Objectives

1. **Understanding the basics of threading**
2. **Create threads**

What Is a Thread?

Multiple strands of execution in a single program are called threads. A more precise definition is that a thread is a sequence of control within a process.

It's important to be clear about the difference between the fork system call and the creation of new threads. When a process executes a fork call, a new copy of the process is created with its own variables and its own PID. This new process is scheduled independently, and (in general) executes almost independently of the process that created it. When we create a new thread in a process, in contrast, the new thread of execution gets its own stack (and hence local variables) but shares global variables, file descriptors, signal handlers, and its current directory state with the process that created it.

Example of pthread Code – 1 to execute:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

void *thread_function(void *arg)
{
   sleep(10);
   printf("Printing HelloWorld from Thread \n");
   return NULL;
}

int main()
{
   pthread_t thread_id;
   printf("Before Thread\n");
   pthread_create(&thread_id, NULL, thread_function, NULL);
//create thread first parameter thread identification, then thread
// behavior, then function that
//execute in thread and in last parameter to function
```

```
   pthread_join(thread_id, NULL); //wait for thread
   printf("After Thread\n");
   exit(0);
}
```

For compiling code named sample.c: *Add* `-lpthread`

$gcc sample.c –o sample –lpthread

$./sample

Effect of Threading

Demo – 1

A code without threading

```
//#include <pthread.h>
#include<unistd.h>
#include<stdio.h>
void myturn() {
  while(1)
  {
   sleep(1);
   printf("My Turn\n");
  }
}
void yourturn() {
  while(1)
  {
   sleep(2);
   printf("Your Turn\n");
  }
}
int main()
{
myturn();
yourturn();
}
```

Demo – 2

A code without threading (without join)

```
#include <pthread.h>
#include<unistd.h>
#include<stdio.h>
void* myturn(void * arg) {
  while(1)
  {
   sleep(1);
   printf("My Turn\n");
  }
return NULL;
}

void yourturn() {
  while(1)
  {
   sleep(2);
   printf("Your Turn\n");
  }
}

int main()
{
pthread_t newthread;

pthread_create(&newthread, NULL, myturn, NULL);
//myturn();
yourturn();
}
```

Task 1:

Change the while loop to for loop, for both methods the iteration should be uneven. For example, myturn() should run for 10 times and yourtun() for 3 times.

Task 2:

See the output and write the reason of incomplete execution in manual. Now use the join for newthread, it is the solution where join will wait for the running thread to complete. Add the following LoC in after calling yourturn() in main method.

```
pthread_join(newthread, NULL);
```

Task 3:

Create a thread with following specifications:

- A method run as a thread
- Pass some integer value variable to thread method and printout the output.

HINT: You can refer to example code given in book page number 170. But remember, the code there is casting character to integer. We might need different type of casting here.

Task 4:

Execute the same above example with java threads, use both approaches of threading, inheritance and interfaces to implement.