C# Thread Synchronization

Synchronization is a technique that allows only one thread to access the resource for the particular time. No other thread can interrupt until the assigned thread finishes its task.

In multithreading program, threads are allowed to access any resource for the required execution time. Threads share resources and executes asynchronously. Accessing shared resources (data) is critical task that sometimes may halt the system. We deal with it by making threads synchronized.

It is mainly used in case of transactions like deposit, withdraw etc.

Advantage of Thread Synchronization

* Consistency Maintain
* No Thread Interference

C# Lock

We can use C# **lock keyword** to execute program synchronously. It is used to get lock for the current thread, execute the task and then release the lock. It ensures that other thread does not interrupt the execution until the execution finish.

Here, we are creating two examples that executes asynchronously and synchronously.

C# Example: Without Synchronization

class Printer

{

public void PrintTable1()

{

{

for (int i = 1; i <= 10; i++)

{

Console.Write(i + ",");

Thread.Sleep(1000);

}

}

}

class Program

{

static void Main(string[] args)

{

Printer p = new Printer();

Thread t1 = new Thread(p.PrintTable1);

Thread t2 = new Thread(p.PrintTable1);

// Thread t1 = new Thread(new ThreadStart(p.PrintTable1));

// Thread t2 = new Thread(new ThreadStart(p.PrintTable1));

t1.Start();

t2.Start();

Console.ReadLine();

}

}

}

C# Example: Synchronization using Lock

class Printer

{

public void PrintTable1()

{

lock (this)

{

for (int i = 1; i <= 10; i++)

{

Console.Write(i + ",");

Thread.Sleep(1000);

}

}

}

}

class Program

{

static void Main(string[] args)

{

Printer p = new Printer();

Thread t1 = new Thread(p.PrintTable1);

Thread t2 = new Thread(p.PrintTable1);

// Thread t1 = new Thread(new ThreadStart(p.PrintTable1));

// Thread t2 = new Thread(new ThreadStart(p.PrintTable1));

t1.Start();

t2.Start();

Console.ReadLine();

}

}

}

Another example With Array :

static void Main(string[] args)

{

Console.WriteLine(" \*\*\*\*\*\*\*\*\*\*\* Multiple Threads \*\*\*\*\*\*");

Printer p = new Printer();

Thread[] Threads = new Thread[3];

for (int i = 0; i < 3; i++)

{

Threads[i] = new Thread(p.PrintTable1);

Threads[i].Name = "Child" + i;

}

foreach (Thread t in Threads)

{

t.Start();

}

Console.ReadLine();

}