SongFS

A FUSE-based MP3 filesystem

By Elaine Guo, Tim Geissler, & Alexis Cruz-Ayala

The Market Problem (Our Project Goal)

- Say you love music, and have downloaded a bunch of mp3 files to a folder on your computer
- But it’s not sorted so it’s hard to find the exact song(s) you’re looking for
- What do you do now?
- Enter SongFS:
  - The filesystem for anyone who uses a lot of mp3 files!
  - “Automated Music Cataloging”
  - “Streamline your Music Library”

Our Solution: SongFS

- A read-only FUSE filesystem which parses mp3 files and generates a file system to automatically organize songs based on their metadata.
- SongFS will organize files based on the artist -> album -> song hierarchy

Our Target Customers

- Individual music lovers looking to sort their personal digital library

- Seasoned businesses in the music industry looking for a scalable way to manage their catalogue
  - With a focus on streaming service providers

The Implementation and Development

- Building off of the FUSE assignment as a starting point - but using Python
  - Offers access to powerful libraries (MP3 parsing, complex data structures)
- Separation of FUSE/directory processes and parsing the mp3 files
- Pyinotify installed in the FUSE program to watch for and update created/deleted files

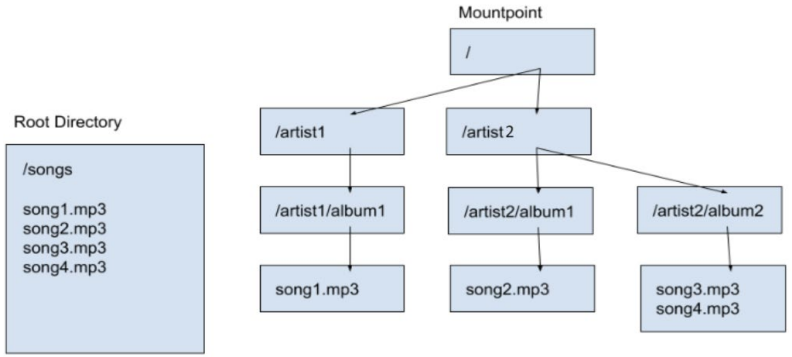
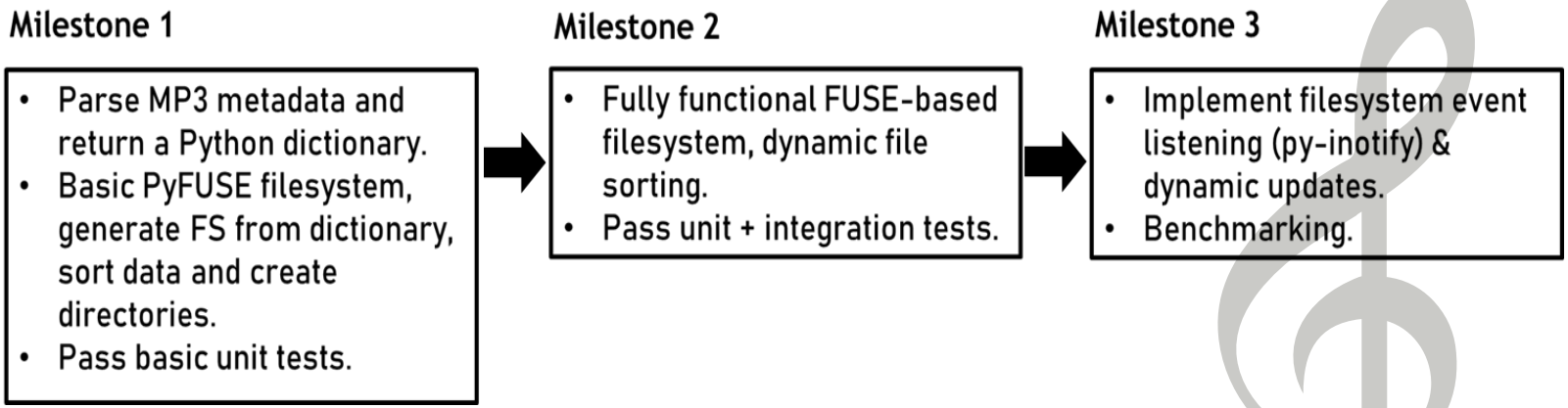


Figure 1: Diagram of how songfs would work: a root directory containing mp3 files can be mounted, splitting it into artist/album/song

Timeline



Updates to our Timeline

- For the most part, we have been successful in following our proposed timeline!
- One update we had to add to our schedule was to add benchmarking and performance testing

Challenges & Triumphs

- The main integration of the mp3 parsing and FUSE
- Accounting for edge cases in source files
  - Missing/incorrect metadata
  - Other file formats
- Pyinotify integration
  - Including synchronization challenges with dynamic filesystem updates

- Extensive unit tests ensured all parts were running smoothly
- Creating our own datasets that emulated/drew from real world music libraries
- Research with Pynotify was used to debug our software and prevent synchronization issues

SongFS Structure

- SongFS only requires a small subset of FUSE functions
- A PyFUSE superclass passes function calls to the kernel
- Specific methods are overridden to create SongFS
- This modular approach ensures flexibility and extendability

Project Evaluation

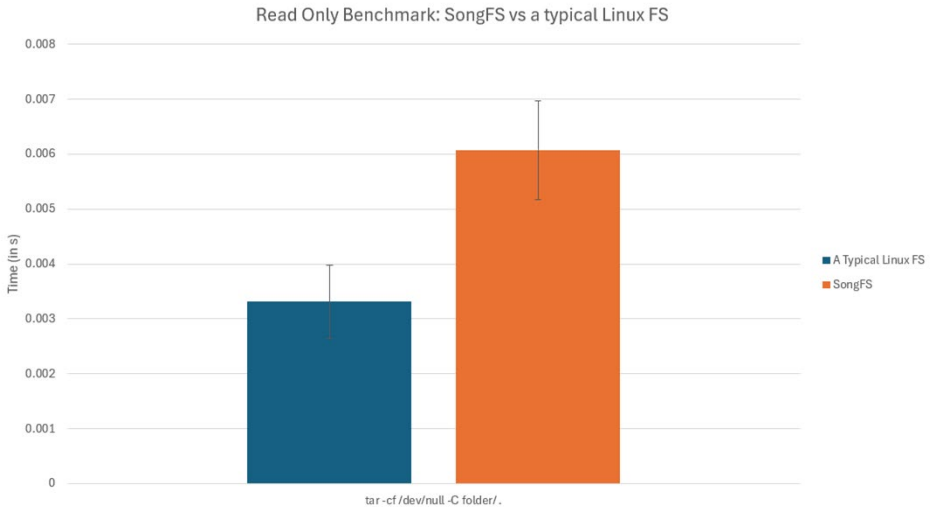
1. Organize songs based on metadata ✓
2. Read-only filesystem ✓
3. Music playback ✓
4. Dynamic updates to source directory ✓
5. Handle broken/missing metadata ✓
6. + Unit & Integration tests ✓

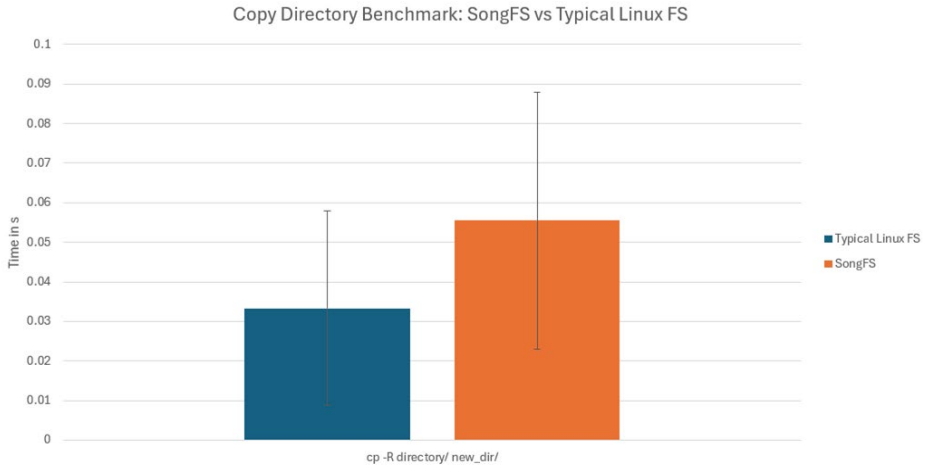
According to our Criteria For Success from our project proposal, we consider this project to meet our original criteria.

Benchmarking

We ran tests on our filesystem to compare it's reading speed to that of a typical Linux filesystem

We did this through timing commands that read our entire mount point or root directory





Demo!!