

**Faculty of** **Technology and Engineering**

# U & P U. Patel Department of Computer Engineering

Date: 14 / 08 / 2022

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Academic Year | : | 2022-23 | Semester | : | 5 |
| Course code | : | CE354 | Course name | : | Operating System |

PRACTICAL – 8

**Aim: Compare the Execution of single Process with threads execution.**

**THREAD**

Threading is a light weight process which shares all the section of the process except for the stack. A process can have multiple threads.

**Fork Vs Thread**

While threads can execute in parallel with same context. Also, memory and other resources are shared between the threads causing less overhead. A thread process is considered a sibling while a forked process is considered a child. Also, threads are known as light-weight processes as they don't have any overhead as compared to processes (as it doesn't issue any separate command for creating completely new virtual address space). A single process can have multiple threads. For all threads of any process, communication between them is direct. While process needs some interprocess communication mechanism to talk to other processes. Thought, threads seem to be more useful for any reason, do note that changes in any thread may lead to changes in other threads of the same process. While, changes in child processes is independent as parent process has its own execution copy.

|  |  |
| --- | --- |
| 1. | #include <stdio.h>  #include <stdlib.h>  #include <pthread.h>  #define NO\_OF\_THREADS 5  #define ELEMENTS 100  static int arr[ELEMENTS];  static int sum;  int retval[5]={1,2,3,4,5};  int j=0;  void\* thread\_sum(void\* arg)  {  int i;  int\* current\_thread\_data =(int\*)arg;  printf("-> Current thread no is : %d\n", current\_thread\_data[j]);  int end= (current\_thread\_data[j]) \* (ELEMENTS / NO\_OF\_THREADS);  int start = end - (ELEMENTS / NO\_OF\_THREADS);  printf("-> Here we will calculate the sum of %d to %d\n", arr[start], arr[end - 1]);  int current\_thread\_sum = 0;  for(i = start; i < end; i++)  {  current\_thread\_sum += arr[i];  }  sum+=current\_thread\_sum;  printf("-> current\_thread\_sum : %d\n", current\_thread\_sum);  pthread\_exit(&retval[j]);  j++;  return NULL;  }  int main()  {  int i, thread\_no = 1;  for(i=0;i<ELEMENTS;i++)  arr[i] = i + 1;  pthread\_t id[NO\_OF\_THREADS];  int data\_arr[NO\_OF\_THREADS];  printf("-> Creating %d number of threads...\n",NO\_OF\_THREADS);  for (thread\_no = 1; thread\_no <= NO\_OF\_THREADS; thread\_no++)  {  data\_arr[thread\_no - 1]= thread\_no;  pthread\_create(&id[thread\_no-1],NULL,thread\_sum,&data\_arr[thread\_no-1]);  }  for(i=1;i<=NO\_OF\_THREADS;i++)  pthread\_join(id[i-1],NULL);  printf("-> Total sum: %d\n",sum);  return 0;} |
| Output: |  |
| **Question:** | **Differentiate between fork and thread.** |
| Answer: | |  |  | | --- | --- | | **Fork** | **Thread** | | * Fork() causes creation of a new process called child process. | * A thread is used to run a partial copy of a process with other parts shared with other processes. | | * The return value of fork() is used to distinguish the parent from the child | * Thread does not return anything, It is an entity within a process that consists of the schedulable part of the process. | | * The parent and child processes run independently of each other. | * Threads can execute in parallel with same context. | |