AMRITA VISHWA VIDYAPEETHAM UNIVERSITY

CASE STUDY – OS/2 WARP REVIEW-2

COURSE CODE : 19CSE213

COURSE NAME: OPERATING SYSTEMS

TEAM MEMBERS :

S.No	Name of the Student	Roll No.
1.	RAVELLA ABHINAV	CB.EN.U4CSE19453
2.	PENUGONDA KOUSHIK	CB.EN.U4CSE19449
3.	SINGADI SHANTHAN REDDY	CB.EN.U4CSE19459
4.	NUSUM KARTHIK	CB.EN.U4CSE19444

THREADS

R. Abhinav
CB.EN.U4CSE19453

- **What is a thread?**
  - A thread is a path of execution within a process. A process can contain multiple threads.
  - The thread consists of a program counter, stack and a set of registers
- **Why threads:-**
  - The main and primary difference between a thread and a process is that thread run in a shared memory space and process run in seperate memory space.
  - Threads share lot of things among themselves
    Eg:- Code section, data section, OS resources etc.,
- **How do we use threads in our program?**

  The API's for using threads &/ managing threads are provided by a library called as "thread library".
- These 'thread libaries can be implemented by user space &/ by kernel space.
- Three most popular thread libraries that are used are as follows:-
  (i) Posix thread
  (ii) Java threads
  (iii) Win32 threads

Windows 7 threads:-

- The windows 7 uses Win-32 thread library to create and manage threads.
- This library can be used by including the following header file in the program:

```
# include <windows.h>
```

Some system calls:-

① createThread ( ) :-

Usage:- This system call is to create a thread to execute within the virtual address space of the calling process.

Parameters that this system call has :-

① Security attributes (default - NULL)

② Thread stack size (default - 0)

③ Thread start routine (function)

④ variable (pointer to variable) that needs to be passed to the thread.

⑤ Creation Flags (default - 0)

⑥ This is a pointer to a variable that receives the thread-id.

- Security attributes:-

This pointer to the structure determines whether the returned handle can be inherited by the child process or not. If 'NULL' the handle cannot be inherited.

2. Thread stack size :-

If this parameter is zero, the thread uses the size of executable. This is basically the initial size of the stack.

3. Thread start routine :-

This is basically the function that has to be executed by the thread.

4. Parameters :-

A pointer to variable that is passed to the thread.

5. Creation Flags :-

This parameter controls the creation of threads. If '0' is passed gets executed immediately after creation.

6. Thread-id :-

A pointer to a variable that recieves the thread-id.

Return-value :-

If thread execution succeed return value is a handle to a new thread & fails returns "NULL"

- Close Handle ( ) :

Usage: This function closes the open handle

parameter :-

Valid handle : to an object that is open.

This system call returns a non-zero value if the function succeeds

- Wait for Single Object ( ) :

* Usage :

waits until the specified object is in the State (or) time out interval elapses

* Parameters :-

Handle : Ther return value that thread creation gives i.e., Thread Handle in this case.

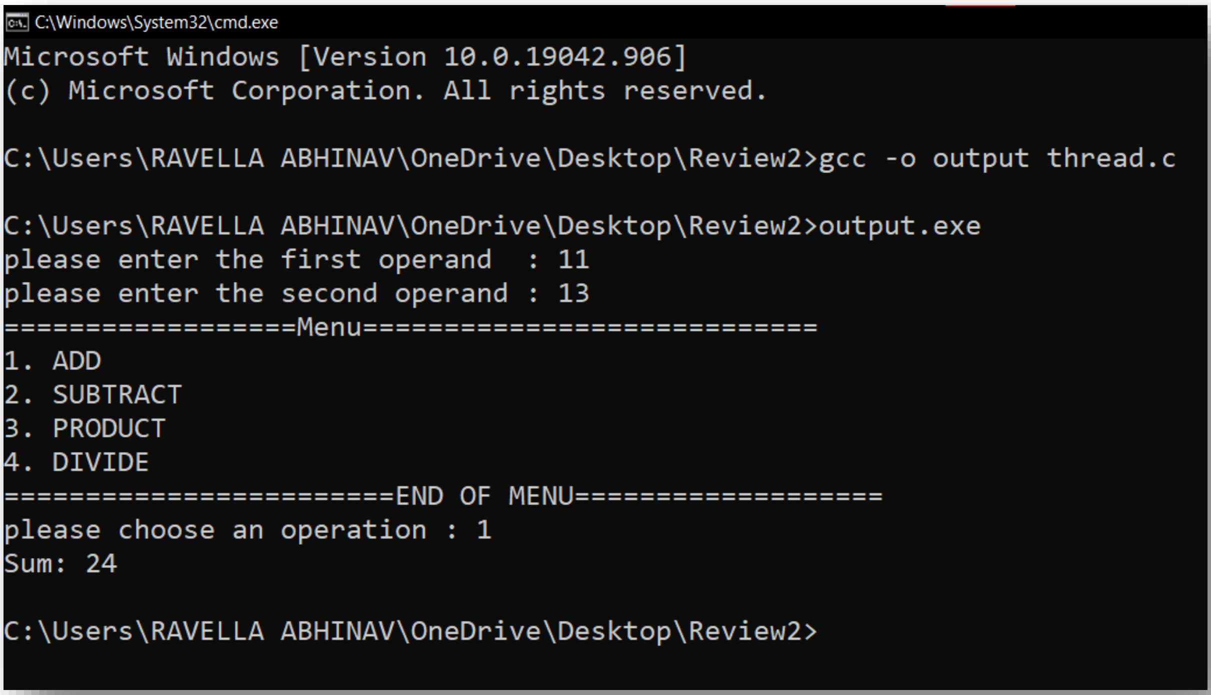
Time out in milliseconds. If a non-zero value is specified. the function waits until the object is signaled (or) interval elapses.

THREADS

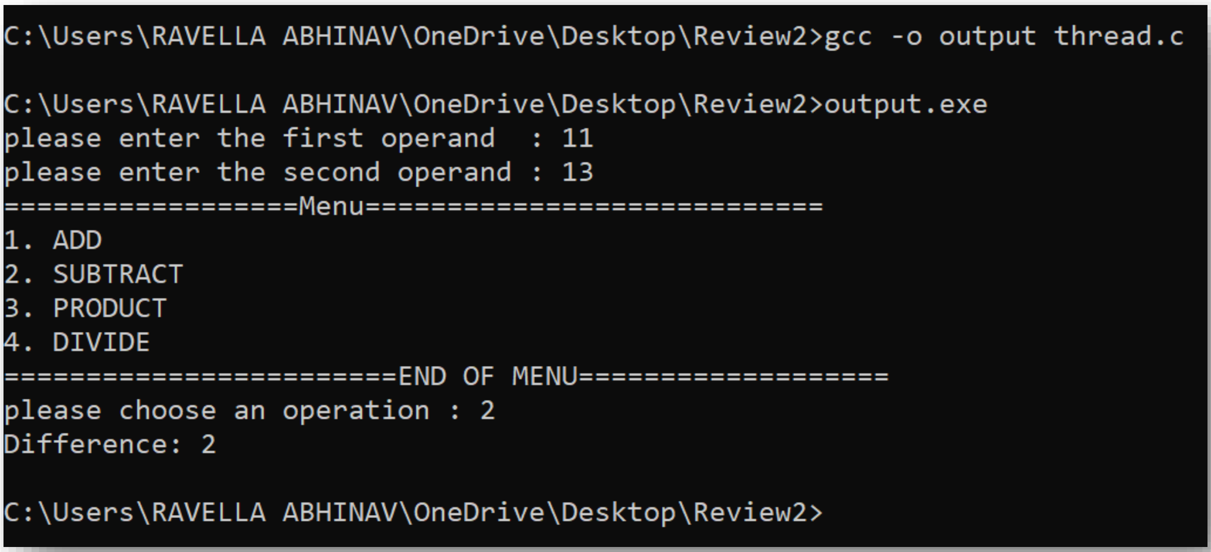
NAME	RAVELLA ABHINAV
ROLL No.	CB.EN.U4CSE19453

```c
#include <windows.h>
#include <stdio.h>

DWORD Sum_of;
DWORD Diff;
DWORD product;
DWORD divi;

typedef struct parameters{
    int operand1;
    int operand2;
}param, *para;


DWORD WINAPI Sum(LPVOID Param){

    para Upper = (para)Param;

    Sum_of = Upper->operand1 + Upper->operand2;

    return 0;
}


DWORD WINAPI Difference(LPVOID Param){
    para Upper = (para)Param;

    if (Upper->operand1 > Upper->operand2){
        Diff = Upper->operand1 - Upper->operand2;
    }
    else{
        Diff = Upper->operand2 - Upper->operand1;
    }
    return 0;
}


DWORD WINAPI Product(LPVOID Param){
    para Upper = (para)Param;
    product = Upper->operand1 * Upper->operand2;
    return 0;
}


DWORD WINAPI Divide(LPVOID Param){
    para Upper = (para)Param;

    if (Upper->operand2 == 0){
        divi = -1;
    }
    else{
        divi = Upper->operand1/Upper->operand2;
    }
    return 0;
}


int main(){
    DWORD thread_id;
    HANDLE ThreadHandle;
    HANDLE ThreadHandle1;
    HANDLE ThreadHandle2;
    HANDLE ThreadHandle3;

    struct parameters param;
    struct parameters *params;

    params = &param;

    printf("please enter the first operand  : ");
    scanf("%d",&params->operand1);

    printf("please enter the second operand : ");
    scanf("%d",&params->operand2);

    int operation;

    printf("==================Menu============================\n");
    printf("1. ADD\n");
    printf("2. SUBTRACT\n");
    printf("3. PRODUCT\n");
    printf("4. DIVIDE\n");
    printf("========================END OF MENU===================\n");

    printf("please choose an operation : ");
    scanf("%d",&operation);

    if (operation == 1){

        ThreadHandle = CreateThread(
                NULL,      //default security attribute
                0,         //default stack size of the thread
                Sum,    //function that is called by the thread
                params,       //structure of arguments that are passed into
the threads
                0,          //default creation flags
                &thread_id      // the thread returns the thread id here
        );

        //this system call will wait for the thread to finish, for infinite
amount of time.

        WaitForSingleObject(ThreadHandle,INFINITE);

        //closing the thread here
        CloseHandle(ThreadHandle);

        printf("Sum: %lu\n",Sum_of);
    }

    else if(operation == 2){

        ThreadHandle1 = CreateThread(
                NULL,     //default security attribute
                0,        //default stack size of the thread
                Difference,   //function that is called by the thread
                params,      //structure of arguments that are passed into
the threads
                0,         //default creation flags
                &thread_id     // the thread returns the thread id here
        );

        //this system call will wait for the thread to finish, for infinite
amount of time.

        WaitForSingleObject(ThreadHandle1,INFINITE);

        //closing the thread here

        CloseHandle(ThreadHandle1);

        printf("Difference: %lu\n",Diff);
    }

    else if(operation == 3){
        ThreadHandle2 = CreateThread(
                NULL,     //default security attribute
                0,        //default stack size of the thread
                Product,   //function that is called by the thread
                params,      //structure of arguments that are passed into
the threads
                0,         //default creation flags
                &thread_id     // the thread returns the thread id here
        );

        //this system call will wait for the thread to finish, for infinite
amount of time.

        WaitForSingleObject(ThreadHandle2,INFINITE);

        //closing the thread here

        CloseHandle(ThreadHandle2);

        printf("product: %lu\n",product);
    }

    else{
        ThreadHandle3 = CreateThread(
                NULL,     //default security attribute
                0,        //default stack size of the thread
                Divide,   //function that is called by the thread
                params,      //structure of arguments that are passed into
the threads
                0,         //default creation flags
                &thread_id     // the thread returns the thread id here
        );

        //this system call will wait for the thread to finish, for infinite
amount of time.

        WaitForSingleObject(ThreadHandle3,INFINITE);

        //closing the thread here

        CloseHandle(ThreadHandle3);

        printf("Quotient: %lu\n",divi);
    }

}
```

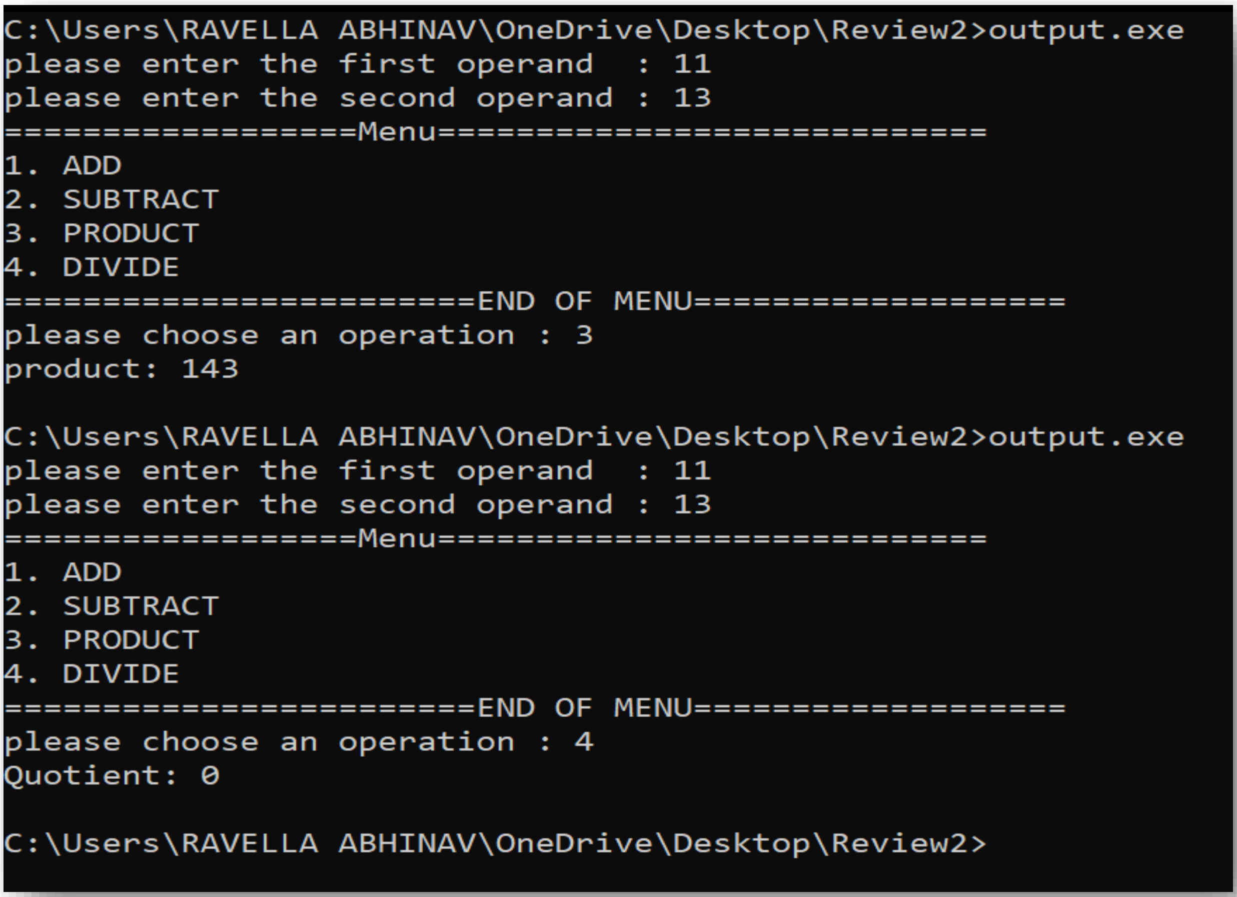
Output :



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.906]
(c) Microsoft Corporation. All rights reserved.

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>gcc -o output thread.c

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>output.exe
please enter the first operand  : 11
please enter the second operand : 13
==================Menu=============================
1. ADD
2. SUBTRACT
3. PRODUCT
4. DIVIDE
========================END OF MENU===================
please choose an operation : 1
Sum: 24

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>
```



```
C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>gcc -o output thread.c

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>output.exe
please enter the first operand  : 11
please enter the second operand : 13
==================Menu============================
1. ADD
2. SUBTRACT
3. PRODUCT
4. DIVIDE
========================END OF MENU===================
please choose an operation : 2
Difference: 2

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>
```



```
C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>output.exe
please enter the first operand  : 11
please enter the second operand : 13
==================Menu============================
1. ADD
2. SUBTRACT
3. PRODUCT
4. DIVIDE
========================END OF MENU===================
please choose an operation : 3
product: 143

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>output.exe
please enter the first operand  : 11
please enter the second operand : 13
==================Menu============================
1. ADD
2. SUBTRACT
3. PRODUCT
4. DIVIDE
========================END OF MENU===================
please choose an operation : 4
Quotient: 0

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>
```