รายงาน

Case Study 2

เสนอ

ผศ.วัจนพงศ์ เกษมศิริ

รายชื่อสมาชิก

1. 63010022 นายกฤต รุ่งโรจน์กิจกุล
2. 63010040 นายกฤษณะพัฒน์ พันธ์เจริญ
3. 63010052 นายก้องเกียรติ ชุนงาม
4. 63010095 นายเกื้อกูล นิยมสิทธิ์
5. 63010187 นางสาวชนัญชิดา ศรีทองดี
6. 63010872 นางสาววัชราภรณ์ ชาแท่น

รายงานนี้เป็นส่วนหนึ่งของรายวิชา 01076011

OPERATING SYSTEMS

Original code

Import library และ ประกาศตัวแปร:

```
using System;
using System.Threading;

namespace OS_Problem_02
{
    0 references
    class Thread_safe_buffer
    {
        2 references
        static int[] TSBuffer = new int[10];
        3 references
        static int Front = 0;
        3 references
        static int Back = 0;
        2 references
        static int Count = 0;
```

Function Enqueue and Dequeue:

```
        2 references
        static void EnQueue(int eq)
        {
            TSBuffer[Back] = eq;
            Back++;
            Back %= 10;
            Count += 1;
        }
        static int DeQueue()
        {
            int x = 0;
            x = TSBuffer[Front];
            Front++;
            Front %= 10;
            Count -= 1;
            return x;
        }

        1 reference
        static void th01()
        {
            int i;

            for (i = 1; i < 51; i++)
            {
                EnQueue(i);
                Thread.Sleep(5);
            }
        }
```

Thread Function:

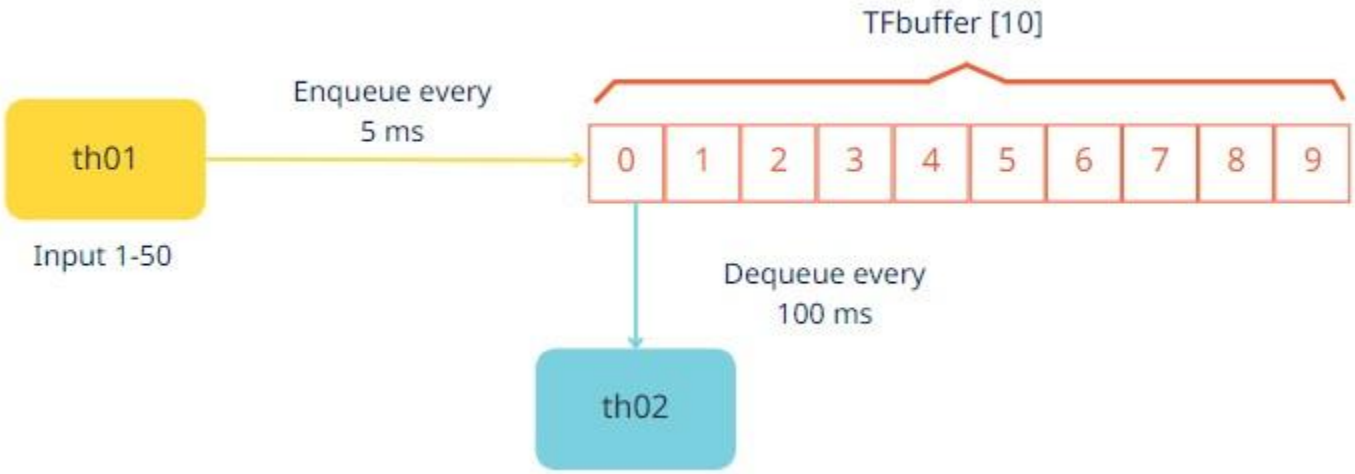
```
static void th01()
{
    int i;

    for (i = 1; i < 51; i++)
    {
        EnQueue(i);
        Thread.Sleep(5);
    }
}
```

```
static void th011()
{
    int i;

    for (i = 100; i < 151; i++)
    {
        EnQueue(i);
        Thread.Sleep(5);
    }
}


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);
    }
}
```

Main:

```
static void Main(string[] args)
{
    Thread t1 = new Thread(th01);
    //Thread t11 = new Thread(th011);
    Thread t2 = new Thread(th02);
    //Thread t21 = new Thread(th02);
    //Thread t22 = new Thread(th02);

    t1.Start();
    //t11.Start();
    t2.Start(1);
    //t21.Start(2);
    //t22.Start(3);
}
```

Overview การทำงาน



ข้อสังเกต

ผลลัพธ์มีค่าที่โดดไปมา ไม่ได้เรียงจาก 1 ถึง 50 อย่างที่ควรจะเป็น และค่าของบางส่วนหายไป

สมมติฐาน

1. เกิดการ enqueue ทั้ง ๆ ที่ buffer เต็ม ทำให้ข้อมูลที่ป้อนเข้า buffer บางตัวหายไปจากการถูกข้อมูลที่ถูกป้อนเข้าคิวใหม่เขียนทับทั้ง ๆ ที่มันยังไม่ถูก dequeue ออกมาแสดงออกทาง console ทำให้เมื่อ dequeue ออกมาแสดง ค่ามีการกระโดด
2. การ enqueue ซึ่งทำโดยThread 1 ทำงานเร็วกว่า การ dequeue ใน Thread 2 จึงทำให้เกิดการทำงานที่ไม่สอดคล้องกัน

Version 1

Import Library และประกาศตัวแปร:

```
using System;
using System.Threading;

namespace OS_Problem_02
{
    0 references
    class Thread_safe_buffer
    {
        2 references
        static int[] TSBuffer = new int[10];
        3 references
        static int Front = 0;
        3 references
        static int Back = 0;
        5 references
        static int Count = 0;
        3 references
        static bool en_ch = false;
```

Function Enqueue and Dequeue:

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

1 reference
static int DeQueue()
{
    int x = 0;
    x = TSBuffer[Front];
    Front++;
    Front %= 10;
    Count -= 1;
    return x;
}
```

Thread Function:

```
1 reference
static void th01()
{
    int i;

    for (i = 1; i < 51; i++)
    {
        while (Count == 10);
        EnQueue(i);
        Thread.Sleep(5);
    }
    en_ch = true;
}
```

```
0 references
static void th011()
{
    int i;

    for (i = 100; i < 151; i++)
    {
        EnQueue(i);
        Thread.Sleep(5);
    }
}
```

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

    for (i = 0; i < 60; i++)
    {
        while (Count == 0 && en_ch == false);
        if (Count == 0 && en_ch == true)
            break;
        j = DeQueue();
        Console.WriteLine("j={0}, thread:{1}", j, t);
        Thread.Sleep(100);
    }
}
```

Main:

```
static void Main(string[] args)
{
    Thread t1 = new Thread(th01);
    //Thread t11 = new Thread(th011);
    Thread t2 = new Thread(th02);
    //Thread t21 = new Thread(th02);
    //Thread t22 = new Thread(th02);

    t1.Start();
    //t11.Start();
    t2.Start(1);
    //t21.Start(2);
    //t22.Start(3);
}
```

ส่วนที่แก้ไข

เพิ่มตัวแปรใหม่ และเพิ่มเงื่อนไขที่ th01 กับ th02 เพื่อใช้ควบคุมการ enqueue และ dequeue

```
using System;
using System.Threading;

namespace OS_Problem_02
{
    0 references
    class Thread_safe_buffer
    {
        2 references
        static int[] TSBuffer = new int[10];
        3 references
        static int Front = 0;
        3 references
        static int Back = 0;
        5 references
        static int Count = 0;
        3 references
        static bool en_ch = false;
```

```
static void th01()
{
    int i;

    for (i = 1; i < 51; i++)
    {
        while (Count == 10);
        EnQueue(i);
        Thread.Sleep(5);
    }
    en_ch = true;
}
```

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

    for (i = 0; i < 60; i++)
    {
        while (Count == 0 && en_ch == false);
        if (Count == 0 && en_ch == true)
            break;
        j = DeQueue();
        Console.WriteLine("j={0}, thread:{1}", j, t);
        Thread.Sleep(100);
    }
}
```

ข้อสังเกต

ผลลัพธ์มีค่าที่เรียงลำดับจาก 1-50 ได้อย่างถูกต้องและครบถ้วน

สมมติฐาน

มีการคุมการทำงานของ th01 และ th02 ด้วยตัวแปร Count และ en_ch โดยตัวแปร Count เป็นตัวแปรนับจำนวนข้อมูลที่อยู่ใน buffer ทำให้ไม่เกิดการ enqueue ที่ทำให้ buffer เกินจำนวน และการ dequeue ใน buffer ที่ว่างเปล่า และ en_ch เอาไว้เช็คว่ามีการ enqueue ครบ 50 ครั้งแล้วหรือไม่ ทำให้ไม่เกิดการ dequeue เกินจำนวนครั้งที่ enqueue ไป

Version 2

Import Library และประกาศตัวแปร:

```
using System;
using System.Threading;

namespace OS_Problem_02
{
    0 references
    class Thread_safe_buffer
    {
        2 references
        static int[] TSBuffer = new int[10];
        3 references
        static int Front = 0;
        3 references
        static int Back = 0;
        6 references
        static int Count = 0;
        3 references
        static bool en_ch = false;
        3 references
        static bool en_ch11 = false;
```

Enqueue and Dequeue Function:

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

1 reference
static int DeQueue()
{
    int x = 0;
    x = TSBuffer[Front];
    Front++;
    Front %= 10;
    Count -= 1;
    return x;
}
```

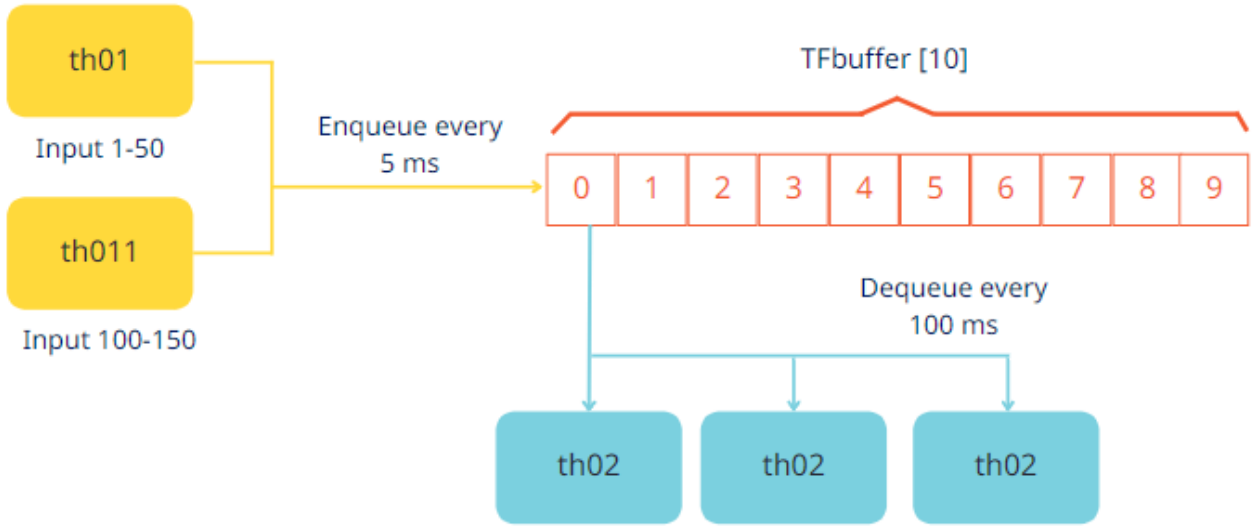
Main:

```
static void Main(string[] args)
{
    Thread t1 = new Thread(th01);
    Thread t11 = new Thread(th011);
    Thread t2 = new Thread(th02);
    Thread t21 = new Thread(th02);
    Thread t22 = new Thread(th02);

    t1.Start();
    t11.Start();
    t2.Start(1);
    t21.Start(2);
    t22.Start(3);
}
```

Thread Function:

```
1 reference
static void th01()
{
    int i;

    for (i = 1; i < 51; i++)
    {
        while (Count == 10);
        EnQueue(i);
        Thread.Sleep(5);
    }
    en_ch = true;
}

1 reference
static void th011()
{
    int i;

    for (i = 100; i < 151; i++)
    {
        while (Count == 10) ;
        EnQueue(i);
        Thread.Sleep(5);
    }
    en_ch11 = true;
}

3 references
static void th02(object t)
{
    int i;
    int j;

    for (i = 0; i < 60; i++)
    {
        while (Count == 0 && (en_ch == false || en_ch11 == false));
        if (Count == 0 && en_ch == true && en_ch11 == true)
            break;
        j = DeQueue();
        Console.WriteLine("j={0}, thread:{1}", j, t);
        Thread.Sleep(100);
    }
}
```

Overview การทำงาน



ส่วนที่แก้ไข

เพิ่มตัวแปรสำหรับ th011, ปรับเงื่อนไขใน th011 และ th02, เปิดใช้ Thread ทั้งหมดใน Main

```
static void Main(string[] args)
{
    Thread t1 = new Thread(th01);
    Thread t11 = new Thread(th011);
    Thread t2 = new Thread(th02);
    Thread t21 = new Thread(th02);
    Thread t22 = new Thread(th02);

    t1.Start();
    t11.Start();
    t2.Start(1);
    t21.Start(2);
    t22.Start(3);
}
```

```
static void th011()
{
    int i;

    for (i = 100; i < 151; i++)
    {
        while (Count == 10) ;
        EnQueue(i);
        Thread.Sleep(5);
    }
    en_ch11 = true;
}

3 references
static void th02(object t)
{
    int i;
    int j;

    for (i = 0; i < 60; i++)
    {
        while (Count == 0 && (en_ch == false || en_ch11 == false));
        if (Count == 0 && en_ch == true && en_ch11 == true)
            break;
        j = DeQueue();
        Console.WriteLine("j={0}, thread:{1}", j, t);
        Thread.Sleep(100);
    }
}
```

ข้อสังเกต

ผลลัพธ์มีค่าที่โดดไปมาอย่างไม่มีรูปแบบ ซึ่งมีการแบ่งกัน dequeue ด้วย 3 Thread

สมมติฐาน

เนื่องจากมีการเพิ่ม Thread แต่ยังไม่มี Thread safety จึงทำให้เกิด Race condition ส่งผลให้ผลลัพธ์ที่ได้มีค่าที่โดดไปมาอย่างไม่มีรูปแบบ

Version 3

Import Library และประกาศตัวแปร:

```
using System;
using System.Threading;

namespace OS_Problem_02
{
    0 references
    class Thread_safe_buffer
    {
        2 references
        static int[] TSBuffer = new int[10];
        3 references
        static int Front = 0;
        3 references
        static int Back = 0;
        6 references
        static int Count = 0;
        3 references
        static bool en_ch = false;
        3 references
        static bool en_ch11 = false;
        3 references
        static object _Lock = new object();
```

Function Enqueue and Dequeue :

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

1 reference
static int DeQueue()
{
    int x = 0;
    x = TSBuffer[Front];
    Front++;
    Front %= 10;
    Count -= 1;
    return x;
}
```

Thread Function:

```
1 reference
static void th01()
{
    int i;
    lock (_Lock)
    {
        for (i = 1; i < 51; i++)
        {
            while (Count == 10) ;
            EnQueue(i);
            Thread.Sleep(5);
        }
    }
    en_ch = true;
}

1 reference
static void th011()
{
    int i;
    lock (_Lock)
    {
        for (i = 100; i < 151; i++)
        {
            while (Count == 10) ;
            EnQueue(i);
            Thread.Sleep(5);
        }
    }
    en_ch11 = true;
}
```

```
3 references
static void th02(object t)
{
    int i;
    int j;
    lock (_Lock)
    {
        for (i = 0; i < 60; i++)
        {
            while (Count == 0 && (en_ch == false || en_ch11 == false)) ;
            if (Count == 0 && en_ch == true && en_ch11 == true)
                break;
            j = DeQueue();
            Console.WriteLine("j={0}, thread:{1}", j, t);
            Thread.Sleep(100);
        }
    }
}
```

Main:

```
static void Main(string[] args)
{
    Thread t1 = new Thread(th01);
    Thread t11 = new Thread(th011);
    Thread t2 = new Thread(th02);
    Thread t21 = new Thread(th02);
    Thread t22 = new Thread(th02);

    t1.Start();
    t11.Start();
    t2.Start(1);
    t21.Start(2);
    t22.Start(3);
}
```

ส่วนที่แก้ไข

เพิ่ม object _Lock และเพิ่ม lock เข้าไปในทุกๆ Thread

```
1 reference
static void th01()
{
    int i;
    lock (_Lock)
    {
        for (i = 1; i < 51; i++)
        {
            while (Count == 10) ;
            EnQueue(i);
            Thread.Sleep(5);
        }
    }
    en_ch = true;
}

1 reference
static void th011()
{
    int i;
    lock (_Lock)
    {
        for (i = 100; i < 151; i++)
        {
            while (Count == 10) ;
            EnQueue(i);
            Thread.Sleep(5);
        }
    }
    en_ch11 = true;
}
```

```
using System;
using System.Threading;

namespace OS_Problem_02
{
    0 references
    class Thread_safe_buffer
    {
        2 references
        static int[] TSBuffer = new int[10];
        3 references
        static int Front = 0;
        3 references
        static int Back = 0;
        6 references
        static int Count = 0;
        3 references
        static bool en_ch = false;
        3 references
        static bool en_ch11 = false;
        3 references
        static object _Lock = new object();
```

```
3 references
static void th02(object t)
{
    int i;
    int j;
    lock (_Lock)
    {
        for (i = 0; i < 60; i++)
        {
            while (Count == 0 && (en_ch == false || en_ch11 == false)) ;
            if (Count == 0 && en_ch == true && en_ch11 == true)
                break;
            j = DeQueue();
            Console.WriteLine("j={0}, thread:{1}", j, t);
            Thread.Sleep(100);
        }
    }
}
```

ข้อสังเกต

โปรแกรมไม่แสดงค่าใดๆ ออกมาเลย

สมมติฐาน

lock ของ th01, th011 และ th02 ใช้ key ตัวเดียวกัน ทำให้ต้องทำงานจนจบทีละ 1 Thread ซึ่งไม่สามารถทำได้เพราะมีเงื่อนไข while (Count == 10) ใน th01 และ th011 และ while (Count == 0 && ...) ใน th02 ส่งผลให้เกิด infinite loop

Version 4

Import Library และประกาศตัวแปร:

```
0 references
class Thread_safe_buffer
{
    2 references
    static int[] TSBuffer = new int[10];
    3 references
    static int Front = 0;
    3 references
    static int Back = 0;
    6 references
    static int Count = 0;
    3 references
    static bool en_ch = false;
    3 references
    static bool en_ch11 = false;
    2 references
    static object _Lock_en = new object();
    1 reference
    static object _Lock_de = new object();
```

Enqueue and Dequeue Function:

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

1 reference
static int DeQueue()
{
    int x = 0;
    x = TSBuffer[Front];
    Front++;
    Front %= 10;
    Count -= 1;
    return x;
}
```

Thread Function:

```
1 reference
static void th01()
{
    int i;
    lock (_Lock_en)
    {
        for (i = 1; i < 51; i++)
        {
            while (Count == 10) ;
            EnQueue(i);
            Thread.Sleep(5);
        }
    }
    en_ch = true;
}
```

```
1 reference
static void th011()
{
    int i;
    lock (_Lock_en)
    {
        for (i = 100; i < 151; i++)
        {
            while (Count == 10) ;
            EnQueue(i);
            Thread.Sleep(5);
        }
    }
    en_ch11 = true;
}
```

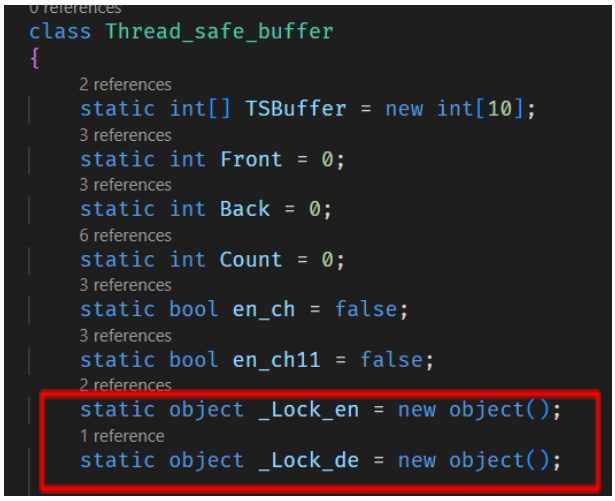
```
3 references
static void th02(object t)
{
    int i;
    int j;
    lock (_Lock_de)
    {
        for (i = 0; i < 60; i++)
        {
            while (Count == 0 && (en_ch == false || en_ch11 == false)) ;
            if (Count == 0 && en_ch == true && en_ch11 == true)
                break;
            j = DeQueue();
            Console.WriteLine("j={0}, thread:{1}", j, t);
            Thread.Sleep(100);
        }
    }
}
```

Main:

```
static void Main(string[] args)
{
    Thread t1 = new Thread(th01);
    Thread t11 = new Thread(th011);
    Thread t2 = new Thread(th02);
    Thread t21 = new Thread(th02);
    Thread t22 = new Thread(th02);

    t1.Start();
    t11.Start();
    t2.Start(1);
    t21.Start(2);
    t22.Start(3);
}
```

ส่วนที่แก้ไข

แยก _Lock ออกเป็น _Lock_en และ _Lock_de เพื่อใช้สำหรับการ enqueue และ dequeue ตามลำดับ



ข้อสังเกต

ผลลัพธ์มีค่าที่ถูกต้องและครบถ้วน แต่การแบ่งการทำงานของ Thread ยังไม่ถูกต้อง คือ Thread ที่ทำการ dequeue ทำงานแค่ 2 Thread โดย Thread แรกจะทำงานต่อเนื่อง 60 ครั้งแรก และ Thread ต่อมาจะทำงานอีก 40 ครั้ง

สมมติฐาน

lock ของ th02 วางไว้ผิดที่ ทำให้ Thread ที่ได้รับสิทธิ์ไป ต้องทำงานจนครบรอบลูปก่อน จึงจะเปลี่ยนสิทธิ์ไปให้ Thread อื่นๆ ได้ ดังนั้นจากการ enqueue 100 ครั้ง ก็จะมีการ dequeue 100 ครั้ง ส่งผลให้การ dequeue มีการทำงานเพียง 2 Thread จากทั้งหมด 3 Thread คือ Thread แรก 60 ครั้ง และ Thread ที่สองอีก 40 ครั้ง

Final Version

Import Library และประกาศตัวแปร:

```
using System;
using System.Threading;

namespace OS_Problem_02
{
    0 references
    class Thread_safe_buffer
    {
        2 references
        static int[] TSBuffer = new int[10];
        3 references
        static int Front = 0;
        3 references
        static int Back = 0;
        6 references
        static int Count = 0;
        3 references
        static bool en_ch = false;
        3 references
        static bool en_ch11 = false;
        2 references
        static object _Lock_en = new object();
        1 reference
        static object _Lock_de = new object();
```

Function Enqueue and Dequeue:

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

1 reference
static int DeQueue()
{
    int x = 0;
    x = TSBuffer[Front];
    Front++;
    Front %= 10;
    Count -= 1;
    return x;
}
```

Thread Function:

```
1 reference
static void th01()
{
    int i;
    lock (_Lock_en)
    {
        for (i = 1; i < 51; i++)
        {
            while (Count == 10) ;
            EnQueue(i);
            Thread.Sleep(5);
        }
    }
    en_ch = true;
}

1 reference
static void th011()
{
    int i;
    lock (_Lock_en)
    {
        for (i = 100; i < 151; i++)
        {
            while (Count == 10) ;
            EnQueue(i);
            Thread.Sleep(5);
        }
    }
    en_ch11 = true;
}
```

```
3 references
static void th02(object t)
{
    int i;
    int j;
    for (i = 0; i < 60; i++)
    {
        lock (_Lock_de)
        {
            while (Count == 0 && (en_ch == false || en_ch11 == false)) ;
            if (Count == 0 && en_ch == true && en_ch11 == true)
                break;
            j = DeQueue();
            Console.WriteLine("j={0}, thread:{1}", j, t);
            Thread.Sleep(100);
        }
    }
}
```

Main:

```
static void Main(string[] args)
{
    Thread t1 = new Thread(th01);
    Thread t11 = new Thread(th011);
    Thread t2 = new Thread(th02);
    Thread t21 = new Thread(th02);
    Thread t22 = new Thread(th02);

    t1.Start();
    t11.Start();
    t2.Start(1);
    t21.Start(2);
    t22.Start(3);
}
```

ส่วนที่แก้ไข

ย้าย lock ของ th02 ไปไว้ใน for loop แทน

```
3 references
static void th02(object t)
{
    int i;
    int j;
    lock (_Lock_de)
    {
        for (i = 0; i < 60; i++)
        {
            while (Count == 0 && (en_ch == false || en_ch11 == false)) ;
            if (Count == 0 && en_ch == true && en_ch11 == true)
                break;
            j = DeQueue();
            Console.WriteLine("j={0}, thread:{1}", j, t);
            Thread.Sleep(100);
        }
    }
}
```

Version 4

```
3 references
static void th02(object t)
{
    int i;
    int j;
    for (i = 0; i < 60; i++)
    {
        lock (_Lock_de)
        {
            while (Count == 0 && (en_ch == false || en_ch11 == false)) ;
            if (Count == 0 && en_ch == true && en_ch11 == true)
                break;
            j = DeQueue();
            Console.WriteLine("j={0}, thread:{1}", j, t);
            Thread.Sleep(100);
        }
    }
}
```

Version 5

สรุปผลการทดลอง

จากโค้ดต้นแบบนั้น การทำงานของ Queue ยังมีขั้นตอนการทำงานที่ไม่ถูกต้อง จึงมีผลทำให้ข้อมูลบางส่วนหายไป ดังนั้น จึงต้องทำการแก้ไขให้ Enqueue และ Dequeue สามารถทำงานได้อย่างถูกต้อง นั่นก็คือการกำหนดเงื่อนไขให้เมื่อ Queue นั้นเต็ม จะไม่สามารถทำการ Enqueue ได้ และหากใน Queue นั้นไม่มีข้อมูลอยู่ จะไม่สามารถทำการ Dequeue ได้ และเนื่องจากเป็นการทำงานแบบ multithread จึงทำให้เกิดการ race condition ดังนั้นจึงต้องทำ thread safe เพื่อแก้ปัญหา race condition โดยการเพิ่ม lock เพื่อให้การทำงานของ thread มีความเป็นระบบมากขึ้น ผลลัพธ์จึงออกมาเรียงกัน ไม่ซ้ำกัน ไม่มีข้อมูลไหนขาดหายไป และทำการสร้าง object lock สำหรับแต่ละ Process เพื่อให้แต่ละ Process สามารถทำงานพร้อมกันได้

การทำงานดังข้างต้นนั้นคือการทำงานแบบ Synchronization คือการที่ process ต่างๆ มีความเกี่ยวข้องกันทั้งในด้านของการทำงานที่ต้องใช้ทรัพยากรในระบบร่วมกันหรือการทำงานเป็นลำดับขั้นต่อกัน ช่วยให้การทำงานในรูปแบบ multithreading ทำงานได้ถูกต้อง และไม่เกิด race condition ระหว่างกัน