THREADS AND PROCESSES
Network Programming Lab (CS334)

April 15, 2021

Sheen Xavier A
Roll No : 57
University Roll No : TVE18CS058
Department of Computer Science and Engineering
College of Engineering, Trivandrum

Contents

Threads and Processes

1.1 Aim

To get started with the familiarization and implementation of programs related to process and thread by developing a program to create 'n' threads.

1.2 Theory

Process

A process can be defined as an entity which represents the basic unit of work to be implemented in the system. Basically speaking, a process is a program in execution. For example, the original code we write and binary code which we process are both programs. When we actually run the binary code, it becomes a process. Processes are mainly used for 'heavyweight' tasks.

Thread

A thread is a path of execution within a process. It has its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history. It shares with its peer threads few information like code segment, data segment and open files. A thread is considered to be a lightweight process and hence they are particularly used for small tasks. Threads within the same process share the same address space.

1.3 Source Code

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

void *threadContent(void *v)
{
    // The executing thread body
    int *threadId = (int *)v;
    sleep(1); // Suspending the process for 1 sec to show delay
    printf("Now running Thread No : %d ...\n", *threadId);
    return NULL;
}

int main()
{
    int i, n;
    pthread_t threadId; // Stores the thread ID

    // Reading the input for creating n threads
    printf("Number of Threads : ");
    scanf("%d", &n);

    printf("Creating Threads...\n");
    for(i = 1 ; i <= n ; i++)
    {
        printf("-----------------------------\n");
        printf("Creating Thread No : %d\n", i);
        // Creating the thread with threafContent as the body
        pthread_create(&threadId, NULL, threadContent, (void *)&i);
        // Join is used to ensure that the threads wait for
        // the currently executing threads to terminate
        pthread_join(threadId, NULL);
        printf("Escaping Thread No : %d\n", i);
        printf("-----------------------------\n");
    }

    printf("Finished running the threads!\n");
    return 0;
}
```

1.4 Output

```
sheenxavi004@Beta-Station:~$ cc threads.c -lpthread
sheenxavi004@Beta-Station:~$ ./a.out
Number of Threads : 4
Creating Threads...
----------------------------
Creating Thread No : 1
Now running Thread No : 1 ...
Escaping Thread No : 1
----------------------------
----------------------------
Creating Thread No : 2
Now running Thread No : 2 ...
Escaping Thread No : 2
----------------------------
----------------------------
Creating Thread No : 3
Now running Thread No : 3 ...
Escaping Thread No : 3
----------------------------
----------------------------
Creating Thread No : 4
Now running Thread No : 4 ...
Escaping Thread No : 4
----------------------------
Finished running the threads!
```

Figure 1.1: Creating four threads

1.5 Result

The program was developed as per requirement and the output was verified. It was also tested against various test cases.