CP386: Assignment 1 – Winter 2023

Due on January 26, 2023 (Before 11:59 PM)

This is an **individual** assignment, and we will try to practice the concept of system calls and implement 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 (version 5.4.0) 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. For submission, Drag and drop your code file(s) 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 this zip file, 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. We 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 (directory.c) to perform various operations on directories and associated system functions. Directories are frequently used in daily life to organize files in a well-structured manner. This program would use various system calls that are available in C programming to perform the operations. The program should perform the following directory operations:

- Creating a directory
- Removing a directory
- Getting the current working directory
- Changing a directory (to one level up from the current directory)
- Reading a directory
- Closing a directory, the current directory

The program should present the user with a list of options to select the directory operation and keep looping until the user enters "q" on the keyboard. You can use the makefile to compile and run the program.

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

```
● vscode → .../W23/CP386/Assignments/A01_W23 $ ./directory
Select the option(s) appropriately by entering the number:
 Enter 1 for creating a directory
 Enter 2 for removing directory
 Enter 3 for printing working directory
 Enter 4 for changing directory one level up
 Enter 5 for reading the contents of directory
 Enter 6 for closing the current directory
 Enter q to exit the program
1
Enter the Directory name you want to create:
test
Directory is Created Successfully.
Select the option(s) appropriately by entering the number:
 Enter 1 for creating a directory
 Enter 2 for removing directory
 Enter 3 for printing working directory
 Enter 4 for changing directory one level up
 Enter 5 for reading the contents of directory
 Enter 6 for closing the current directory
 Enter q to exit the program
2
Enter the Directory name you want to remove:
test
Directory is removed Successfully.
Select the option(s) appropriately by entering the number:
 Enter 1 for creating a directory
 Enter 2 for removing directory
 Enter 3 for printing working directory
 Enter 4 for changing directory one level up
 Enter 5 for reading the contents of directory
 Enter 6 for closing the current directory
 Enter q to exit the program
3
Current Working Directory is: /workspaces/W23/CP386/Assignments/A01_W23
Select the option(s) appropriately by entering the number:
 Enter 1 for creating a directory
 Enter 2 for removing directory
 Enter 3 for printing working directory
 Enter 4 for changing directory one level up
 Enter 5 for reading the contents of directory
 Enter 6 for closing the current directory
 Enter q to exit the program


4
Working Directory Before Operation:/workspaces/W23/CP386/Assignments/A01_W2
Directory Changed Successfully.
Working Directory After Operation: /workspaces/W23/CP386/Assignments
Select the option(s) appropriately by entering the number:
 Enter 1 for creating a directory
 Enter 2 for removing directory
 Enter 3 for printing working directory
 Enter 4 for changing directory one level up
 Enter 5 for reading the contents of directory
 Enter 6 for closing the current directory
 Enter q to exit the program
5
.DS_Store
.metadata
.vscode
A01_W23
A02_F22
A03_F22
A04_F22
A05_F22
Assignment 03.xlsx
Assignments_Questions
BookCode
C_projects
Producer-Consumer
RemoteSystemsTempFiles
autograder_samples-master
banker_yousuff.c
final-src-osc10e
test
test12
test_project_make
.
..
Select the option(s) appropriately by entering the number:
 Enter 1 for creating a directory
 Enter 2 for removing directory
 Enter 3 for printing working directory
 Enter 4 for changing directory one level up
 Enter 5 for reading the contents of directory
 Enter 6 for closing the current directory
 Enter q to exit the program


6
Directory Closed Successfully.
Select the option(s) appropriately by entering the number:
 Enter 1 for creating a directory
 Enter 2 for removing directory
 Enter 3 for printing working directory
 Enter 4 for changing directory one level up
 Enter 5 for reading the contents of directory
 Enter 6 for closing the current directory
 Enter q to exit the program
q
vscode → .../W23/CP386/Assignments/A01_W23 $
```

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

- `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, and macOS systems use the dtruss command to print the system calls made by the program. (The dtruss command, a front end to dtrace, requires admin privileges, so it must be run using sudo.)

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`