ras70, ghb5
mlw55, nsl8

Project Proposal: Write-Once File System

Group 5: G. Bernard, N. Lockett, R. St. Pierre, M. Wu

Executive Summary

The goal of the project is to build a write-once file system. For the project to succeed, there should be two utilities: a mastering utility, which writes the file system to either a proprietary image file or to a standard like .iso; and a mounting utility, which reads from the file system image in real time. At this time, we believe this project will require few resources beyond development time.

Product Summary

Our project involves making a space efficient, write-once file system conversion tool that can do the following:

- ❖ Master an existing filesystem into either our *proprietary* or a standard (such as .iso) filesystem image
- ❖ Mount that file and be able to read from it like a true file-system (not extracted from the file at load time)

Resources Required

This project will not require any hardware components, and may not even require development time on the server. That said, we will likely need to learn FUSE, or another file system library. Due to this, we will all need a consistent Linux based development environment, even though we are language agnostic at this early stage. This will most likely mean using our server through ssh or using a VM on our personal systems.

Success Criteria

Our baseline success criteria are listed below in order of concern:

1. Functional correctness (i.e. it works). More formally, this means the file system is capable of mastering, mounting, and reading.
2. Completing milestones in a timely manner.
3. Space and time efficiency of the file system.
4. Extremely successful if we implement one to two ‘stretch goals’

Stretch Goals

Some features that would be valuable to learn and apply to our system would be:

- ❖ Error correcting codes so that the image could be left alone on a system for a longer time without losing data safety.

- Compression would improve our space efficiency for the file system (though this may affect read latency).
- A GUI front-end that incorporates our mastering and mounting program to have a seamless user ability to use our utilities.
- A **tree** command for quickly viewing the directory structure of our image file.

Major Risks/Challenges

- Development time is hard to know due to the unknowns at this point.
- Demanding space efficiency may also be a challenge at this early in the project plan.
- Binary (i.e. it works or it doesn't) success criteria means difficult to deliver partially finished work at the end of the semester.
- Mastering a standard filesystem is an important tradeoff to be decided early. If we were to use a standardized file system image format it would likely make our project easier to test, and in the event of unforeseen circumstances, submit a partially done project. However, using a standardized format may limit our freedom to add complex new features (such as ECC or compression).