| **Assignment Title** | System Programming Project | **Group** | 6 |
|---|---|---|---|
| **Subject Code** | TMN4133 | **Subject Name:** System Programming | |

| **Name** | **Matric No** | **E-mail** | **Signature** |
|---|---|---|---|
| CHRISTOPHER SII HOW CHIONG | 69385 | 69385@siswa.unimas.my | Chris |
| HO WAN YU | 69860 | 69860@siswa.unimas.my | Wan Yu |
| HENG ZHEN YE | 69847 | 69847@siswa.unimas.my | Zhen Ye |
| SONG WANG YE | 71680 | 71680@siswa.unimas.my | Swye |

| **Name of Lecturer: Professor Madya Dr. Johari bin Abdullah** | |
|---|---|
| **Due Date**: 6th January 2023 | Date received and approved (for office use only) |

***Plagiarism and Collusion are methods of cheating that falls under PeraturanAkademikUniversiti Malaysia Sarawak para 11: EtikaAkademik***

**Plagiarism**

Plagiarism is the presentation of work which has been copied in whole or in part from another person's work, or from any other source such as the internet, published books or periodicals without due acknowledgement given in the text.

**Collusion**

Collusion is the presentation of work that is the result in whole or in part of unauthorized collaboration with another person or persons.

Where there are reasonable grounds for believing that cheating has occurred, the only action that may be taken when plagiarism or collusion is detected is for the staff member not to mark the item of work and to report or refer the matter to the Dean. This may result in work being disallowed and given a fail grade or if the circumstances warrant, the matter may be referred to a Committee of inquiry for investigation. Such investigation may result in the matter being referred to the University Discipline Committee, which has the power to exclude a student.

**Note:** Assignments are returned after being assessed by lecturer and to be collected from the lecturer's office. Students should endeavour to collect their assignments prior to commencement of next semester. All unclaimed assignments become the property of the faculty and will become subject to destruction.

**Student's statement:**

I certify that I have not plagiarized the work of others or participated in unauthorized collusion when preparing this assignment. I also certify that I have taken proper case in safeguarding my work and have made all reasonable efforts to ensure that may work not be able to be copied.

| **MARK :** | Comments: |
|---|---|

## Table of Contents

**Task Contribution**

| Name | Task | Percentage (%) |
|---|---|---|
| Christopher Sii How Chiong | supercommand.c | 100 |
| Ho Wan Yu | FileOperation.c | 100 |
| Heng Zhen Ye | DirectoryOperation.c | 100 |
| Song Wang Ye | KeyLoggerOperation.c | 100 |

## 1.0 How to Compile and Execute the Program

SYNOPSIS

supercommand [OPTION] [SUBOPTION] [FILE] [PATH] [PERMISSIONS]

```
EXAMPLES
          Create a file called "test.txt" in the "/home/user/documents" directory:
                 supercommand 1 1 test.txt /home/user/documents

          Change the permission of the file "test.txt" in the "/home/user/documents" directory to 755:
                 supercommand 1 2 test.txt /home/user/documents 755

          Read and print the contents of the file "test.txt" in the "/home/user/documents" directory:
                 supercommand 1 3 test.txt /home/user/documents

          Remove the file "test.txt" in the "/home/user/documents" directory:
                 supercommand 1 4 test.txt /home/user/documents

          Create a directory called "test" in the "/home/user/documents" directory:
                 supercommand 2 1 test /home/user/documents

          Delete the directory "test" in the "/home/user/documents" directory:
                 supercommand 2 2 test /home/user/documents

          Get the current working directory:
                 supercommand 2 3

          List the contents of the current directory:
                 supercommand 2 4

          To start the keylogger:
                 supercommand 3
```

First, you need to compile the file by typing “gcc supercommand.c -o supercommand”.

This is the example of how to execute the program which can be read from the man page of the supercommand.

To access the man page, you can type man supercommand. The tasks can be executed by referring to the examples section of the man page of the supercommand. Below are the examples of the command:

Create a file called "test.txt" in the "/home/user/documents" directory:

supercommand 1 1 test.txt /home/user/documents

Change the permission of the file "test.txt" in the "/home/user/documents" directory to 755:

supercommand 1 2 test.txt /home/user/documents 755

Read and print the contents of the file "test.txt" in the "/home/user/documents" directory:

supercommand 1 3 test.txt /home/user/documents

Remove the file "test.txt" in the "/home/user/documents" directory:

supercommand 1 4 test.txt /home/user/documents

Create a directory called "test" in the "/home/user/documents" directory:

supercommand 2 1 test /home/user/documents

Delete the directory "test" in the "/home/user/documents" directory:

supercommand 2 2 test /home/user/documents

Get the current working directory:

supercommand 2 3

List the contents of the current directory:

supercommand 2 4

To start the keylogger:

supercommand 3

## 2.0 Test Case

### 2.1 Test Case 1: Execute File Operation successfully

#### 2.1.1 Test Case 1.1: Create or open a file

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 1 1 file1.txt /home/swye/Desktop/sp_project
File has been created or opened successfully
```

file1.txt has been created or opened successfully in Test Case 1.1 by using "./supercommand 1 1 file1.txt /home/swye/Desktop/sp_project".

#### 2.1.2 Test Case 1.2: Change the permission of the file

```
-rwxrwxr-x 1 swye swye     0 Jan  6 05:53  file1.txt
```

Before changing the permission of the file, the permission of the file1.txt is -rwxrwxr-x.

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 1 2 file1.txt /home/swye/Desktop/sp_project 0664
File permission has been changed successfully
swye@SP-Project:~/Desktop/sp_project$ ls -l file1.txt
-rw-rw-r-- 1 swye swye 0 Jan  6 05:53 file1.txt
```

The command "./supercommand 1 2 file1.txt /home/swye/Desktop/sp_project 0664" is tested and the permission of the file1.txt has been changed successfully to 664.

#### 2.1.3 Test Case 1.3: Read and print the contents of the file

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 1 3 file1.txt /home/swye/Desktop/sp_project
File has been opened successfully
today is a good day
```

The command "./supercommand 1 3 file1.txt /home/swye/Desktop/sp_project" is tested and the file can be opened, readed and printed the contents.

#### 2.1.4 Test Case 1.4: Remove the file

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 1 4 file1.txt /home/swye/Desktop/sp_project
File has been deleted successfully
```

The command "./supercommand 1 4 file1.txt /home/swye/Desktop/sp_project" is tested and the file1.txt can be deleted successfully.

### 2.2 Test Case 2: Execute Directory Operation successfully

#### 2.2.1 Test Case 2.1: Create a directory

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 2 1 sp /home/swye/Desktop/sp_project
Successfully created directory /home/swye/Desktop/sp_project/sp
```

The command "./supercommand 2 1 sp /home/swye/Desktop/sp_project" is tested and the sp directory is successfully created.

#### 2.2.2 Test Case 2.2: Delete the directory

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 2 2 sp /home/swye/Desktop/sp_project
Successfully deleted directory /home/swye/Desktop/sp_project/sp
```

The command "./supercommand 2 2 sp /home/swye/Desktop/sp_project" is tested and the sp directory can be deleted successfully.

#### 2.2.3 Test Case 2.3: Get the current working directory

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 2 3
Current working directory: /home/swye/Desktop/sp_project
```

The command "./supercommand 2 3t" is tested and the current working directory can be shown.

#### 2.2.4 Test Case 2.4: List the current directory

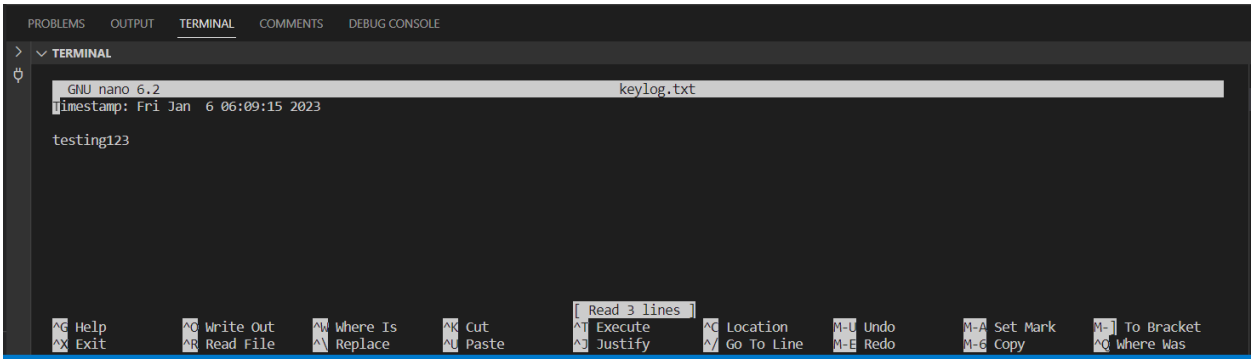
```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 2 4
List of Item in Current Directory:
FileOperation.c
Assignment1_71680.c
suppercommand manpage 1.txt
suppercommand
suppercommand manpage 2.txt
DirectoryOperations.c
FileOperation.o
supercommand.c
sp_project_old
Read Me =).txt
```

The command "./supercommand 2 4" is tested and the list of items in the current directory will be listed.

### 2.3 Test Case 3: Execute Keylogger Operation

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 3
keylog.txt has been opened successfully
testing 123
^C
```

The command "./supercommand 3" is tested and the keylog.txt can be opened successfully and can capture the keystroke after the user press Enter. By pressing the Ctrl + C will stop capturing and will be saved in the keylog.txt.




The keystroke that has been captured, will be recorded in the keylog.txt which can use the nano keylog.txt to verify.

### 2.4 Test Case 4: Handling error while executing the process

#### 2.4.1 Test Case 4.1: Invalid option

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 4 1 file1.txt /home/swye/Desktop/sp_project
Invalid option. Please refer man and enter your option again.
```

Error message "Invalid Option. Please refer man and enter your option again" will prompt out if the user types option other than 1, 2 and 3 for the second argument.

#### 2.4.2 Test Case 4.2: Invalid sub-option

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 1 5 file1.txt /home/swye/Desktop/sp_project
Invalid sub-option. Please refer man and enter your option again.
```

Error message "Invalid sub-option. Please refer man and enter your option again" will prompt out if the user types sub-options other than 1, 2, 3 and 4 for the third argument if the user choose 1 or 2 as the second argument (option).

### 2.4.3 Test Case 4.3: Error opening file for reading

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 1 3 file2.txt /home/swye/Desktop/sp_project
Error opening file for reading
: No such file or directory
```

Error opening file for reading when executing the command "./supercommand 1 3 file2.txt /home/swye/Desktop/sp_project" due to the file2.txt does not exist in the directory.

### 2.4.4 Test Case 4.4: Error deleting file

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 1 4 file2.txt /home/swye/Desktop/sp_project
Error deleting file
: No such file or directory
```

Error deleting file when executing the command "./supercommand 1 4 file2.txt /home/swye/Desktop/sp_project" due to the file2.txt does not exist in the directory.

### 2.4.5 Test Case 4.5: Error creating directory

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 2 1 sp /home/swye/Deskt
Error creating directory
```

Error creating directory when executing the command "./supercommand 2 1 sp /home/swye/Deskt" due to wrong directory path.

### 2.4.6 Test Case 4.6: Error deleting directory

```
swye@SP-Project:~/Desktop/sp_project$ ./supercommand 2 2 sp1 /home/swye/Desktop/sp_project
Error deleting directory
```

Error deleting directory when executing the command "./supercommand 2 2 sp1 /home/swye/Desktop/sp_project" due to the wrong name of the non-existing directory.

### 3.0 Complete Source Code

### 3.1 supercommand.c

```
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <dirent.h>
#include <stdlib.h>
#include <fcntl.h>
#include <stdio.h>
#include <time.h>
#include <string.h>
#include "FileOperation.c"
#include "DirectoryOperations.c"
#include "KeyLoggerOperation.c"

int main(int argc, char *argv[]){
    char *function_name=argv[1];
    char *function_name2= argv[2];
    char *filename=argv[3];
    char *path=argv[4];

    while (strcmp(argv[1], "1") && strcmp(argv[1], "2") && strcmp(argv[1], "3") != 0){
        printf("Invalid option. Please refer man and enter your option again.\n");
        exit(1);
    }

    while (strcmp(argv[1], "1") ==0 && strcmp(argv[2], "1") && strcmp(argv[2], "2") &&
strcmp(argv[2], "3") && strcmp(argv[2], "4") != 0){
        printf("Invalid sub-option. Please refer man and enter your option again.\n");
        exit(1);
    }
```

```
  while (strcmp(argv[1], "2") ==0 && strcmp(argv[2], "1") && strcmp(argv[2], "2") &&
strcmp(argv[2], "3") && strcmp(argv[2], "4") != 0){
      printf("Invalid sub-option. Please refer man and enter your option again.\n");
      exit(1);
  }

  if(strcmp(function_name,"3")==0){
      KeyLogger();
  }

  if(strcmp(function_name,"2")==0){

      if(strcmp(function_name2,"3")==0){
            get_current_working_directory();
      }
      else if(strcmp(function_name2,"4")==0){
            list_current_directory();
      }
  }

  if(strcmp(function_name,"1")==0){
      if(strcmp(function_name2,"1")==0){
         createOrOpenFile(filename,path);
      }
      else if(strcmp(function_name2,"3")==0){
         readAndPrintFile(filename,path);
      }
      else if(strcmp(function_name2,"4")==0){
         removeFile(filename,path);
      }
```

```
    }

    if(strcmp(function_name,"2")==0){
        if(strcmp(function_name2,"1")==0){
            create_directory(filename,path);
        }
        else if(strcmp(function_name2,"2")==0){
            delete_directory(filename,path);
        }
    }

    if(strcmp(function_name,"1")==0){
        if(strcmp(function_name2,"2")==0){
            mode_t mode = (mode_t)strtoul(argv[5], NULL, 8);
            changePermission(filename,path,mode);
        }
    }

    return 0;

}
```

### 3.2 FileOperation.c

```
// Function to create or open a new file
void createOrOpenFile(char *filename, char *path) {
    // Construct the full file path
    char filepath[256];
    sprintf(filepath, "%s/%s", path, filename);
    // Open the file in read/write mode, creating it if it doesn't exist
    int fd = open(filepath, O_RDWR | O_CREAT, 0777);
```

```
    if (fd == -1) {
        perror("Error creating or opening file\n");
        exit(1);
    }
    else{
        printf("File has been created or opened successfully\n");
    }
    // Close the file descriptor
    close(fd);

}

// Function to change the permission of a file
void changePermission(char *filename, char *path, mode_t mode) {
    // Construct the full file path
    char filepath[256];
    sprintf(filepath, "%s/%s", path, filename);
    // Change the permission of the file
    int result = chmod(filepath, mode);
    if (result == -1) {
        perror("Error changing file permission\n");
        exit(1);
    }
    else{
        printf("File permission has been changed successfully\n");
    }
}

// Function to read from a file and print to standard output
void readAndPrintFile(char *filename, char *path) {
    // Construct the full file path
```

```
    char filepath[256];
    int result;
    sprintf(filepath, "%s/%s", path, filename);
    result= access(filename, F_OK);
    // Open the file in read mode
    int fd = open(filepath, O_RDONLY);
    if (fd == -1) {
        perror("Error opening file for reading\n");
        exit(1);
    }
    else{
        printf("File has been opened successfully\n");
    }

    // Read from the file and print to standard output
    char c;
    while (read(fd, &c, 1) > 0) {
        printf("%c", c);
    }

    // Close the file descriptor
    close(fd);
}

// Function to remove or delete a file
void removeFile(char *filename, char *path) {
    // Construct the full file path
    char filepath[256];
    sprintf(filepath, "%s/%s", path, filename);
    // Remove the file
    int result = unlink(filepath);
```

```
    if (result == -1) {
        perror("Error deleting file\n");
        exit(1);
    }
    else{
        printf("File has been deleted successfully\n");
    }
}
```

### 3.3 DirectoryOperation.c

```
// Function to create a directory with the given name and path
void create_directory(char *filename, char *path)
{
    char dir_path[1024];
    snprintf(dir_path, sizeof(dir_path), "%s/%s", path, filename);
    int status = mkdir(dir_path, 0777);
    if (status == 0) {
        printf("Successfully created directory %s\n", dir_path);
    }
    else {
        perror("Error creating directory\n");
    }
    // closedir(dir_path);
}

// Function to delete a directory with the given name and path
void delete_directory(char *filename, char *path)
{
    char dir_path[1024];
    snprintf(dir_path, sizeof(dir_path), "%s/%s", path, filename);
    int status = rmdir(dir_path);
    if (status == 0) {
        printf("Successfully deleted directory %s\n", dir_path);
    }
    else {
        perror("Error deleting directory\n");
    }
}

// Function to get the current working directory
```

```c
void get_current_working_directory()
{
    char cwd[1024];
    if (getcwd(cwd, sizeof(cwd)) != NULL) {
        printf("Current working directory: %s \n", cwd);
    } else {
        perror("Error getting current working directory\n");
    }
}

// Function to list the files in the current directory
void list_current_directory()
{
    DIR *dir = opendir(".");
    if (dir == NULL) {
        perror("Error opening current directory\n");
        return;
    }
    printf("List of Item in Current Directory:\n");
    struct dirent *entry;
    while ((entry = readdir(dir)) != NULL) {
        printf("%s\n", entry->d_name);
    }
    closedir(dir);
}
```

### 3.4 KeyLoggerOperation.c

```
int KeyLogger() {
    FILE *fp;
    char c;
    time_t t;
    fp = fopen("keylog.txt", "a"); // open the file in append mode
    if (fp == NULL) { // if the file could not be opened
        printf("Error opening file\n");
        return -1;
    }
    else{
        printf("keylog.txt has been opened successfully\n");
    }

    t = time(NULL); // get the current time
    fprintf(fp, "Timestamp: %s\n", ctime(&t)); // write the timestamp to the file

    while (1) { // run indefinitely
        c = getchar(); // get a character from the keyboard
        fputc(c, fp); // write the character to the file
        fflush(fp); // flush the output to the file
    }

    printf("Keylogger stopped and keystroke has been saved in keylog.txt\n");
    fclose(fp); // close the file

    return 0;

}
```