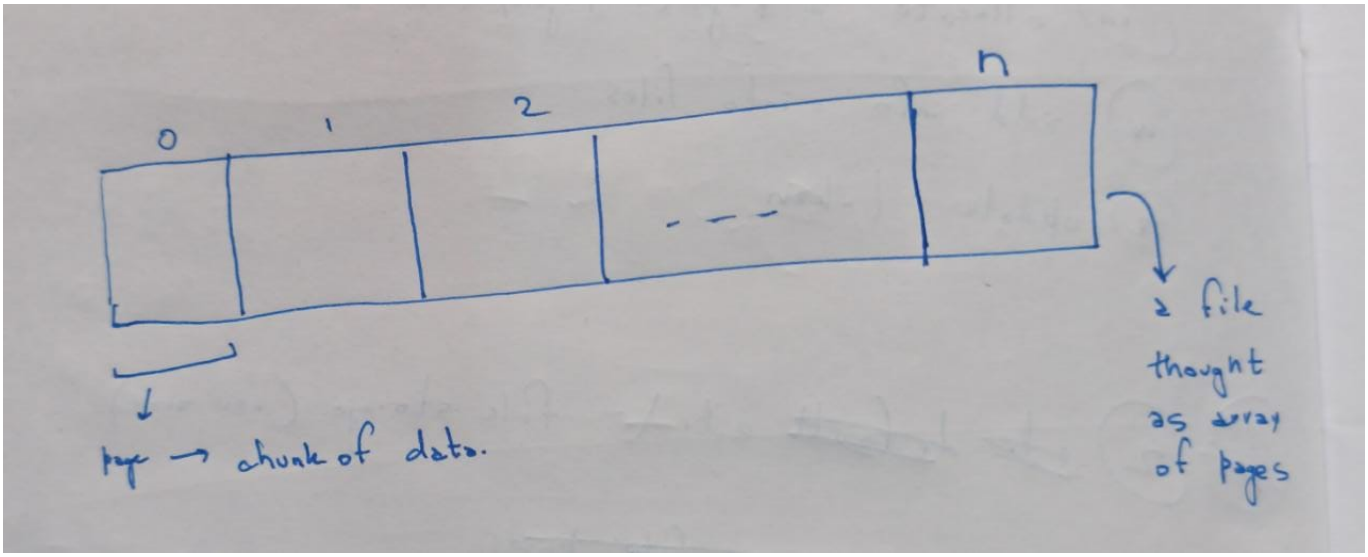
Operating System Assignment 12: Design a File System

Basic Idea (similar to the First File System)



- Imagine the file created via `int init_File_dd(const char *fname, int bsize, int bno)` as a file system.
- we divide the file system into predefined size pages to essentially convert the file system into a page array
- each file will have access to at least one page and can have access about multiple pages if the file size is more than that of the page size

Helper Data Structures

1. `unordered_map<int, bool> taken`
    - int will be index of the file system
    - `true` means the pages is occupied by some file and should not be given to any other file
    - `false` means that the page in not occupied by any file and can be given to any other new file.
    - default value of the map will be `false`
2. A `file` structure which has following parameters
    - `string name`: the name of the file
    - `vector<int> pages`: the pages occupied by the file
3. Root Directory Structure content
    - `vector<file> files` which defines the files in current directory

Functions that could be implemented

1. `int mount(char* mount_name, int file_system_size, int page_size)`
    - will call `int init_file_dd(const char *fname, int bsize, int bno)` to make a file_system are returns it's ddid
2. `int unmount(int ddid)`
    - executes and returns `int del_file_dd(const int ddid)`
3. `float statusFileSystem(int ddid)`
    - return the amount of space free in the file_system given by the ddid
4. `int addFile(char* filename, int size)`
    - calculate the number of pages required for making the file
    - finds a untaken pages via linear search
    - allocates a pages for the file by editing the `taken` vector
    - make a file struct for this file using the found untaken pages
    - return the idx position of the files_array or -1 if fail
5. `int editFileContent(int files_arr_idx, char* data)`
    - first check if the data to be inserted will need more pages or not and allocate accordingly
    - add data in the page_idx found via the file_arr
6. `vector<int> readData(int file_arr_idx)`
    - returns the sequential pages to read to get the file data
7. `int delete_file(int files_arr_idx)`
    - updates the taken_arr and "deallocates" the pages in the file found via file_arr_idx
    - also flushes the pages
8. `int editFileName(int file_arr_idx, string new_name)`
    - updates the filename of the file in the file_arr_idx

Future Additions

- Directory Structure recursive