# Department of Computing

**CS 330: Operating Systems**

**BESE-9AB**

**Lab 6**: **File Management**

**CLO4(Develop programs to interact with OS components through its API)**

**Date: 20 Nov 2020**

**Time: 10:00 AM – 01:00PM**

**&**

**02:00 PM – 05:00 PM**

**Instructor: Mr. Taufeeq Ur Rehman**

**Lab 6**:**File Management**

**Instructions**

* You are allowed to form groups of upto 2/3 students per group.
* The code should be developed within the team and any form of plagiarism will automatically result in zero for both the groups involved without any questions asked.
* Each student is responsible to understand the code being submitted under his or her name. Division of work is not explicitly required; however, each group member will be given average viva marks of the group. This implies that if a group member gets zero in viva and another gets full marks then both will score 50%.
* The submission deadline is one-week midnight Thursday 26 Nov 2020. Viva will be held during the lab hours.
* Any questions and comments on the lab must reach the faculty before Wednesday midnight through email or in person meeting. No questions will be answered after the said time..

**Introduction**

The purpose of this lab is for you to build the file management component of distributed file management system. You will build the structure for file management to provide access to user(s) to create, delete, update and query files in the system. You are required to build the system in python language and in the operating system of your choice. You are also free to make the design choices as discussed in book or a combination thereof as long as the requirements for the system are met.

**Objectives**

By the end of this lab you will be able to build a file structure and understand the operations on files and directories.

**Tools/Software Requirement**

* Programming

**Requirements**

* + - 1. Your program must allow the users to apply following actions on the files.

1. Create(fname)
2. Delete(fname)
3. Mkdir(dirName)
4. chDir(dirName)
5. Move(source\_fname, target\_fname)
6. Open(fName,mode)
7. Close(Fname)
8. fileObj.Write\_to\_file(text), fileObj.write\_to\_file(int write\_at ,text)
9. fileObj.Read\_from\_file(),fileObj.Read\_from\_file(start,size)
10. fileObj.Move\_within\_file(start,size,target)
11. fileObj.Truncate\_file(maxSize)
12. Show memory map
    * + 1. The files created should be maintained in some form of structure which allows some form of directories.
        2. The action to create should create a file entry in your file structure. This may or may not include creating space in your disk.
        3. Delete should remove the file form your file structure. This may or may not mean deleting the actual content of the file
        4. Move should change the association of file in the directory structure and must not require physical movement of the content.
        5. Open file should return a file Object and all read, write move and truncate should be through this object.
        6. Write to a file should be through a function and should have two modes.
13. Append mode writes to end of the file and write\_at should write at a specific point in the file.
14. The write\_at may overwrite data at the location specified.
    * + 1. Read from a file can have two modes
15. sequential access reads from first word and returns the entire content
16. readFrom(start,size) reads from the start memory location for size number of characters.
    * + 1. Move content within file allows user to move forward or back data. The input should be of the form. Move(Filename, from, to, size) where filename is the name of the file, from is the starting location of data to be moved, to is the location where the data should be placed and size is the size of the data to be moved.
        2. Truncatefile(size) should reduce the size of the file to size. Data within the file in memory location after size should be considered deleted.
        3. Show memory map should show the distribution of files in the memory.
        4. The system must maintain persistent data.
        5. **You are allowed only to create a single data file which should contain all the internal structure and data of the user created files.**
        6. The positive change of the size of the file should be automatic, that is if more data is written on a file, or data is moved within a file to new location then the size should automatically increase.
        7. You may use paging technique to manage your file system. The maximum file size and page count is left as your design choice.

**Tasks**

Design a program to implement the aforementioned functions and provide interface to execute the functions.

**Deliverables**

Submit

1. Complete code
2. Sample data file (sample.dat) consisting of files and directories to show your output.
3. User guide to teach how to use your system. The user guide must include description of your directory structure.