**LAB REPORT NO 7**



Submitted by:  **Muhammad Ali**

Registration No: - **19PWCSE1801**

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.Mian Ibad Ali shah**

Data:(15,07,2021)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**Operating Systems LAB 7**

**Threads Creation and Execution**

**Objective: -**

* To understand thread creation.
* To learn how to exit, terminate a thread.
* To learn thread waiting.
* How to passed argument in thread.

**Thread: -**

A thread is a semi-process, that has its own stack, and executes a given piece of code. Unlike a real process, the thread normally shares its memory with other threads (where as for processes we usually have a different memory area for each one of them). A Thread Group is a set of threads all executing inside the same process. They all share the same memory, and thus can access the same global variables, same heap memory, same set of file descriptors, etc. All these threads execute in parallel (i.e. using time slices, or if the system has several processors, then really in parallel).

**Pthreads: -**

Pthreads are defined as a set of C language programming types and procedure calls. Vendors usually provide a Pthreads implementation in the form of a header/include file and a library which you link with your program

**Assignment no 1: -**

#include <pthread.h>

#include <stdio.h>

void \*ChildThread(void \*argument)

{

      int i;

      for (i = 0; i < 10; ++i ){

   printf(" Child thread - %d\n", i);

}  
    pthread\_exit(NULL);

}  
  
int main(void)

{

        pthread\_t   hThread;        int   ret;

        ret=pthread\_create(&hThread, NULL, (void \*)ChildThread, NULL); /\* Create

Thread \*/

        if (ret < 0)

printf("Thread Creation Failed\n");   return 1;

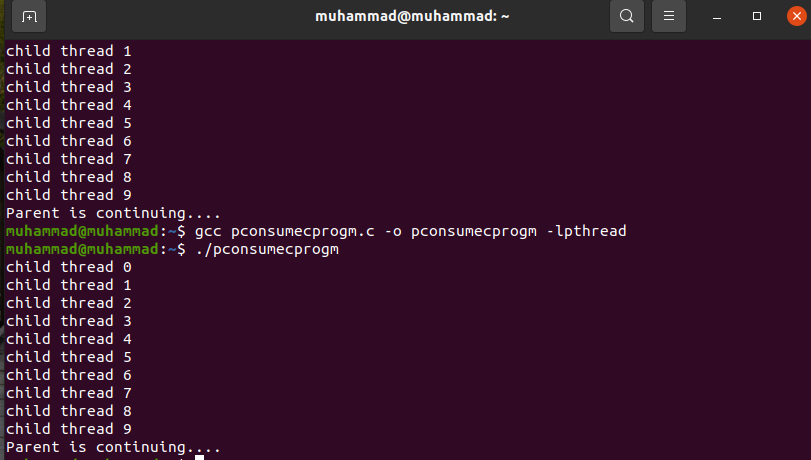
        pthread\_join (hThread, NULL);  /\* Parent waits for  \*/

          printf("Parent is continuing....\n");

return 0;

}

**Output: -**

****

**Explanation: -**

Box1 print print child thread from 0 to 10 as I limited print function to 10 instead of 20. In the viod pointer function loop is executed 10 times and pthread exit when it become NULL, in the main function I declare htread variable with prototype pthread\_t for a thread and ret variable is for to store a thread then I apply condition that if ret is less than 0 then print thread creation failed else wait for its parent.

**Assignment no 2: -**

#include <pthread.h>

#include <stdio.h>

void \*ChildThread (int argument){

int i;

for(int i=0; i<10; i++){

printf("child thread %d\n", i);

}

pthread\_exit(NULL);

}

int main(void){

pthread\_t hThread;

pthread\_create (&hThread,NULL,(void\*)ChildThread,NULL);

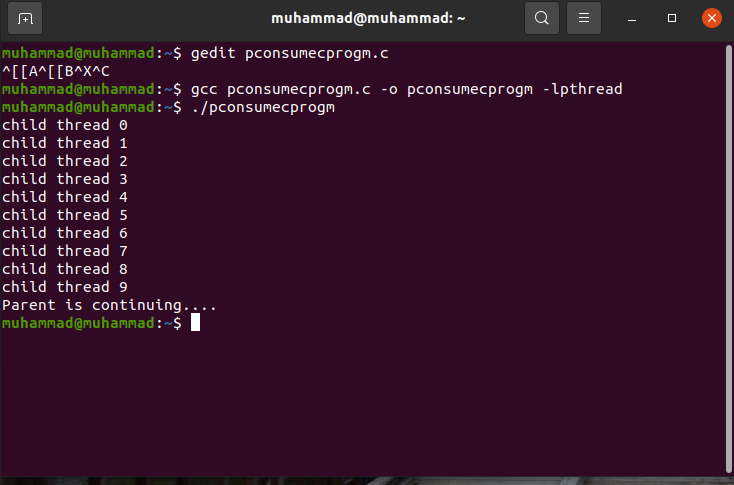
pthread\_join (hThread, NULL);

printf ("Parent is continuing....\n");

return 0;

}

**Output: -**

****

**Explanation: -**

Its perform the same functionality as above.

**Assignment no 3: -**

#include <pthread.h>

#include <stdio.h>

void \*ChildThread (int argument){

int i;

for(int i=0; i<10; i++){

printf("child thread %d\n", i);

}

}

int main(void){

pthread\_t hThread;

pthread\_create (&hThread,NULL,(void\*)ChildThread,NULL);

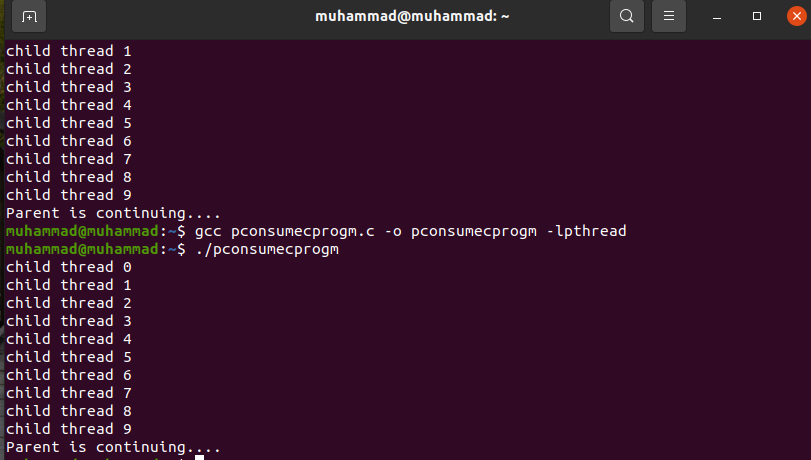
pthread\_join (hThread, NULL);

printf ("Parent is continuing....\n");

return 0;

}

**Output: -**



**Explanation: -**

It will also print the same b/c exit function is outside the loop in the above assignment but if we remove it that will not effect the loop.