Lab 5
Threads

As with processes, threads appears to run concurrently; the Linux kernel schedules them asynchronously, interrupting each thread time to time to give others a chance to execute. Threads exists within a process. GNU/Linux implements the POSIX standard thread API (pthreads). All thread functions and data types are declared in the header file *<pthread.h>*. The *pthread* functions are not included in the standard C library; they are in *libpthread*, therefore *-lpthread* should add when linking program.

5.1 Thread Creation

Each thread have their own thread ID as process, thread ID referred by type *pthread_t*.

The *pthread_create* function create new threads. It has following formate.

*int pthread_create (pthread_t *thread, pthread_attr_t *attr, void *(*start_routine) (void*), void *arg);*

The *pthread_exit* function terminates the thread.

*thread_exit(void *return_val);*

The *pthread_join* function waits other process for termination – equivalent of *wait.*

*int pthread_join(pthread_t th, void **thread_return);*

Ex 5.1: Thread Creation (threadc.c)

```
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
struct param
        {
                char ch;        /* The character to print*/
                int count;      /* number of times to print it */
         };

void * printc(void * parameter)         /* prints number of character in stderr*/
         {
                struct param * p = (struct param *) parameter;
                int i;

                for(i=0;i<p->count; ++i)
                                fputc(p->ch, stderr);
                return NULL;
            }
```

```
int main()
        {
              pthread_t thread1_id;
              pthread_t thread2_id;
              struct param thread1_args;
              struct param thread2_args;

              thread1_args.ch = 'T';   /* new thread to print 30000 Ts*/
              thread2_args.count = 30000;
              pthread_create(&thread1_id, NULL, &printc, &thread1_args);

              thread2_args.ch ='t';   /* new thread to print 20000 ts */
              thread2_args.count = 20000;
              pthread_create(&thread2_id, NULL, &printc, &thread2_args);

              pthread_join(thread1_id, NULL)  /* wait first thread to finish*/
              pthread_join(thread_id, NULL) /*wait second thread to finish*/

              return 0;
          }
```

Warning! :   Run this program as : gcc -o threadc threadc.c -lpthread

Assignment #L5

1. Run the program Ex 5.1 and analyze the output; what changes will in your out put when you remove last two line (pthread_join), if any changes, give reason.
2. Write a program using threads that prints sum of numbers up to given positive number n.