PROJECT REPORT
by
Ankit Dave
(9211-5520)
for
Operating System
(COP 4600)

This project was made to create a C POSIX call to read file and return the file text in a string format. We began by implementing a header file that has the declaration of the read feel function, such that it took in a char array and retuned one as well. In the definition of this function, we use the POSIX library unistd to make a system call, which allows for the file to be read and return the read file in a null terminated character array. We also go though the process of finding out the characters in the file first so that we can dynamically allocate space in the program for the reading of the file.

The function once implemented can just be called form any code, and will return a string of the text file content. I also went ahead and created a shell script to make the call easier.

Now, to read any file, one only needs to call the shell script bash displayfile.sh <texfile name>.

This completed the text read part of the project. The second part of the project was creating an android app the uses these c functions to read the file. Since, Android uses java, we needed to create shared library to use the c function. We use the Cmake functionality of Android Studio to make a native-lib that has these c functions that can be called from java files. Using this shared library, we were able to call the c function from the java code and read the file sing system calls.

Bugs:

1. Size allocation. The need to allocate a size of 2 more than count was confusing. The problem was that the read was happening for count-1 chars as the while loop was not called for the last character. Other that that '\0' is a character in the end that needed space. So the size allocated was count+2.
2. Integrating the library. I tried integrating a library created using my terminal, directly in the code, skipping the C make portion of it. This should technically work, since that is what is on the website, but it just doesn't. In the end, the cpp code had to made and edited and directly called in the CMAKE to create a separate library by Android Studio.
3. The character pointer that was created, was first created locally in the read file function, which created a set fault when trying to read it. The char* had to be globally initialised to remove that problem
4. The permission issue was a problem that took a lot of time. The c code in android studio, if trying to access a file in a folder that does not give read permission to that app, simply could not read that file.
5. The function name in Android studio for read file could not just be readfile, it had to have the location of the function built into it for the Cmake to build the library.