Custom Filesystem

Devi Sumanth
Lohith Reddy
Aasritha
Bhargav Ganesh
Gurrala Abhishek
Jaya Vishnavi
Moumitha
Phalgun
Pramodh Kumar
Sai Harshith
Sree Vasthav
Yeshaswini

TA : Pappu Kumar
Under the supervision of Prof. Janakiram

Problem Statement

Design and build a lightweight system for managing structured database files with fixed-size records by extending the functionality of existing system calls. This system will facilitate creating, accessing, reading, and writing records stored in a tabular format, indexed by row and column, and it will use a companion **index file** for each table file to store metadata about the table structure for fast direct record access.

Goal

Enable efficient management of table files by providing functions (or binaries) to:

- Create table files with specified rows, columns, and record size
- Open files to access specific records
- Read from a selected record
- Write to a specified record
- Delete table file

Design

Table Structure:

- Records are stored sequentially in row-major order (row-by-row) within the table file.
- The table is created with specific dimensions (rows, columns) and a record size, which will remain constant for the file's lifetime.

Index File:

- A separate index file will be created alongside each table file, storing essential metadata (such as number of rows and columns, record size).
- This index file will be placed in the same directory as its corresponding table file.

Functions to Implement

- **create_tablefile:**
  A function to create the main file and index file based on filename, nrows, ncols, and record_size. Also provided as an executable binary for terminal use, similar to touch.

- **open_tablefile:**
  Opens filename at a specific row_index and col_index, returning a file descriptor pointing to the start of the specified record.

- **delete_tablefile:**
  A function to delete the specified filename and its associated index file. Also available as a binary for terminal use, similar to rm.

Functions to Implement

- **read_tablefile:**
  Reads from a file descriptor into buffer up to size, with checks to ensure the read does not exceed the record boundary.

- **write_tablefile:**
  Writes from buffer to the file descriptor up to size, with boundary checks to prevent overwriting adjacent records.

Future Enhancements

Dynamic Record Resizing

- Enable records to expand or shrink as needed to support flexible data storage

String-Based Row and Column Access

- Allow access to records using string keys instead of numeric indices by storing row and column names in the index file.

Team Contributions

Team 1: Core File Operations

- Responsible for implementing create_tablefile, delete_tablefile, and managing the main and index file structures.
- Focuses on implementing open_tablefile, setting up accurate positioning within files for specified rows and columns.
- Ensures file creation, deletion, and index management are robust and reliable.

Bhargav, Lohith, Harshith, Aasritha

Team Contributions

Team 2: Record Access and Positioning, Testing

- Implements read_tablefile and write_tablefile with boundary checks to maintain data integrity within records.
- Conducts thorough testing and validation of the implementation, providing usage examples and highlighting future enhancement possibilities.

Sumanth, Abhishek, Pramodh, Vasthav

Team Contributions

Team 3: Documentation

- Document each function’s purpose, design, and usage, assembling a final report.

Aasritha, Abhishek (acts as bridge b/w implementation and documentation)

Moumitha, Yeshaswini, Phalgun, Jaya Vishnavi