MP7 Report

Summary

In this machine problem, I implemented a simple file system and file handler and added that to our OS. The file system and file handler work and are able to successfully read and write from files as defined in *exercise_file_system* function. This report goes over some implementation details and the changes made from the code provided.

Changes

There were quite a few changes made to the starter code this time around.

- The biggest change was fixing the "*Undefined symbol XpmCreatePixmapFromData*" in the Bochs environment when starting the Bochs environment and running the code. After looking up this problem on the internet, the solution I found was to add "*display_library: sdl"* to bochsrc.bxrc file. But to use the sdl display library, I needed to install the library first by running the command *"sudo apt-get install bochs-sdl"* in the command line terminal. I found the fix at this link: https://titanwolf.org/Network/Articles/Article?AID=11475774-799e-44d4-a2df-b9b00dd4ec4b#gsc.tab=0.
- The static function *Format* defined in *FileSystem* class as changed to be a normal member function. This is because the implementation I used required *Format* function to access some non-static variables. As such, I edited *kernel.C* to call the *Format* function from the file system object rather than calling it directly from the class.
- In *File.H*, I changed the scope of the data members from private to public. This was done because I was accessing the data members in FileSystem class, and I was not familiar with friend classes enough to access private data members from another class.

Implementation

Two classes, *File* and *FileSystem* were used to implement the simple file system in this machine problem.

File

The class *File* implements six member functions, each of which are described below.

- The constructor for File is straightforward—it initializes the data members and does not do much. It points to the starting of the memory block, among other things.
- *Read* function reads n characters from the file starting at the current location and copies them to the buffer, returning the number of characters read. It also makes sure the end of file has not been reached.
- *Write* function writes n characters to the file starting at the current location. It also makes sure that if we ever run past the end of file, we increase the size of the file. This is achieved through a helper function called *GetNode*, which gets a new node for the file to write on.
- *Reset* simply resets the current position to the beginning of the file.

- *Rewrite* function erases the contents of the file. It also resets the current position of the file and file size.
- *Eof* just checks if the end o file has been reached, or if the file is empty.
- *GetNode* is a helper function I implemented to add data nodes to the file from the file system. This function invokes getNode of the FileSystem, which is another helper function.

FileSystem

- The constructor is simple, it just initializes the data members and clears the buffer by setting the buffer with zeros.
- *Mount* just assigns the disk data member to the one passed in as a parameter. This associates the file system with a disk. Since our file system is very simple, there is no error handling and the function returns true no matter what.
- *Format* wipes the file system from the given disk. It also installs an empty file system in its place.
- *LookupFile* will find the file with an id passed as a parameter. If it is able to successfully find the file with the matching file id, it returns a pointer to that file. Otherwise, it will return null to signify failure of operation.
- *CreateFile* just creates a file with the id passed in as a parameter. But it also makes sure that the file already does not exist, and if it does, it aborts the operation.
- *DeleteFile,* on the other hand, is responsible for deleting a given file with the provided file id passed in as a parameter. It also frees the disk space occupied by the file.
- *GetNode* is a helper function which creates a new node for file to write on and returns a reference to it.