**Assignment 6**

**Due, Sunday, November 21, 2021 for maximum 100%**

**Monday, November 22, 2021 for maximum 90%**

**Tuesday, November 23, 2021 for maximum 80%**

**Wednesday, November 24, 2021 for maximum 70%**

**Deliverables**

To complete this assignment you must submit your **OSManagement.c** to Webcourses.

**Project description**

This project will require students to simulate the behaviors of an operating system with a series of assignments.

1. Simulate process allocation to memory blocks based on memory management algorithms First Fit, Best Fit, Next Fit, and Worst Fit.
2. Simulate file management of directories and files stored in a directory.
3. Simulate multi-threaded programming with the POSIX (Portable Operating System Interface) threads (a.k.a. pthreads).

**C programming language integrated development environment (IDE)**

1. Code::Blocks ~ NOT Mac compatible
2. Visual Studio Code
3. Atom
4. <https://replit.com/>
5. XCode

**Assignment Scope: File Attributes**

1. Update source code file OSManagement.c to include preprocessor directives, function fileAtttributes, and function printAttributes.
2. Replicate the behavior of the file control block (FCB) on an operating system.

**References**

1. 61\_CGS3763 Operating System On Disk Data Structures.pptx
2. Directory entries
   1. <dirent.h>
      1. <https://pubs.opengroup.org/onlinepubs/7908799/xsh/dirent.h.html>
3. Structure of the data returned by the functions fstat(), lstat(), and stat()
   1. <sys/stat.h>
      1. <https://pubs.opengroup.org/onlinepubs/007904875/basedefs/sys/stat.h.html>
4. Permissions in attribute st\_mode
   1. <https://c-for-dummies.com/blog/?p=4101>

**Tasks**

|  |  |
| --- | --- |
| Activity | |
| OSManagement.c | 1. Make a copy of **OSManagement.c** then update the C source code |
| preprocessor | 1. Include the following C libraries    1. time.h    2. dirent.h    3. sys/stat.h 2. Function prototype for function **printAttributes()** 3. Function prototype for function **fileAttributes** **()** |
| main | 1. Update the main function so when the user enters menu selection value 2, function **fileAttributes** is called |
| fileAttributes | 1. Write function **fileAttributes** to do the following    1. Return type **void**    2. Empty parameter list    3. Declare a variable of data type **struct stat** to store the attribute structure (i.e. **statBuff**)    4. Declare a variable of data type **int** to store an error code (i.e. **err**)    5. Declare a variable of data type **struct dirent** as a **pointer** (i.e. **de**)    6. Declare a variable of data type **DIR** as a **pointer** set equal to function call **opendir()** passing explicit text **“.”** as an argument to indicate the current directory (i.e. **dr**)    7. If the **DIR** variable is equal to **NULL** do the following       1. Output to the console that "Could not open current directory"    8. While reading the directory contents is **NOT NULL** do the following       1. Set variable **err** equal to function call **stat()** passing as arguments          1. the **d\_name** of the directory entry          2. struct stat **statBuff**       2. Evaluate if variable **err** is equal to -1 (i.e. indicating the file could not be accessed)          1. Output to the console explicit text             1. “Error in stat”             2. “Press `Enter' to continue . . .”             3. Call function **getChar()**       3. Call function **printAttributes()** passing as arguments          1. **d\_name** of the directory entry          2. struct stat **statBuff** |
| printAttributes() | 1. Write function **printAttributes()** to do the following    1. Return type **void**    2. Parameter list includes       1. **char name[]**       2. **struct stat statBuff**    3. Declare a variable of data type **time\_t** (i.e. **t**)    4. Declare an array of data type **char,** size 100,to store the time in a string (i.e. **timeStr**)    5. Output the file name    6. Output the file device id    7. Output the file serial number    8. Output the file user id    9. Output the file group id    10. Output the file mode        1. Evaluate the file mode for owner permissions           1. Read           2. Write           3. Execute        2. Evaluate the file mode for group permissions           1. Read           2. Write           3. Execute        3. Evaluate the file mode for other permissions           1. Read           2. Write           3. Execute    11. Output the file creation date    12. Output the file last modification date    13. Output the file last accessed date    14. Output the file size |
| OSManagement executable |  |
| Test Case 1 | Test Case 1 passes |
| Test Case 2 | Test Case 2 passes |
|  | Source compiles with no errors |
|  | Source compiles with no warnings |
|  | Source runs with no errors |
|  | Source includes comments |

|  |  |  |
| --- | --- | --- |
| Test Case 1 | Run executable | The executable runs  The output in the command prompt should be similar to *Figure 1 displayMenu function output* |
| Test Case 2 | **User enters 2 to select File Managment** | Console output should be similar to *Figure 2 FCB Attributes per File in Directory* relative to where the program is running |

Text

Description automatically generated

Figure 1 displayMenu function output

Text

Description automatically generated

Figure 2 FCB Attributes per File in Directory