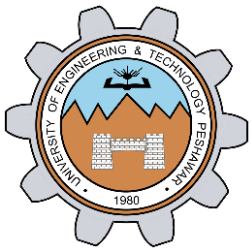
OPERATING SYSTEMS LAB 08

Threads Creation and Execution



Spring 2024

Submitted by: **Hassan Zaib Jaoon**

Registration No: **22pwsce2144**

Class Section: **A**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Engr. Abdullah Hamid

Month Day, Year (16 May, 2024)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

OBJECTIVES:

This lab examines aspects of threads and multiprocessing (and multithreading). The primary objective of this lab is to implement Thread Management Functions:

Creating Threads

Terminating Thread

Execution Thread Identifiers

Joining Threads

Title:

Threads Creation and Execution

Task1:

Example: Pthread Creation and Termination:

Code:

```c
#include <stdio.h>
#include <pthread.h>
void *kidfunc(void *p)
{
    printf ("Kid ID is ---> %d\n", getpid( ));
}
int main ( )
{
   pthread_t kid ;
pthread_create (&kid, NULL, kidfunc, NULL);
printf ("Parent ID is ---> %d\n", getpid( )) ;
pthread_join (kid, NULL) ;
printf ("No more kid!\n") ;
return 0;
}
```

Output:

```
Parent ID is ---> 4610
Kid ID is ---> 4610
No more kid!
```

Question: Are the process id numbers of parent and child thread the same or different? Give reason(s) for your answer.

Answer:

Yes, we observe the process id will be same because the process is only one thread is multiple.

Example 02:

```c
#include <stdio.h>
#include <pthread.h>
int glob_data = 5 ;
void *kidfunc(void *p)
{
    printf ("Kid here. Global data was %d.\n", glob_data) ;
glob_data = 15 ;
   printf ("Kid Again. Global data was now %d.\n", glob_data) ;
}
int main ( )
{
    pthread_t kid ;
    pthread_create (&kid, NULL, kidfunc, NULL) ;
    printf ("Master thread here. Global data = %d\n", glob_data) ;
    glob_data = 10 ;
    pthread_join (kid, NULL) ;
   printf ("End of program. Global data = %d\n", glob_data) ;
   return 0;
}
```

Output:

```
Master thread here. Global data = 5
Kid here. Global data was 10.
Kid Again. Global data was now 15.
End of program. Global data = 15
```

Question:

Do the threads have separate copies of glob_data? Why? Or why not?

Answer:

In thread global variable always shared between the all existing thread present in program at current time so the global variable is shared between all thread.

Example 03:

Multiple Threads: The simple example code below creates 5 threads with the pthread_create( ) routine. Each thread prints a "Hello World!" message, and then terminates with a call to pthread_exit( ).

```c
#include <pthread.h>
#include <stdio.h>
#define NUM_THREADS 5
void *PrintHello(void *t)
{
    long threadid;
    threadid=(long)t;
    printf("\n%ld: Hello World!\n", threadid);
    pthread_exit(NULL);
}
int main( )
{
    pthread_t   threads[NUM_THREADS];
    int rc, t;
    for(t=0; t < NUM_THREADS; t++)    {
            printf ("Creating thread %d\n", t);
            rc = pthread_create (&threads[t], NULL, PrintHello, (void*)t);
            if (rc) {
                printf("ERROR; return code from pthread_create() is %d\n", rc);
                exit(-1);
```

Output:

```
Creating thread 0
Creating thread 1

0: Hello World!
Creating thread 2
Creating thread 3
Creating thread 4

1: Hello World!

3: Hello World!

2: Hello World!

4: Hello World!
```

Problem 1:

The following Box #1 program demonstrates a simple program where the main thread creates another thread to print out the numbers from 1 to 20. The main thread waits till the child thread finishes.

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
void *ChildThread(void *argument)
{
int i;
for ( i = 1; i <= 20; ++i ){
printf(" Child Count - %d\n", i);
}
pthread_exit(NULL);
}
int main(void)
{
pthread_t hThread; int ret;
ret=pthread_create(&hThread, NULL, ChildThread, NULL);
if (ret!=0)
{
printf("thread creation failed\n");
exit(1);
}
pthread_join(hThread, NULL);
printf("Parent is continuing....\n");
return 0;
}
```

Output:

```
 Child Count - 1
 Child Count - 2
 Child Count - 3
 Child Count - 4
 Child Count - 5
 Child Count - 6
 Child Count - 7
 Child Count - 8
 Child Count - 9
 Child Count - 10
 Child Count - 11
 Child Count - 12
 Child Count - 13
 Child Count - 14
 Child Count - 15
 Child Count - 16
 Child Count - 17
 Child Count - 18
 Child Count - 19
 Child Count - 20
Parent is continuing....
```

PROBLEM 02:

Write a program Box # 2 by removing pthread_exit function from child thread function and check the output? Is it the same as output when pthread_exit is used? If so Why? Explain?

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

void *ChildThread(void *argument) {
    int i;
    int arg = *((int *)argument); // Retrieve the argument pas
    for (i = 0; i < 10; i++) {
        printf("Child thread is running... %d, Argument: %d\n"
    }
    pthread_exit(NULL); //with exit function
}

int main(void) {
    pthread_t hThread;
    int arg = 42; // Example argument to pass to the thread

    // Create the child thread
    pthread_create(&hThread, NULL, ChildThread, (void *)&arg);

    // Wait for the child thread to finish
    pthread_join(hThread, NULL);

    printf("Master thread is continuing....\n");
    return 0;
}
```

Output:

```
Child Count - 1
Child Count - 2
Child Count - 3
Child Count - 4
Child Count - 5
Child Count - 6
Child Count - 7
Child Count - 8
Child Count - 9
Child Count - 10
Child Count - 11
Child Count - 12
Child Count - 13
Child Count - 14
Child Count - 15
Child Count - 16
Child Count - 17
Child Count - 18
Child Count - 19
Child Count - 20
Parent is continuing....
```

CSE 302L: Operating Systems Lab

LAB ASSESSMENT RUBRICS

Marking Criteria	Exceeds expectation (2.5)	Meets expectation (1.5)	Does not meet expectation (0)	Score
1. Correctness	Program compiles (no errors and no warnings). Program always works correctly and meets the specification(s). Completed between 81-100% of the requirements.	Program compiles (no errors and some warnings). Some details of the program specification are violated, program functions incorrectly for some inputs. Completed between 41-80% of the requirements.	Program fails to or compile with lots of warnings. Program only functions correctly in very limited cases or not at all. Completed less than 40% of the requirements.	
2. Delivery	Delivered on time, and in correct format (disk, email, hard copy etc.)	Not delivered on time, or slightly incorrect format.	Not delivered on time or not in correct format.	
3. Coding Standards	Proper indentation, whitespace, line length, wrapping, comments and references.	Missing some of whitespace, line length, wrapping, comments or references.	Poor use of whitespace, line length, wrapping, comments and references.	
4. Presentation of document	Includes name, date, and assignment title. Task titles, objectives, output screenshots included and good formatting and excellently organized.	Includes name, date, and assignment title. Task titles, objectives, output screenshots included and good formatting.	No name, date, or assignment title included. No task titles, no objectives, no output screenshots, poor formatting.	

Instructor:

Name: Engr. Abdullah Hamid Signature: _____