Synchronization Part II

1. Discussion

Ex-00 , Ex-01 , Ex-02 , Ex03

- คาดว่าผลลัพธ์ของแต่ละโปรแกรม จะออกมาเป็นแบบใด
- ความเร็วในการทำงานคาดว่าโปรแกรมใดจะเร็วที่สุดเพราะเหตุใด
- ทำการ run แต่ล่ะโปรแกรมเพื่อทดสอบสมมติฐานที่ได้อภิปราย
- เปรียบเทียบผลที่คาดการณ์กับผลที่ run ได้
- หาเหตผุลสนับสนุน
- สรุป

64010761 :

ผลลัพธ์ที่คาดการณ์ : คาดว่าผลลัพธ์ของโปรแกรมที่ run ออกมาแล้วได้ค่า sum ที่ถูกต้องคือ EX00, EX02, EX03 ส่วนค่า sum ของ EX01 นั้นไม่ถูกต้อง

ระยะเวลาในการทำงานที่คาดการณ์ : EX02 > EX00 > EX03 ส่วน EX01 ค่าที่ได้ไม่ถูกต้อง จึงไม่นับรวม

ผลลัพธ์จากการ run :

Ex00 :

```
Start...
sum = 1000000
Time used: 5ms
```

Ex01 :

```
Start...
sum = 2058193861
Time used: 2ms
```

Ex02 :

```
Start...
sum = 1000000
Time used: 81ms
```

Ex03 :

```
Start...
sum = 1000000
Time used: 8ms
```

เปรียบเทียบผลลัพธ์ : จากผลลัพธ์พบว่า Ex02 ใช้เวลาในการทำงานมากที่สุดจริง รองลงมาคือ Ex03 และใช้เวลาน้อยที่สุดคือ Ex00 ซึ่งไม่ตรงกับที่คาดการณ์ไว้

เหตุผลสนับสนุน : เนื่องจาก Ex02 มีการ lock ค่าทุกครั้งที่ทำการวนลูปในแต่ละรอบทำให้เกิด delay เป็นจำนวนมาก และจากการทดลองจะเห็นได้ว่า Ex00 กับ Ex03 มีเวลาห่างกันเพียง 3 ms ซึ่งใกล้เคียงกันมาก จากโปรแกรมใน Ex03 ที่มาการ lock ทั้ง loop ที่ต้องรอให้ Thread นึงทำเสร็จก่อนอีก Thread นึงจึงสามารถทำต่อได้ ซึ่งมีค่าไม่ต่างกับ Ex00 ที่ทำงาน Thread เดียว

สรุป : จากระยะเวลาในการทำงาน และผลลัพธ์จากการ run จะเห็นได้ว่า Ex01 ใช้ระยะเวลาในการทำงานน้อยที่สุด แต่ผลลัพธ์ไม่ถูกต้อง เพราะ EX01 ใช้เวลาการทำงานที่เร็วที่สุดเนื่องจากมีการแบ่ง Thread ในการทำงาน และมีการ lock ค่าเฉพาะตอนที่ update ค่าทำให้ฟังก์ชัน plus กับ minus ทำงานพร้อมกันได้ และมี update ค่าที่ถูกต้อง ดังนั้นระยะเวลาในการทำงาน EX02 จะมากที่สุด ตามด้วย EX03 และสุดท้าย EX00 (เรียงจากการใช้เวลามากสุดไปน้อยสุด ไม่นับรวม EX01)

64010845 :

Discussion

1. คาดว่าโปรแกรมที่ EX00 EX02 และ EX 03 ผลลัพธ์ถูกโดยความเร็ว EX02 < EX03 < EX00 (เรียงเวลาจากน้อยไปมาก) และ EX01 ผลลัพธ์ไม่ถูกต้อง
2. คาดว่า EX02 เร็วสุดเนื่องจากมีการแบ่งเทรดการทำงาน และมีการ lock ค่าเฉพาะตอนที่อัพเดทค่าทำให้ฟังก์ชัน plus กับ minus ทำงานพร้อมกันได้ และมีการอัพเดทค่าที่ถูกต้อง
3. Ex00: sum = 1000000, Time used: 3ms

   Ex01: sum = 1392241157, Time used: 3ms ***

   Ex02: sum = 1000000, Time used: 72ms

   Ex03: sum = 1000000, Time used: 4ms

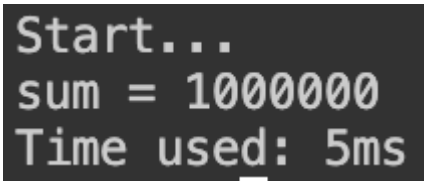
4. หลังจากการรันปรากฏว่าผลที่คาดการณ์ไว้กับการรันจริงๆ ไม่ตรงกันโดย Ex00 เร็วสุดและจากการคาดการณ์ว่า Ex02 จะเร็วสุดกลับได้ผลช้า
5. เนื่องจาก Ex02 มีการ lock ค่าทุกครั้งทำให้เกิดการรอกันจนเกิด delay เป็นจำนวนมาก และจากการทดลองจะเห็นได้ว่า Ex00 กับ Ex01 มีเวลาห่างกันเพียง 1 ms ซึ่งใกล้เคียงกันมากเนื่องจากโปรแกรมใน Ex03 ที่มาการ lock ทั้ง loop ที่ต้องรอให้ Thread นึงทำเสร็จก่อนอีก Thread นึงจึงสามารถทำต่อได้ ซึ่งมีค่าไม่ต่างกับ Ex00 ที่ทำงาน Thread เดียว
6. การที่ Ex00 ทำงานไวสุดแม้จะเปนการทำงาน single core เนื่องจากการทำงานไม่ได้เยอะมากจึงไม่เห็นผล หากเพิ่ม loop เป็น 1000000000 การทำงานแบบ single จะเห็นผลว่าช้า

64010659 :

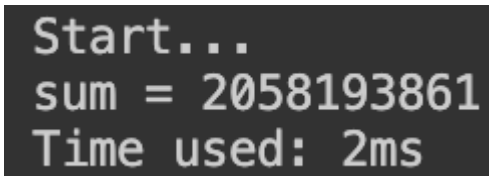
ผลลัพธ์ที่คาดการณ์ : Ex01 ที่ run ออกมาแล้วพบว่าผลลัพธ์ไม่ถูกต้อง ส่วน Ex00 พบว่าผลลัพธ์ถูกต้อง ด้วยความเร็ว 5 ms รองลงมาด้วย Ex03 พบว่าผลลัพธ์ถูกต้อง ด้วยความเร็ว 8 ms รองลงมาด้วย และสุดท้ายคือ Ex02 พบว่าผลลัพธ์ถูกต้อง ด้วยความเร็ว 81 ms ตามลำดับ

ระยะเวลาในการทำงาน & ผลลัพธ์จากการ run & เปรียบเทียบผลลัพธ์ :

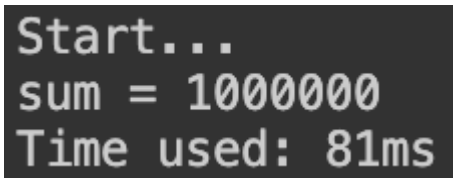
Ex00 :



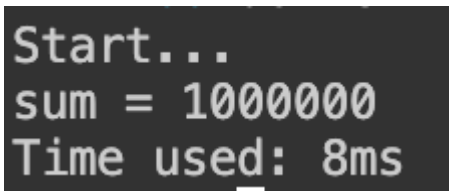
Ex01 :



Ex02 :



Ex03 :



สรุป : จากระยะเวลาในการทำงาน และผลลัพธ์จากการ run จะเห็นได้ว่า Ex01 ใช้ระยะเวลาในการทำงานน้อยที่สุด แต่ผลลัพธ์ไม่ถูกต้อง เพราะ EX01 ใช้เวลาการทำงานที่เร็วที่สุดเนื่องจากมีการแบ่ง Thread ในการทำงาน และมีการ lock ค่าเฉพาะตอนที่ update ค่าทำให้ฟังก์ชัน plus กับ minus ทำงานพร้อมกันได้ และมี update ค่าที่ถูกต้อง ดังนั้นระยะเวลาในการทำงาน : EX02 > EX00 > EX03 (เรียงจากการใช้เวลาน้อยสุดไปมากสุด ไม่นับรวม EX01)

64010683 :

1. คาดว่าโปรแกรม Ex00, Ex02, Ex03 จะได้คำตอบที่ถูกต้อง แต่โปรแกรม Ex01 จะได้คำตอบที่ไม่ถูกต้อง
2. Ex00 เพราะตัวเลขที่ต้องคำนวณไม่ได้มากจนต้องใช้หลาย Thread การทำงานแบบ Single Thread จึงเร็วกว่า
3. Ex00: sum = 1000000, Time used: 3ms

   Ex01: sum = 1392241157, Time used: 3ms

   Ex02: sum = 1000000, Time used: 72ms

   Ex03: sum = 1000000, Time used: 4ms

4. ผลที่ได้ตรงกับที่ได้คาดการณ์ไว้
5. เพราะตัวเลขที่ต้องคำนวณไม่ได้มากจนต้องใช้หลาย Thread การทำงานแบบ Single Thread จึงเร็วกว่า
6. เนื่องจาก Ex01 ไม่ได้มีการล็อกตัวแปรเอาไว้ จึงเกิด Race condition ทำให้ผลลัพธ์ที่ได้ผิดพลาด

   Ex02 มีการเช็คล็อกตัวแปรเอาไว้ทุกครั้งที่ loop จึงทำให้ใช้เวลานาน

   Ex03 มีการเช็คล็อกตัวแปรก่อนทำ loop จึงใช้เวลาไม่นานเท่า Ex02 แต่ก็ยังใช้เวลานานกว่า Ex00

2. Modification

Ex-04

Ex-04

```
using System.Threading;

namespace OS_Sync_01
{
    class Program
    {
        private static string x = "";
        private static int exitflag = 0;

        static void ThReadX()
        {
            while(exitflag==0)
                Console.WriteLine("X = {0}", x);
        }
        static void ThWriteX()
        {
            string xx;
            while (exitflag == 0)
            {
                Console.Write("Input: ");
                xx = Console.ReadLine();
                if (xx == "exit")
                    exitflag = 1;
                else
                    x = xx;
            }
        }
        static void Main(string[] args)
        {
            Thread A = new Thread(ThReadX);
            Thread B = new Thread(ThWriteX);

            A.Start();
            B.Start();
        }
    }
}
```

R-01

```
Input: 1
X = 1
Input: 2
X = 2
Input: 3
X = 3
Input: 4
X = 4
Input: 5
X = 5
Input: 6
X = 6
Input: 7
X = 7
Input: 8
X = 8
Input: 9
X = 9
Input: 99
X = 99
Input: 999
X = 999
Input: exit
Thread 1 exit
```

ให้ได้แปลงแก้ไขโปรแกรม Ex-04 ให้ทำงานแล้วได้ผลลัพธ์ดังรูป R-01 ในรูปถัดไป

** ให้ run 2 thread เท่านั้น **

64010659 :

```
using System;
using System.Diagnostics;
using System.Threading;


namespace Synchronous
{
    class EX04
    {
        private static int exitflag = 0;
        private static string x = "";
            private static readonly object lockObject = new object();
        private static bool displayX = false;
```

```
static void ThReadX()
{
    while(exitflag == 0)
        lock(lockObject)
            if (displayX)
            {
                displayX = false;
                Console.WriteLine("X = {0}",x);
            }
}

static void ThWriteX()
{
    string xx;
    while (exitflag == 0)
    {
        lock (lockObject)
        {
            Console.Write("Input: ");
            xx = Console.ReadLine();
            if (xx == "exit")
            {
                exitflag = 1;
                Console.WriteLine("Thread 1 exit...");
            }
            else
            {
                x = xx;
                displayX = true;
            }
```

```
                }
            }
        }

        static void main(String[] args)
        {
            Thread B = new Thread(ThWriteX);
            Thread A = new Thread(ThReadX);

            B.Start();
            A.Start();
        }
    }
}
```

3. Modification

Ex-05

Ex-05

```
using System;
using System.Threading;

namespace cv_lab
{
    class Program
    {
        private static string x ="";
        private static int exitflag = 0;
        private static int updateFlag = 0;

        static void ThReadX(object i)
        {
            while(exitflag == 0)
            {
                if (x != "exit")
                {
                    Console.WriteLine("***Thread {0} : x = {1}***",i,x);
                }
            }
            Console.WriteLine("---Thread {0} exit---", i);
        }
        static void ThWriteX()
        {
            string xx;
            while(exitflag == 0)
            {
                Console.Write("Input: ");
                xx = Console.ReadLine();
                if (xx == "exit")
                    exitflag = 1;
                x = xx;
            }
        }
        static void Main(string[] args)
        {
            Thread A = new Thread(ThWriteX);
            Thread B = new Thread(ThReadX);
            Thread C = new Thread(ThReadX);
            Thread D = new Thread(ThReadX);

            A.Start();
            B.Start(1);
            C.Start(2);
            D.Start(3);
        }
    }
}
```

R-02

```
Input: 1
***Thread 1 : x = 1***
Input: 2
***Thread 3 : x = 2***
Input: 3
***Thread 3 : x = 3***
Input: 4
***Thread 3 : x = 4***
Input: 5
***Thread 3 : x = 5***
Input: 6
***Thread 1 : x = 6***
Input: 7
***Thread 1 : x = 7***
Input: 1111
***Thread 1 : x = 1111***
Input: 99
***Thread 3 : x = 99***
Input: exit
---Thread 2 exit---
---Thread 3 exit---
---Thread 1 exit---
```

• ให้ดัดแปลงแก้ไขโปรแกรม Ex-05 ให้ทำงานแล้วได้ผลลัพธ์ดังรูป R-02 ใน หน้ารูปถัดไป

** ถ้ามี thread ใดแสดงออกมาแล้ว thread นั้นจะไม่แสดงซ้ำอีก **

64010659 :

```
using System;

using System.Threading;


namespace Ex_05

{

    class Program

    {

        private static string x = "";

        private static int exitflag = 0;

        private static int updateFlag = 0;

        private static int key = 0;

        private static object _Lock = new object();
```

```
static void ThReadX(object i)
{
    while (exitflag == 0)
    {
        lock (_Lock)
        {
            if (x != "exit" && key == 1)
            {
                Console.WriteLine("Thread {0} : x = {1}", i, x);
                key = 0;
            }
            else if (x == "exit")
                Console.WriteLine("---Thread {0} exit---", i);
        }
    }
}

static void ThWriteX(object i)
{
    while (exitflag == 0)
    {
        lock (_Lock)
        {
            if (key == 0)
            {
                string xx;
                Console.Write("Input : ");
                xx = Console.ReadLine();
                if (xx == "exit")
                    exitflag = 1;
```

```
                    x = xx;
                    key = 1;
                }
            }
        }
    }

    static void Main(string[] args)
    {
        Thread A = new Thread(ThWriteX);
        Thread B = new Thread(ThReadX);
        Thread C = new Thread(ThReadX);
        Thread D = new Thread(ThReadX);

        A.Start();
        B.Start(1);
        C.Start(2);
        D.Start(3);

        //A.Join();
        //B.Join();
        //C.Join();
        //D.Join();
    }
  }
}
```