Project Report - FMoT

Kandarp (110050005)
Rohan (110050011)

Aim of our project

- Build a working file manager on the terminal
- Support for common file manager tasks
  - Cutting, copying, pasting
  - Renaming
  - Deleting
  - Opening files
  - Searching for files (pattern-matching supported)

Interacting with FMoT

- To run it -> ./fmot
- Interface consists of the main window, command window and a few others
- Use the arrow keys to navigate
  - Up/down to move around in the working dir
  - Left to go to parent dir
  - Right to enter a directory / open a file
- For help with the commands -> press h
- When a command is run, the command window changes accordingly

Teamwork Details

- Kandarp
  - FMoT Interface (ncurses)
    - Menus (self-written), windows, panels
  - File Management (interface)
- Rohan
  - File Management (system code)
    - Renaming, moving, deleting, opening files
    - Searching (database + simple recursive)
- Overall contribution -> 50% each

Design Details

- Algorithms used
  - DFS (context: recursive search)
  - Serialization (context: database building)
- Data Structures used
  - General tree (context: filesystem)
  - Vector (context: files in working directory)
- Libraries / support
  - dirent (access to underlying filesystem)

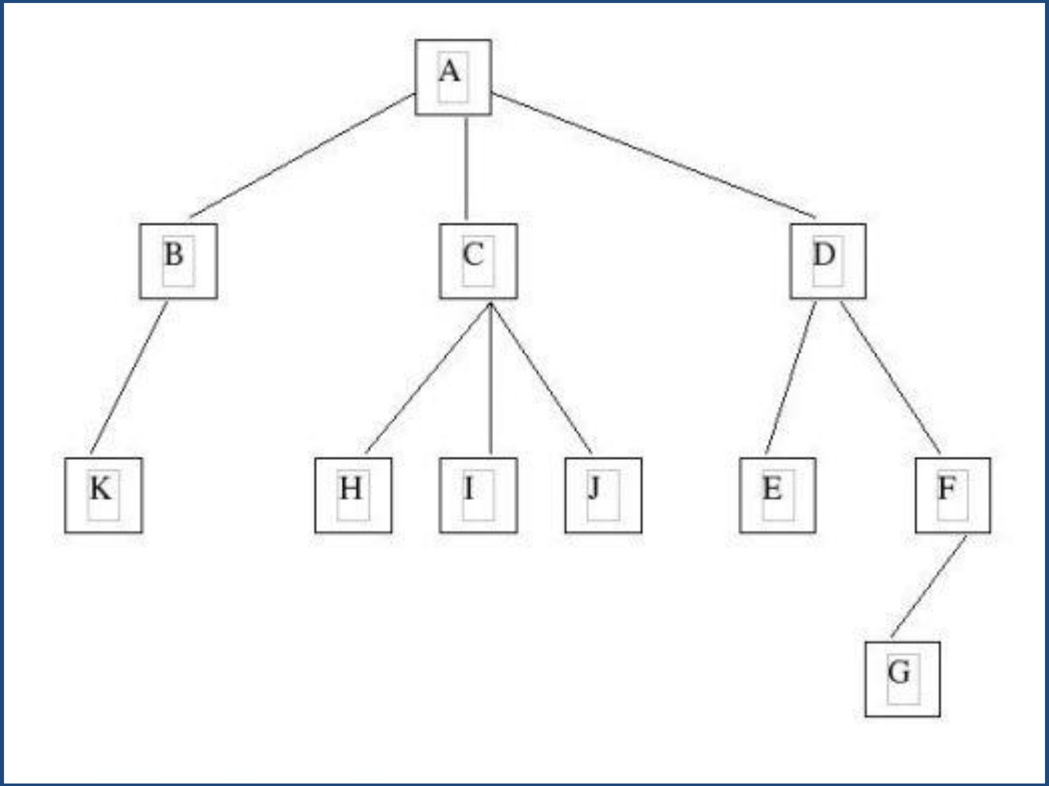
Design Details

- Libraries / support (contd.)
  - ncurses (low-level wrapper for controlling the terminal)
  - Lua 5.2.1 (pattern matching)

Algorithm Design

- Problem – make a database of the files on the filesystem
- Solution – Store it as a serialized general tree
- Explanation
  - Serialization is the conversion of a data structure in a store-able, recoverable format
    - In this context, we converted a tree to a general string
  - Minimizes space taken up by database

Serialization: tree -> string



- When serialized, this tree will become:

`A/B/K/\\C/H/\I/\J/\\D/E\F/G\\\`

Serialization explanation

- We are storing the tree nodes (in our case filenames) in a single string
  - Separated by / and \
  - A/B/ means that B is a child of A
  - \ means "go up one level"
  - A/B/\C/ means B is a child of A
    - But C is not a child of B! It is another child of A
  - A/B/\C/D/\\E means A has children B C and E, and D is C's child

Serialization (contd.)

- In conclusion:
  - Build the general tree for the filesystem (‘layered search’)
  - Serialize the tree
  - Store the serialized string in a file
  - Indexed searches will refer to this file (will be updated if necessary)

Algorithm Design (contd.)

- Problem – non-indexed recursive search for a filename matching a pattern
- Solution – Depth first search starting from the current directory
  - Each filename is matched against the pattern with a simple Lua script

Class Design

Class Name	Brief Description
customMenu	Contains functions for drawing windows and menus and working with menus (selecting objects, actions to be taken on mouse-click etc.)
database	Contains functions for building a database and recovering the tree from the serialized file and searching through it
tree	General tree class implemented as parents with a pointer to the leftmost child and two to their neighbouring siblings

Source code

Filename	Description	Author
fmot.cpp	Uses the custom menu to present the interface	K
generalTree.hpp	General tree class to represent the filesystem	K
menu&Interface.hpp	Custom menu class	K
fileManagement.hpp	File management functions for the interface	K
fileIndexing.hpp	Database class	K, R

Source code (contd.)

Filename	Description	Author
fileManagementFunctions .hpp	Functions for opening, mmoving, renaming and deleting files	R, K
matchWithLua.hpp	C++ code that calls a Lua script for pattern matching	R
patternSearchScript.Lua	Checks if its arguments pattern-match or not	R
macros	Macros to simplify the code	K

Conclusion

- Achieved a working file manager on the terminal
- Added support for all the common file management operations

Thank You – Questions?