Lab 8

PROGRAM TO CREATE MULTI-THREADED PROCESS

OBJECTIVES:

1. Implement multithreading using C

2. Study how to use POSIX threads in Linux

BACKGROUND THEORY

A thread is a basic unit of CPU utilization, consisting of a program counter, a stack, and a set of registers,
(and a thread ID.) Traditional (heavyweight) processes have a single thread of con trol. There is one program counter, and one sequence of instructions that can be carried out at any given time.

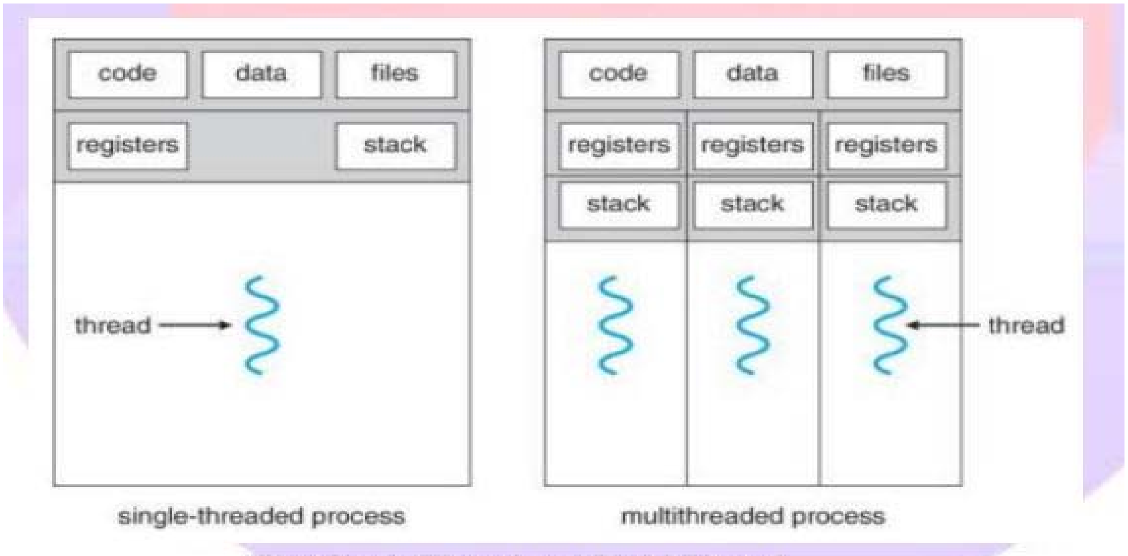

Fig:1 Single Thread and Multi Thread

As shown in Figure 1, multi-threaded applications have multiple threads within a s ingle process, each having their own program counter, stack and set of registers, but sharing common code, data, and certain structures such as open files.

Pthreads

- POSIX standard for thread creation and synchronization.
- These are mere specifications for thread behavior, not its implementation

- Mostly implemented by UNIX type system.
- Windows doesn't support it natively

Source code

For Single Thread

```
#include <stdio.h>
 #include <stdlib.h>
#include <unistd.h> //Header file for sleep(). man 3 sleep for details.
 #include <pthread.h>
//A normal C function that is executed as a thread
// when its name is specified in pthread_create()
void *myThreadFun(void *vargp)
{
sleep(1);
printf("Printing EIC College from Thread \n");
return NULL;
}
int main()
{
pthread_t thread_id;
printf("Before Thread\n");
pthread_create(&thread_id, NULL, myThreadFun, NULL);
pthread_join(thread_id, NULL);
printf("After Thread\n");
exit(0);
}
```

Multi Thread:

A C program to show multiple threads with global and static variables

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

// Let us create a global variable to change it in threads

 int g=0;

// The function to be executed by all threads

void *myThreadFun(void *vargp)

{

// Store the value argument passed to this thread

int  *myid = (int*)vargp;

// Let us create a static variable to observe its changes.

static int s =0;

//Change static and global variables

++s; ++g;

//Print the argument, static and global variables

printf("Thread ID: %d.,Static: %d, Global: %d\n", *myid,++s, ++g);

}

int main()

{

int i;
pthread_t  tid;

// Let us create three threads

for (i=0;i<3;i++)

{

pthread_create(&tid, NULL, myThreadFun, (void *)&tid);

}

pthread_exit(NULL);

return 0;

}
```

OUTPUT: Screenshots of outputs

DISCUSSION:

CONCLUSION: