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.

* 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 and nondetached.
  + It has a a default stack and stack size.
  + It inhetits the parent's priority.
* Int pthread\_exit (void \*status);

**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 | t1 | pthread\_t | Create thread in c. |
|  |  |  |  |
| 2 | t2 | pthread\_t | Create thread in c. |
|  |  |  |  |
| 3 | t3 | pthread\_t | Create thread in c. |
|  |  |  |  |
| 4 | even | void \* | Function for thread t1. Prints even numbers. |
|  |  |  |  |
| 5 | odd | void \* | Function for thread t2. Prints odd numbers. |
|  |  |  |  |
| 6 | prime | void \* | Function for thread t3. Prints prime numbers. |
|  |  |  |  |

**Program:**

#include<stdio.h>

#include <stdlib.h>

#include <pthread.h>

void \*even(void \*mid)

{

int count = 0, sum = 0;

float avg = 0.0;

int\* id = (int\*)mid;

for(int i=1;i<=20;i++)

{

if(i%2==0)

{

count++;

sum = sum + i;

printf("Even %d: %d\n",count,i);

}

}

avg = (float)sum/count;

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

}

void \*odd(void \*mid)

{

int count = 0, sum = 0;

float avg = 0.0;

for(int i=1;i<=20;i++)

{

if(i%2!=0)

{

count++;

sum = sum + i;

printf("Odd %d: %d\n",count,i);

}

}

avg = (float)sum/count;

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

}

void \*prime(void \*mid)

{

int count = 1, sum = 2, c = 0;

float avg = 0.0;

printf("Prime %d: %d\n",count,2);

for(int i=3;i<=20;i++)

{

c = 0;

for(int j=2;j<i;j++)

{

if(i%j==0)

{

c++;

break;

}

}

if(c == 0)

{

count++;

sum = sum + i;

printf("Prime %d: %d\n",count,i);

}

}

avg = (float)sum/count;

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

}

int main()

{

pthread\_t t1,t2,t3;

// Let us create three threads

pthread\_create(&t1, NULL, &odd, NULL);

pthread\_create(&t2, NULL, &even, NULL);

pthread\_create(&t3, NULL, &prime, NULL);

pthread\_exit(NULL);

return 0;

}

**Output:**

Odd 1: 1

Odd 2: 3

Odd 3: 5

Odd 4: 7

Odd 5: 9

Odd 6: 11

Odd 7: 13

Odd 8: 15

Odd 9: 17

Odd 10: 19

Average is 10.000000

Prime 1: 2

Prime 2: 3

Prime 3: 5

Prime 4: 7

Prime 5: 11

Prime 6: 13

Prime 7: 17

Prime 8: 19

Average is 9.625000

Even 1: 2

Even 2: 4

Even 3: 6

Even 4: 8

Even 5: 10

Even 6: 12

Even 7: 14

Even 8: 16

Even 9: 18

Even 10: 20

Average is 11.000000

**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