**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**OPERATING SYSTEM REPORT #07**

**SUBMITTED BY:**

Pradeep Dahal

017BSCIT029

2nd year/ 4th sem

|  |  |
| --- | --- |
|  | Signature |
| Mr. Rabin Maharjan  (Lecturer) |  |
| Department of Computer Science | |

**SUBMITTED TO:**

# 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.

t*hread\_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<pthread.h>

#include<unistd.h>

void \*fun(void \*para)

{

printf("This is the thread\n");

}

int main()

{ pthread\_t id;

pthread\_create(&id,NULL,&fun,NULL);

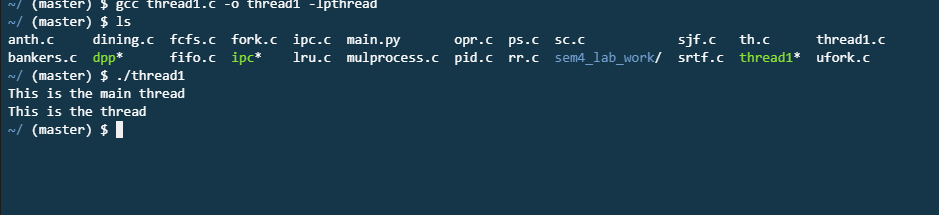
printf("This is the main thread\n");

pthread\_join(id,NULL);

return 0;

}

**OUTPUT:**

****

**Ex 5.2: Thread Creation (threadc.c)**

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

char ch;

int count;

struct param

{ char ch;

int count; };

void \*printc(void \*parameter)

{ struct param \*p = (struct param\*) parameter;

int i;

for(i=0;i<p->count;++i)

{ fputc(p→ch,stderr); }

}

int main()

{ pthread\_t thread1\_id;

pthread\_t thread2\_id;

struct param thread1\_args;

struct param thread2\_args;

thread1\_args.ch = 'T';

thread2\_args.count = 3000;

pthread\_create(&thread1\_id,NULL,&printc,&thread1\_args);

thread2\_args.ch ='t';

thread2\_args.count = 2000;

pthread\_create(&thread2\_id,NULL,&printc,&thread2\_args);

pthread\_join(thread1\_id,NULL);

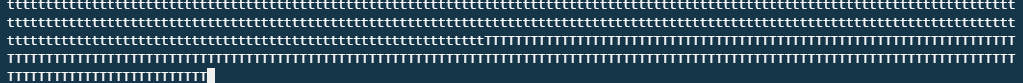
pthread\_join(thread2\_id,NULL);

return 0;

}

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

**OUTPUT**

****

**STATEMENT : WRITE A PROGRAM USING THREADS THAT PRINTS SUM OF NUMBERS UP TO GIVEN POSITIVE NUMBER.**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

int sum = 0;

void\* sum\_runner(void\* arg)

{ int \*limit\_ptr = (int\*)arg;

int limit = \*limit\_ptr;

int i;

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

sum += i;

}

int main(int argc, char \*\*argv)

{ if (argc < 2)

{ printf("Usage: %s <num>\n", argv[0]);

exit(-1); }

int limit = atoi(argv[1]);

pthread\_t tid;

pthread\_attr\_t attr;

pthread\_attr\_init(&attr);

pthread\_create(&tid, &attr, sum\_runner, &limit);

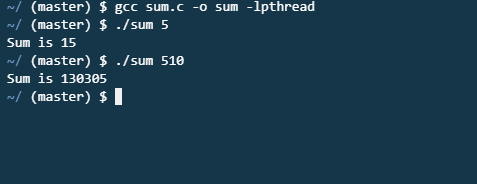
pthread\_join(tid, NULL);

printf("Sum is %d\n", sum);

return 0;

}

**OUTPUT:**

****