Operating Systems (COMPSCI 3SH3) Lab 5
File Block Allocation Simulator Header File

Prof. Neerja Mhaskar and Samkith Jain

Fall 2025

- You must show your working solution of this lab to the TA for a grade.
- Working directly on your local Linux machine, instead of a virtual machine (VM), may lead to kernel panic if mistakes are made. This could require you to reinstall your operating system and possibly lose data. Therefore, it is advisable to work on your VM for this lab.
- For Mac M1-4 chip users (all Macs with 64-bit ARM CPUs), you need to install UTM for virtualization: `https://mac.getutm.app/`
- The TA will check your solution and will quiz you on your work. After which they will enter your mark and feedback on Avenue.
- If you do not show your work to your Lab TA, you will get a zero (unless you provide an MSAF, in which case you will get 5 days extension to get this lab graded).
- It is your responsibility to connect with your Lab TA to get a grade and ensure that your grade has indeed been posted on Avenue.

Overview

In this lab you will create a header file named `fs_os.h` for your Assignment 5. Assignment 5 requires implementing the ***Indexed Allocation Scheme*** for a simulated flat file system in C. This scheme uses an index block to store pointers to data blocks, enabling efficient random access.

File System Specifications (in `fs_os.h`)

Simulate a flat file system with a single-level directory structure mounted on the `RootDirectory` folder, which supports creating files where data is stored in bytes, with the following specifications in `fs_os.h` header file:

- Block Size: 1 KB
- Total Blocks: 64
- Maximum Files: 10

These specifications should be implemented using `macros` in C within your header file.

A block may be simulated as such

```
typedef struct Block{
    unsigned char data[BLOCK_SIZE];
    int blockNumber;
} Block;
```

File Information Block (FIB)

Each file created in the file system must have a File Information Block (FIB) associated with it. The FIB must contain the following metadata related to the file:

1. file information block ID,
2. file name,
3. file size,
4. block count, and
5. a pointer to the index block (that contains pointers to the data blocks allocated for the file).

You must define a C structure to represent an FIB.

Supported Operations

The file system supports the following operations:

1. `initFS()` – initializes the file system.
2. `createFile(filename, size)` – creates a file in file system.
3. `deleteFile(filename)` – deletes a file with the given file name.
4. `listFiles()` – lists all the files in the flat file system.

In addition to the above functions, the file system supports the below three utility functions:

1. `allocateFreeBlock()` – removes a free block from the head of the free block list and returns it.
2. `returnFreeBlock(Block)` – adds a free block back to the tail of the free block list.
3. `printFreeBlocks()` – displays all free block numbers and the total count of free block.

File system structure

You should simulate the file system as a structure in your header file. This structure must contain the `Volume Control Block` structure that stores the following information about the volume:

1. free block list simulated as linked list of free blocks.
2. logical disk blocks available in the volume simulated as an array of Blocks
3. file control blocks ID and their status
4. list of files (represented as FIB) created
5. number of files created

Deliverables

Your group is required to show your `fs_os.h` header file to your Lab TA for a grade.

After getting a grade and implementing the feedback from the respective Lab TA to generate a correct header file (if not correct) use it to work on your Assignment 5.