Homework 5: Reading the xv6 Filesystem in Linux

Students: 207477753, 208076919

Implementation Overview

This application provides a utility to read and extract files from an `xv6` filesystem image (`fs.img`) within a Linux environment. The solution uses memory mapping (`mmap`) to access the filesystem image efficiently and parses the internal data structures defined in the `xv6` documentation (Chapter 6).

Key Components

1. Memory Mapping (`mmap`)
The program opens the `fs.img` file and maps it into the process address space using `mmap`. This allows us to access disk blocks using pointers, treating the image like a large array of bytes.

2. Core Structures
We implemented the following structures matching the `xv6` source code:
- `struct superblock`: contains filesystem metadata (inode start, data block start, etc.).
- `struct dinode`: the disk-based inode representation.
- `struct dirent`: directory entries containing file names and inode numbers.

3. Filesystem Traversal
- **`read_superblock()`**: Initializes the global pointer to the superblock located at block 1.
- **`get_inode(inum)`**: Calculates the memory offset for a specific inode based on the `inodestart` block and the size of a disk inode.
- **`ls()`**: Accesses the root inode (inode 1) and iterates through its data blocks. It safely prints each directory entry's name (handling the 14-character limit), type, inode number, and size.
- **`cp(src, dst)`**: Searches the root directory for the specified file. If found, it reads the data blocks associated with the file's inode and writes them to a new file in the host Linux system.

4. Handling Large Files (Indirect Blocks)
To support files larger than 12 blocks (6,144 bytes), we implemented support for the indirect block pointer (`addrs[NDIRECT]`). This involves:
1. Reading the indirect block containing 128 pointers to data blocks.
2. Iterating through these pointers to access the remaining data in the file.

Error Handling
The program includes checks for:
- Command-line argument validity.
- File system image accessibility.

- Existence of the requested file in the root directory (printing the required error message: `"File %s does not exist in the root directory"`).

Building the Project
Running `make` will build the executable named `hw5` using `gcc` with the `-Wall` and `-g` flags.