CP386: Assignment 1 – Fall 2023

Due on Sep 28, 2023 (Before 11:59 PM)

This is an **individual** assignment, and we will try to practice the concept of system calls and implement the Operating System's basic functionality.

General Instructions:

- For this assignment, you must use C99 language syntax. Your code must compile using make **without errors.** You will be provided with a Makefile and instructions on using it.
- **Test your program thoroughly with the GCC compiler in a Linux environment.**
- If your code does not compile, **then you will score zero**. Therefore, ensure you have removed all syntax errors from your code.
- **Gradescope** platform would be used to upload the assignments for grading. The link to the Gradescope assignment is available on Myls course page. Drag and drop your code file(s) for submission into Gradescope. **Make sure that your file name should be as suggested in the assignment; using a different name may score zero.**
- Please note that the submitted code will be checked for plagiarism. By submitting these .c files, you would confirm that you have not received unauthorized assistance in preparing the assignment. You also confirm that you are aware of course policies for submitted work.
- Marks will be deducted from any questions where these requirements are not met.
- Multiple attempts will be allowed, but your last submission will be graded before the deadline. Instructors reserve the right to take off points for not following directions.

Warning:

Follow the assignment instructions to the letter in terms of the file names and function names, as this assignment will be auto-graded. If anything is not as per the description, the auto-grading fails, and your assignment will be given a **Zero** mark.

Question 1

Write a C program (file_directory.c) to perform various operations on files, directories, and associated system functions. Files and directories are frequently used in daily life to organize files well-structured. This program would use various **system calls** that are available in C programming to perform the operations. Create the C functions (as function declarations given below) to perform the following file and directory operations:

1. *void create_directory(const char *dir_name, mode_t mode): Create a directory with a given name and permissions.*
2. *void create_write_file(const char *file_name, const char *content): Create a file with a given name and write "Operating System is Fun!!" to the file.*
3. *void get_working_dir(void): Get the working directory path.*
4. *void read_historylog_to_file(const char *log_file_name): Extract a terse list of what package actions occurred and save last 10 lines of that file to a (.log) file. Hint: you should use popen() to run the system command in a C program.*
5. *void read_proc_mem_info_to_file(const char *file_name): Extract and save key information about the system's memory from /proc/meminfo to a file.*
6. *void create_subdirectory_and_move_file(const char *sub_dir_name, mode_t mode, const char *file_name): Create a subdirectory of the given name and permissions and move a file (must exist) to the this subdirectory. Hint: you should use rename() function for moving the file.*
7. *void change_directory(const char *dir_name): Change the working directory to the given name.*
8. *void directory_listing(const char *start_dir): Recursively lists all files and subdirectories in a given directory and its subdirectories. Print this list to standard output.*
9. *void remove_file(const char *file_name): Removes a given file from the system.*
10. *void remove_directory(const char *dir_name): Removes a given directory and its subdirectories from the system. Note: Must prints a message: "Warning: You are about to delete the directory DIRECTORY_NAME and all its contents recursively. Proceed? (y/n)"*

The program should give the user options to select the operations and keep looping until the user enters 99 on the keyboard. You can use the makefile to compile and run the program. Make sure to report an appropriate error message when a system call fails, you are free to choose the text for error messaging.

The expected output for executing make runq1 or. /file_directory is:

```
----- Menu -----
1: Create a directory
2: Create and write to a file
3: Get Working Directory
4: Save /var/log/apt/history.log to a file
5: Save /proc/meminfo memory info to a file
6: Create a sub-directory and move a file to sub-directory
7: Change directory
8: List directory contents recursively
9: Remove a file
10: Remove a Directory
99: Quit
Enter your choice: 1
Enter the directory name you want to create: test
Enter the mode of the created directory: 755
The directory test is created successfully with mode 755

----- Menu -----
1: Create a directory
2: Create and write to a file
3: Get Working Directory
4: Save /var/log/apt/history.log to a file
5: Save /proc/meminfo memory info to a file
6: Create a sub-directory and move a file to sub-directory
7: Change directory
8: List directory contents recursively
9: Remove a file
10: Remove a Directory
99: Quit
Enter your choice: 2
Enter the file name you want to change to: testing.txt
File testing.txt is created and written successfully.

----- Menu -----
1: Create a directory
2: Create and write to a file
3: Get Working Directory
4: Save /var/log/apt/history.log to a file
5: Save /proc/meminfo memory info to a file
6: Create a sub-directory and move a file to sub-directory
7: Change directory
8: List directory contents recursively
9: Remove a file
10: Remove a Directory
99: Quit
Enter your choice: 3
The current working directory is: /workspaces/F23/Assignments/A01_F23

----- Menu -----
1: Create a directory
2: Create and write to a file
3: Get Working Directory
4: Save /var/log/apt/history.log to a file
5: Save /proc/meminfo memory info to a file
6: Create a sub-directory and move a file to sub-directory
7: Change directory
8: List directory contents recursively
9: Remove a file
10: Remove a Directory
99: Quit
Enter your choice: 4
Enter the file name you want to store history logs into: history.log
The last 10 lines of history/log are copied to history.log successfully.

----- Menu -----
1: Create a directory
2: Create and write to a file
3: Get Working Directory
4: Save /var/log/apt/history.log to a file
5: Save /proc/meminfo memory info to a file
6: Create a sub-directory and move a file to sub-directory
7: Change directory
8: List directory contents recursively
9: Remove a file
10: Remove a Directory
99: Quit
Enter your choice: 5
Enter the file name you want to store memory information to: meminfo.log
Proc info is written to meminfo.log successfully.

----- Menu -----
1: Create a directory
2: Create and write to a file
3: Get Working Directory
4: Save /var/log/apt/history.log to a file
5: Save /proc/meminfo memory info to a file
6: Create a sub-directory and move a file to sub-directory
7: Change directory
8: List directory contents recursively
9: Remove a file
10: Remove a Directory
99: Quit
Enter your choice: 6
Enter the sub-directory name you want to create: subdir
Enter the mode of the created directory: 755
Enter the file name you want to move to the sub-directory: testing.txt
Directory subdir created successfully with mode 755
The file testing.txt has been moved successfully to the sub-directory.

----- Menu -----
1: Create a directory
2: Create and write to a file
3: Get Working Directory
4: Save /var/log/apt/history.log to a file
5: Save /proc/meminfo memory info to a file
6: Create a sub-directory and move a file to sub-directory
7: Change directory
```

```
8: List directory contents recursively
9: Remove a file
10: Remove a Directory
99: Quit
Enter your choice: 7
Enter the directory name you want to change to: subdir
The working directory has been successfully changed to subdir

----- Menu -----
1: Create a directory
2: Create and write to a file
3: Get Working Directory
4: Save /var/log/apt/history.log to a file
5: Save /proc/meminfo memory info to a file
6: Create a sub-directory and move a file to sub-directory
7: Change directory
8: List directory contents recursively
9: Remove a file
10: Remove a Directory
99: Quit
Enter your choice: 8
Enter the directory name you want to list the contents for (if you want to list of the contents of current directory, then pass `.`): .
testing.txt
.
..

----- Menu -----
1: Create a directory
2: Create and write to a file
3: Get Working Directory
4: Save /var/log/apt/history.log to a file
5: Save /proc/meminfo memory info to a file
6: Create a sub-directory and move a file to sub-directory
7: Change directory
8: List directory contents recursively
9: Remove a file
10: Remove a Directory
99: Quit
Enter your choice: 9
Enter the file name you want to remove: testing.txt
The testing.txt file has been successfully removed:

----- Menu -----
1: Create a directory
2: Create and write to a file
3: Get Working Directory
4: Save /var/log/apt/history.log to a file
5: Save /proc/meminfo memory info to a file
6: Create a sub-directory and move a file to sub-directory
7: Change directory
8: List directory contents recursively
9: Remove a file
10: Remove a Directory
99: Quit
Enter your choice: 7
Enter the directory name you want to change to:
The working directory has been successfully changed to ../

----- Menu -----
1: Create a directory
2: Create and write to a file
3: Get Working Directory
4: Save /var/log/apt/history.log to a file
5: Save /proc/meminfo memory info to a file
6: Create a sub-directory and move a file to sub-directory
7: Change directory
8: List directory contents recursively
9: Remove a file
10: Remove a Directory
99: Quit
Enter your choice: 10
Enter the directory name you want to remove: subdir
Warning: You are about to delete the directory subdir and all its contents recursively. Proceed? (y/n): y

----- Menu -----
1: Create a directory
2: Create and write to a file
3: Get Working Directory
4: Save /var/log/apt/history.log to a file
5: Save /proc/meminfo memory info to a file
6: Create a sub-directory and move a file to sub-directory
7: Change directory
8: List directory contents recursively
9: Remove a file
10: Remove a Directory
99: Quit
Enter your choice: 99
```

Important Note: When submitting a source code file to Gradescope, make sure to name it like:

- `file_directory.c`

Question 2

Create a C program (name it "filecopy.c") that copies the contents of one file to a destination file. This program will read data from one file and copy them to another. The first input that the program will need is the names of the two files: input file (“input.txt”) and output file (“output.txt”). Once the two file names have been obtained, the program must open the input file and create and open the output file. Each of these operations requires another system call. Possible error conditions for each system call must be handled. When the program tries to open the input file, it may find that there is no file of that name or that the file is protected against access. In these cases, the program should output an error message (another sequence of system calls) and then terminate abnormally (another system call).

If the input file exists, then we must create a new output file with the file name supplied above. We may find that there is already an output file with the same name. This situation may cause the program to delete the existing file (another system call) and create a new one (yet another system call).

When both files are set up, we enter a loop that reads from the input file (a system call) and writes to the output file (another system call). Each read and write must return status information regarding various possible error conditions. On input, the program may find that the end of the file has been reached. There are **optional** tests that your program can test, such as a hardware failure in the read (such as a parity error), or the write operation may encounter various errors, depending on the output device (for example, no more available disk space).

Once the program is written, use the makefile provided to run the command that copies the input file ("input.txt") to the output file ("output.txt"). If you have correctly designed and tested the program, use the "strace" utility provided by Linux systems.

The expected output for executing:

1. **`./filecopy`**: If no argument (input and output files names) is not supplied ./filecopy, then it should print on console:

   Insufficient parameters passed.

2. **`./filecopy input.txt output.txt`**: this should print on console:

   The contents of file input.txt have been successfully copied into the output.txt file.

3. **`strace -c ./filecopy input.txt output.txt`**: This should print the count of the system calls made by the program:

```
The contents of file 'input.txt' has been successfully copied into 'output.txt' file
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
  0.00    0.000000           0         2           read
  0.00    0.000000           0         2           write
  0.00    0.000000           0         4           open
  0.00    0.000000           0         4           close
  0.00    0.000000           0         3           fstat
  0.00    0.000000           0         7           mmap
  0.00    0.000000           0         4           mprotect
  0.00    0.000000           0         1           munmap
  0.00    0.000000           0         3           brk
  0.00    0.000000           0         4         3 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         1           arch_prctl
------ ----------- ----------- --------- --------- ----------------
100.00    0.000000                    36         3 total
```

   Note: Your count of system calls may or may not change based on the implementation logic.

Important Note: When submitting a source code file to Gradescope, make sure to name it like:

- `filecopy.c`