**4.6** Write a program using p-thread library of Linux. Create threethreads to take odd, even and prime respectively and print their average respectively.

**Objectives:**

1. To learn about threading in Linux/Unix and Java and difference between them..
2. Use of system call/library to write effective programs

**Theory:**

Historically, hardware vendors have implemented their own proprietary versions of threads. These implementations differed substantially from each other making it difficult for programmers to develop portable threaded applications.

* + In order to take full advantage of the capabilities provided by threads, a standardized programming interface was required.
* For UNIX systems, this interface has been specified by the IEEE POSIX 1003.1c standard (1995).
* Implementations adhering to this standard are referred to as POSIX threads, or Pthreads.
* Most hardware vendors now offer Pthreads in addition to their proprietary API's.
* The POSIX standard has continued to evolve and undergo revisions, including the Pthreads specification.

Pthreads defines a set of [**C**](https://en.wikipedia.org/wiki/C_(programming_language)) programming language [**types**,](https://en.wikipedia.org/wiki/Data_type) [**functions**](https://en.wikipedia.org/wiki/Function_(computer_science)) and constants. It is implemented with a pthread.h header and a thread [**library**.](https://en.wikipedia.org/wiki/Library) Pthread programs are compiled using gcc -pthread.

There are around 100 threads procedures, all prefixed ‘pthread\_’ and they can be categorized into four groups:

* Thread management - creating, joining threads etc.
* [Mutexes.](https://en.wikipedia.org/wiki/Mutex)
* [Condition variables.](https://en.wikipedia.org/wiki/Condition_variable)
* [Synchronization](https://en.wikipedia.org/wiki/Synchronization_(computer_science)) between threads using read/write locks and barriers.

On Linux, both fork() and pthreads use the same system call clone(), which creates a new process. The difference between them is simply the parameters they send to clone(), when creating a new thread, it simply makes both processes use the same memory mappings.

Pthreads are created using pthread\_create(). Pthreads terminate when the function returns, or the thread can call pthread\_exit() which terminates the calling thread explicitly. One Thread can wait on the termination of another by using pthread\_join(). The exit status is returned in status\_ptr.

* int **pthread\_create**(pthread\_t \*thread\_id,const pthread\_attr\_t \*attributes,void \*(\*thread\_function)(void \*), void \*arguments);

When an attribute object is not specified, it is NULL, and the *default* thread is created with the following attributes:

* + It is unbounded.
  + It is nondetached.
  + It has a a default stack and stack size.
  + It inhetits the parent's priority.
* Int **pthread\_exit** (void \*status);
* int **pthread\_join** (pthread\_t thread, void \*\*status\_ptr);

**Advantages:**

* **Light Weight**: When compared to the cost of creating and managing a process, a threadcan be created with much less operating system overhead. Managing threads requires fewer system resources than managing processes.
* **Efficient Communications/Data Exchange**: The primary motivation for considering theuse of Pthreads in a high performance computing environment is to achieve optimum performance. In particular, if an application is using MPI for on-node communications, there is a potential that performance could be improved by using Pthreads instead. For Pthreads there is no intermediate memory copy required because threads share the same address space within a single process. There is no data transfer, per se. It can be as efficient as simply passing a pointer.
* A perfect example is the typical web browser, where many interleaved tasks can be happening at the same time, and where tasks can vary in priority.

**Data Dictionary:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr Number | Variable/Function | Datatype | Use |
|  |  |  |  |
| 1 | t | int | Input variable. |
|  |  |  |  |
| 2 | tid | pthread\_t | Creating a thread in c. |
|  |  |  |  |
| 3 | st1 | void\* | Number 1 |
|  |  |  |  |
| 4 | st2 | void\* | Number 2 |
|  |  |  |  |
| 5 | st3 | void\* | Number 3 |
|  |  |  |  |
| 6 | st4 | void\* | Number 4 |
|  |  |  |  |
| 7 | avg | float | Average of above 4 numbers. |
|  |  |  |  |

**Program:**

#include <pthread.h>

#include <stdio.h>

void \*myThread()

{

int t;

scanf("%d",&t);

return (void \*)t;

}

int main()

{

pthread\_t tid;

void \*st1,\*st2,\*st3,\*st4;

printf("First number\n");

pthread\_create(&tid, NULL, myThread, NULL);

pthread\_join(tid, &st1);

printf("Second number\n");

pthread\_create(&tid, NULL, myThread, NULL);

pthread\_join(tid, &st2);

printf("Third number\n");

pthread\_create(&tid, NULL, myThread, NULL);

pthread\_join(tid, &st3);

printf("Fourth number\n");

pthread\_create(&tid, NULL, myThread, NULL); pthread\_join(tid, &st4);

float avg = ((int)st1+(int)st2+(int)st3+(int)st4)/4;

printf("%f\n",avg);

return 0;

}

**Output:**

$ ./a.out

First number

12

Second number

23

Third number

34

Fourth number

45

28.000000

$ ./a.out

First number

100

Second number

200

Third number

400

Fourth number

500

300.000000

**Conclusion:**

Use of p\_threads in C language to create and synchronize using semaphore can be done.

**References:**

https://www.cs.nmsu.edu/~jcook/Tools/pthreads/library.html