Lab - 05

Objectives:

1. Understanding Linux Threads.
2. Creation of threads using the `pthread` library Threads in Linux

Threads

Task 01: **[30 Marks]**

Which one is faster and why?

1. The creation of 1000 child processes using a process fan.
2. The Creation of 1000 child threads.

You need to write down the code to create 1000 child processes and the code to create 100 child threads, and measure the time that each process took to execute.

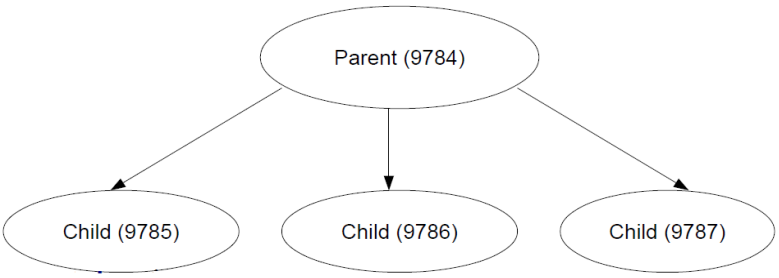
Help: To measure time see the code below:

```
time_t t1, t2;
time(&t1);
// <Your Code Here>
time(&t2);
printf("It took us %d seconds to complete the task\n", t2-t1);
```

Help: For the understanding of the process fan:

In Process fan, the parent is responsible for every fork. Child processes will break, while parent process will iterate again to create another child process. It is used for simultaneous execution.

```
for (int i=1;i<=n;i++){
    if(fork() == 0){
        break;
    }
}
fprintf(stderr, "PID=%ld, PPID=%ld\n",(long)getpid(), (long)getppid());
```



Task 02: **[30 Marks]**

Consider the following information:

```
#define MAX_THREAD 3
#define ROWS 3
#define COLS 4
int mat_A[ROWS][COLS] = {1,2,3,4,1,6,3,2,9,2,3,4};
int mat_B[ROWS][COLS] = {5,4,6,2,4,1,2,1,0,7,5,2};
int sum[ROWS][COLS];
```

Write a code that will make an array of threads of size **MAX_THREAD** and compute the sum of **mat_A** and **mat_B** and store it in **sum**. (For sum, you have to add the corresponding elements of both matrices).

Help: To create array of threads and work with them see code below:

```
pthread_t tids[MAX_THREAD];
for (i = 0; i < MAX_THREAD; i++)
      pthread_create(&tids[i], NULL, &function, (void*)NULL);
```

Task 03: **[40 Marks]**

Consider the given source codes in the **Task03.zip** shared with your Lab

Task3_linear_search.c is a C program that searches **a value at random index** in an array of large size using **linear search**. One way to improve the performance is to use *binary search* BUT we can also enhance its performance by creating *multiple threads*.

Since this Lab is related to threads, you are going to improve the performance by creating multiple threads.

Task03_solution.c contains basic code structure that will help you to solve the task.

You have to create **four** threads that will simultaneously search for the **key** in the array. Each thread will search in a different range, for example **thread 0** will search for the key from index 0 to 250000 and **thread 1** will search from **250001** to **500000**, and so on.

Note :

```
struct arguments
{
    int key;
    int s, e;
}
```

1. The **struct argument** will be passed to every thread. **int key** will store the value that has to be searched in the array. you don't have to assign value to it. The solution file already contains the statement that will assign values to the **key.** you just have to set the **starting** and **ending** point which are **s** and **e** respectively.
2. Add your code under the Comments in the given solution file.
3. If one of your threads finds the value in the array the remaining threads should exit too and you can do this by using the global **flag** variable that is already initialized in the solution file.

```
void * linear(void * arg)
{
    struct arguments args = *(struct arguments *) arg;
}
```

1. The above code snippet is for the function you will be executing in each thread **`struct arguments args = *(struct arguments *) arg`** is necessary since our argument struct will be **type casted** into **`void *`** when it will be passed to **pthread_create** in the main **thread.** Complete this function to complete the task.
2. Look at code in the file **Task03_arguments.c** if you don't know how arguments are passed to threads.