written instead of sending that system call to the Linux file system implementation in the kernel. And in return it will show you the response by the handler function rather than the original file system.

So, FUSE helps you in writing a virtual file system. Thus, you can test your file system as if it were any other file system. More importantly, you can compare the results that your file system produces to the results produced by the Linux file system. This comparison between the results of your file system vs linux file system is how we will evaluate your work for phase 1 at the time of submission.

Task:

Implement the basic File I/O operations i.e. open/close/read/write/seek and directory functions such as mkdir, readdir etc. To perform this task, you are required to build an emulator for secondary storage that can be loaded in memory with your file system. This emulator need not persist beyond the lifetime of a mount i.e. it does not implement persistence.

- Implement mkfs (https://linux.die.net/man/8/mkfs) to make a file system in the emulator
- Implement the file abstractions (block management, block maps, directories, etc.) using the in-memory emulator
- Integrate the file system implementation with FUSE

At the end of phase 1, you should have a basic file system that works in memory only. When you unmount the file system all files are lost. Similarly, when you mount it, you must make a new file system before any operations start.

Student Group Formation and Project Assignment:

1. Students are required to work in groups of 3. Student groups are created within their own lab batch. For e.g. section A is divided into A1 and A2. If student belongs to A1, they are supposed to make a group of 3 within A1 batch.
2. Students are required to research on their own and come up with a design followed by an implementation.
3. **Students Hand In:** Students should submit the source code (including Makefile) and documentation. They are supposed to make a report (documentation) which documents the file system and its features, as they have implemented them. They must specify amongst other features, how they handled meta data and persistence in their design. There are separate marks for documentation and based on how well they document their design, marks will be allotted. Also, marks will be allotted based on individual contributions.
4. **Students Grading :** Students will work in a group of 3. All members of the team should be from the same lab group. They will demonstrate their file system to the lab faculty handling their lab as a group. They will have to make a short presentation along with the demonstration of your project describing the features of your implementation. Faculty will test their file system's functionality and speed with functionality being the most important. This last point bears repeating. A slow file system that works without failure is worth more

than a fast file system that fails. They are expected to demonstrate high ethical standards and not plagiarize for that will lead to loss of a grade. While discussion amongst teams is encouraged, plagiarism is not.

Block Diagram of File System: