# Case Study 2

Operation Systems

section 53

สมาชิกในกลุ่ม 14


64015019 นายจารุพัฒน์ เคนพรม

64015051 นายเตชินท์ ไม้ทอง

64015102 นายพิสิฐพงศ์ พิสิฐแก้วเพชร

64015112 นายภูมิพัฒน์ ลาลุน

64015163 นายอภิสิทธิ์ชัย ทองโต

64015172 นายเอกรินทร์ องอาจ

# Result (Original Version)

```
j=1, thread:1
j=2, thread:1
j=13, thread:1
j=14, thread:1
j=25, thread:1
j=36, thread:1
j=37, thread:1
j=48, thread:1
j=49, thread:1
j=50, thread:1
j=41, thread:1
j=42, thread:1
j=43, thread:1
j=44, thread:1
j=45, thread:1
j=46, thread:1
j=47, thread:1
j=48, thread:1
j=49, thread:1
j=50, thread:1
j=41, thread:1
j=42, thread:1
j=43, thread:1
j=44, thread:1
```

```
j=46, thread:1
j=47, thread:1
j=48, thread:1
j=49, thread:1
j=50, thread:1
j=41, thread:1
j=42, thread:1
j=43, thread:1
j=44, thread:1
j=45, thread:1
j=46, thread:1
j=47, thread:1
j=48, thread:1
j=49, thread:1
j=50, thread:1
j=41, thread:1
j=42, thread:1
j=43, thread:1
j=44, thread:1
j=45, thread:1
j=46, thread:1
j=47, thread:1
j=48, thread:1
```

# Version 1

## สิ่งที่คาดว่าเป็นปัญหา

1.ตัวเลขรันไม่เรียงกันตามลำดับ

2.ตัวเลขรันกระโดดข้ามกันไปมา

สาเหตุ : Thread แต่ละตัวมีการทำงานพร้อมกัน ทำให้เกิดการSharing Memory ซึ่งทำให้เกิดการดึงข้อมูลซ้ำกันและค่าเกิดความผิดเพี้ยน

```
j=1, thread:1
j=100, thread:2
j=0, thread:3
j=0, thread:2
j=3, thread:1
j=102, thread:3
j=14, thread:1
j=108, thread:3
j=114, thread:2
j=21, thread:2
j=120, thread:3
j=22, thread:1
j=126, thread:1
j=28, thread:3
j=23, thread:2
j=29, thread:1
j=35, thread:3
j=134, thread:2
j=140, thread:1
j=135, thread:3
j=42, thread:2
j=146, thread:2
j=147, thread:3
j=48, thread:1
j=143, thread:3
j=148, thread:1
j=149, thread:2
j=50, thread:2
j=50, thread:1
j=50, thread:3
j=47, thread:1
j=47, thread:2
j=47, thread:3
```

# Version 1 (using lock)

```
2 references
static void EnQueue(int eq)
{
    TSBuffer[Back] = eq;
    Back++;
    Back %= 10;
    Count += 1;
}

1 reference
static void th02(object t)
{
    int i;
    int j;

    for (i=0; i< 60; i++)
    {
        j = DeQueue();
        Console.WriteLine("j={0}, thread:{1}", j, t);
        Thread.Sleep(100);
    }
}
```

```
private static int n = 0;
2 references
private static object _Lock = new object();

2 references
static void EnQueue(int eq)
{
    lock (_Lock)
    {

        TSBuffer[Back] = eq;
        Back++;
        Back %= 10;
        Count += 1;
        //Console.WriteLine(eq);

    }
}

3 references
static void th02(object t)
{

    int j;

    for (int i = 0; i < 60; i++)
    {
        lock (_Lock)
        {

            j = DeQueue();
            Console.WriteLine("j={0}, thread:{1}", j, t);
            Thread.Sleep(100);

        }

    }
}
```

# Result (Version 1)

```
j=1, thread:1
j=100, thread:2
j=0, thread:3
j=101, thread:1
j=0, thread:2
j=0, thread:3
j=0, thread:3
j=0, thread:2
j=0, thread:1
j=0, thread:3
j=1, thread:2
j=100, thread:1
j=2, thread:3
j=101, thread:2
j=102, thread:1
j=3, thread:3
j=103, thread:2
j=4, thread:1
j=5, thread:3
j=104, thread:2
j=6, thread:1
j=105, thread:3
j=7, thread:2
j=106, thread:1
j=8, thread:3
j=107, thread:2
j=108, thread:1
j=9, thread:1
j=109, thread:3
```

```
j=30, thread:2
j=130, thread:1
j=31, thread:3
j=131, thread:2
j=32, thread:1
j=132, thread:3
j=33, thread:2
j=133, thread:1
j=34, thread:3
j=134, thread:2
j=35, thread:1
j=135, thread:3
j=36, thread:2
j=136, thread:1
j=137, thread:3
j=37, thread:2
j=33, thread:1
j=38, thread:3
j=34, thread:2
j=134, thread:1
j=35, thread:3
j=135, thread:2
j=36, thread:1
j=136, thread:3
j=137, thread:2
j=37, thread:1
j=138, thread:3
j=38, thread:2
j=139, thread:1
```

# Version 2

## สิ่งที่คาดว่าเป็นปัญหา

1.ตัวเลขรันไม่เรียงกันตามลำดับ

2.ตัวเลขรันกระโดดข้ามกันไปมา

สาเหตุ :

1.มากจาก lock ที่ใช้มันไปrelease thread อื่น

2.Lock ปกติไม่สามารถสร้างจังหวะการทำงานได้

3.มีการเข้าถึง shared data ตัวอื่นที่ไม่ lockไว้

# Version 2 (using lock design pattern)

Monitor.wait(object obj);

Releases the lock on an object and blocks the current thread until it reacquires the lock(waiting).

Monitor.PulseAll(object obj);

Notifies all waiting threads of a change in the object's state(Ready).

# Version 2 (using lock design pattern)

```
private static int n = 0;
2 references
private static object _Lock = new object();

2 references
static void EnQueue(int eq)
{
    lock (_Lock)
    {

        TSBuffer[Back] = eq;
        Back++;
        Back %= 10;
        Count += 1;
        //Console.WriteLine(eq);

    }
}
```

```
3 references
static void th02(object t)
{

    int j;

    for (int i = 0; i < 60; i++)
    {
        lock (_Lock)
        {

            j = DeQueue();
            Console.WriteLine("j={0}, thread:{1}", j, t);
            Thread.Sleep(100);

        }

    }
}
```

```
2 references
static void EnQueue(int eq)
{
    lock (_Lock)
    {
        while (Count >= 10)
        {
            Monitor.Wait(_Lock);
            //Console.WriteLine("Queue is full");
        }
        TSBuffer[Back] = eq;
        Back++;
        Back %= 10;
        Count += 1;
        //Console.WriteLine(eq);
        if (Count <= 1)
            Monitor.PulseAll(_Lock);
    }
}
```

```
static void th02(object t)
{

    int j;

    for (int i = 0; i < 60; i++)
    {
        lock (_Lock)
        {
            while (Count < 1)
            {
                Monitor.Wait(_Lock);
                //Console.WriteLine("Queue is empty");
            }
            j = DeQueue();
            Console.WriteLine("j={0}, thread:{1}", j, t);
            Thread.Sleep(100);
            if (Count == 10)
                Monitor.PulseAll(_Lock);
        }
        if(n >=101)
        {
            Thread.Sleep(100);
            System.Environment.Exit(0);
        }
    }
}
```

# Result (Version 2)