Institute of Space Technology, Islamabad

BS CS-01

Operating Systems

Assignment 04

Due Date: Wednesday, 4th January, 2023

Submitted by: Faraz Ahmad Qureshi

Regno: (200901045)

Section: B

Instructor: Ma’am Asia

Question:

You have to create four threads other than main thread.

1. Input thread
2. Reverse thread
3. Capital thread
4. Shift thread

Input thread will take string input from user, reverse thread will reverse the string and output it, capital thread will capitalize the characters of string and output it and shift thread will shift each character of the string two time (e.g. a will become c) and output it. All the threads wait for input thread when input thread finishes his task all the waiting thread start their work simultaneously. You also have to handle the exceptions of input thread. Also take care the state of each thread. Do not waste your memory resources.

Solution:

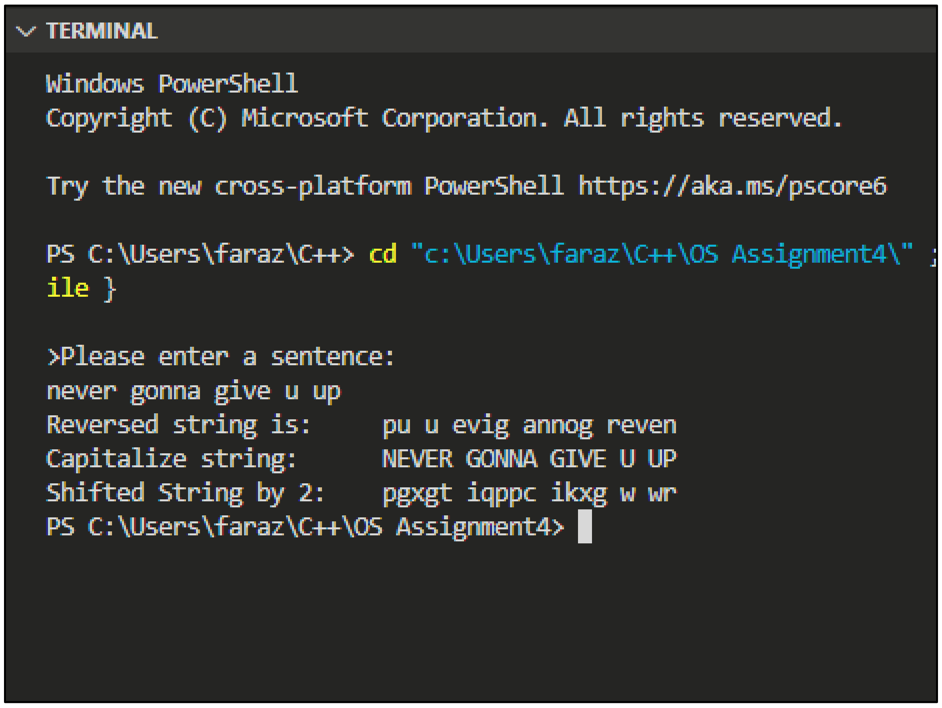
This program helps us better understand multithreading concept in operating system and how multiple different functions are happening concurrently to commit even the most basic OS tasks such as inputting a string and manipulating it. We use C++ as well as thread library for this task. Our other dependencies include the C++ string class and certain string class methods which help solve this task. We use multiple thread objects and multiple functions which perform a specific task. These functions include our Data entering or input function, our reverse string function, our capitalize function and our Shift input by 2 digits function. To better understand the program, we can take a look at our code:

Code:

```
#include <iostream>

#include <thread>

#include <string>

using namespace std;

void enterInput(string&);

void reverseInput(string&);

void capitalInput(string);

void shiftInput(string&);

int main(){

    string userString;

    //create 4 threads other than main thread.

    thread threadInput(enterInput,ref(userString)); //passing string as reference to input
thread
```

```
    threadInput.join(); //once input thread is done then we pass string to other threads (and
avoid race conditions)

    thread threadReverse(reverseInput,ref(userString));

    threadReverse.join();

    thread threadCapital(capitalInput,userString);

    threadCapital.join();

    thread threadShift(shiftInput,ref(userString));

    threadShift.join();

}

void shiftInput(string& str){

    string newString;

    cout << "Shifted String by 2:\t";

    for (int i = 0; i < str.length(); i++)

    {

        if (str[i] == ' ')  newString[i] = ' ';

        else    newString[i] = ((unsigned char)str[i])+2;

         cout << newString[i];      }      }

void capitalInput(string s){

    cout << "Capitalize string:\t";

    for (auto & c: s) c = (char)toupper(c);

    cout  << s << endl;

}

void reverseInput(string& s){

    string reversed(s.rbegin(),s.rend());

    cout << "Reversed string is:\t" << reversed<< "\n" ;

}


void enterInput(string &s)

{

    //we need to take string user input

    cout << "\n>Please enter a sentence:\n";

    getline(cin,s);

}
```

Output:



TERMINAL

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\faraz\C++> cd "c:\Users\faraz\C++\OS Assignment4\" ;
ile }

>Please enter a sentence:
never gonna give u up
Reversed string is:     pu u evig annog reven
Capitalize string:      NEVER GONNA GIVE U UP
Shifted String by 2:    pgxgt iqppc ikxg w wr
PS C:\Users\faraz\C++\OS Assignment4>
```

Our input function has no exceptions therefore we don’t need to separately deal with thread exceptions however in case our function has errors, we can add a condition to ensure other functions do not start running until the error is resolved.

As we can see in the output, our user enters a string. This can be any string include numbers and characters which is one line long. Then we reverse the string using our reverse string function. We can simultaneously capitalize and shift our string by 2 digits but to better format the output we have opted to display them one at a time. These threads can function concurrently and lessen our processing time resulting in a more efficient use of our program.