# Project 1 Report

Name: Yao-Cheng Li
ID: 58609663

In project 1 we implement the paged file system, which allows the caller to perform page-level operations on files. The basic idea is to have a paged file manager (PF_Manager) to create, destroy, open and close files. Once we open a file, we get a file handler (PF_FileHandle) to modify the pages in a given file. We can append new pages, read from pages, or write data to pages. Here is how I implement the functions in the paged file system.

## (1) The PF_Manager class:

(a) *CreateFile*: First check if the file exists or not (By using fopen and check if the file pointer is NULL or not). If not, create a new file with the given file name; otherwise return -1.

(b) *DestroyFile*: First check if the file exists or not. If exists, delete the file with the given file name; otherwise return -1.

(c) *OpenFile*: First check if the file exists or not. Then check if the file handler is used by other file. In my implementation I do this by having a file pointer in the PF_FileHandle class. If the file pointer is set then it is being used by some file; otherwise it's free to use it. If the file exists and the file handler is not being used, then open the file and set the file pointer in file handler. Meanwhile get the size of the file to calculate the total number of pages in it.

(d) *CloseFile*: First check if the given file handler is handling a file or not (by checking its file pointer). If it's being used, flush file stream and close the file. Then reset the file handler so that it can be used by other file.

## (2) The PF_FileHandle class:

(a) *ReadPage*: First check if the given page number is valid or not (It is valid if it's more less than 0 and less than the current number of pages). If valid, move the file pointer (by the fseek function) to the corresponding page (PF_PAGE_SIZE * page number) and read the page to the given buffer.

(b) *WritePage*: First check if the given page number is valid or not. If valid, move the file pointer to the corresponding page and write the given data.

(c) *AppendPage*: First move the file pointer to the end of the file, then write the given data and increase the number of pages by one.

(d) *GetNumberOfPages*: In PF_FileHandle I have a attribute to remember the number of pages, and this function just returns that number.

(e) About resetting the file handler: Once PF_Manager's CloseFile function is called, a file handler would not be used by the file. To reset it for later use, I set the file pointer to be NULL and the number of pages to be zero.