Synchronization Activity
Operating Systems
Group 14

สมาชิก
62010052 กันต์ มากทรัพย์สิน
62010453 นนทพันธุ์ รุจิรกาล
62010472 นวพรรษ ศรีบุญเรือง
62010474 นวพล กรุดพันธ์
62010494 นิติพัฒน์ บุญเกตุ
62010496 นิติภูมิ คล้ายเนียม

ผลการอภิปรายภายในกลุ่มสมาชิก

โดยผลการอภิปรายในแต่ละตัวอย่างมีดังนี้

Ex00 เป็นการทำงานแบบปกติไม่มีการเรียกThred มาช่วยในกระบวนการ Process เพิ่มเติม สมาชิกในกลุ่มมีความคิดเห็นในทางเดียวกันว่า อาจจะเป็นวิธีที่มีการทำงานที่ช้าที่สุด จากโปรแกรมทั้งหมด และมีผลลัพธ์ของโปรแกรมที่ถูกต้องแน่นอน

Ex01 สมาชิกในกลุ่มเห็นพ้องต้องกันว่า Process นั้นจะมีผลลัพธ์ผิดพลาด แต่มีการทำงานที่เร็วที่สุด เพราะว่า ตัว code ยังไม่มีการปรับแก้ในการใช้ฟังชันก์ Lock ในการแก้ปัญหา จึงมีการเรียกใช้ตัวแปร sum ทำงานซ้อนกัน ทำให้ผลลัพธ์ของโปรแกรมนั้นผิดพลาด

Ex02 นั้นสมาชิกในกลุ่มมองเป็นเสียงเดียวกันว่า จะมีความเร็วในการ Process มากที่สุด เพราะว่ามีการเพิ่ม Thread ที่ช่วยในการ Process มากขึ้น และมองว่ามีผลลัพธ์ที่ถูกต้องเพราะว่าฟังชั่นก์การทำงานนั้น จะสลับกันทำงาน ตัวฟังชันก์ Lock นั้นจะถูกใช้งานภายใน Loop ครอบการเรียกใช้งานตัวแปร sum จึงไม่มีการเรียกใช้งานตัวแปร sum ในเวลาเดียวกัน ทำให้ผลลัพธ์ออกมาถูกต้อง

Ex03 สมาชิกภายในกลุ่มมีความคิดเห็นว่า การทำงานของโปรแกรม Ex03 อาจจะมีการทำงานที่คล้ายกับโปรแกรม Ex00 เนื่องจากโปรแกรมมีการเรียกใช้งาน Lock ที่ภายนอกการทำงาน Loop ทำให้ Process ต้องมีการทำงาน Loop ที่อยู่ภายในคำสั่ง Lock ก่อน ซึ่งทำให้ลำดับการทำงานนั้น คล้ายกับการทำงานของ Ex00 ดังนั้นทำให้ผลลัพธ์ของโปรแกรมเหมือนกัน และความเร็วอาจจะมีผลไม่ต่างกันมาก

ผลลัพธ์ที่ได้จากการทดสอบโปรแกรม

```
PS C:\Users\Pommland\OneDrive - KMITL\Desktop\Os_Discuss\Ex00> dotnet run
Start...
sum = 10000000
EX00 Time used : 34ms
PS C:\Users\Pommland\OneDrive - KMITL\Desktop\Os_Discuss\Ex00> dotnet run
Start...
sum = 1030480253
EX01 Time used : 75ms
PS C:\Users\Pommland\OneDrive - KMITL\Desktop\Os_Discuss\Ex00> dotnet run
Start...
sum = 10000000
EX02 Time used : 470ms
PS C:\Users\Pommland\OneDrive - KMITL\Desktop\Os_Discuss\Ex00> dotnet run
Start...
sum = 10000000
EX03 Time used : 44ms
```

เปรียบเทียบผลที่คาดการณ์กับผลที่ run ได้

จากผลการทดสอบโปรแกรมจะได้ว่า

- จากผลลัพธ์ของโปรแกรม Ex00 พบว่า Process ให้ผลลัพธ์ที่ถูกต้อง แต่ความเร็วในการทำงาน สามารถทำงานได้เร็วที่สุด จากโปรแกรมทั้งหมด เนื่องจากการทดลองอื่นๆ มี thread มีการเรียกใช้ join ทำให้ต้องรอให้ thread ที่เรียกใช้ join ทำงานให้เสร็จก่อน thread อื่นถึงจะทำงานต่อได้ จึงไม่ต่างจากการทำงานแบบ single thread และการทดลองอื่น ๆ มีการเพิ่มคำสั่งเข้าไปจึงมีผลให้ทำงานได้ช้าลง

- จากผลลัพธ์ของโปรแกรม Ex01 มีความเร็วช้ากว่า Ex00 และมีผลลัพธ์ที่ผิดพลาด เนื่องจากยังไม่ได้ใช้ฟังก์ชัน lock ในการแก้ไขปัญหา

- จากผลลัพธ์ของโปรแกรม Ex02 มีผลลัพธ์ที่ถูกต้องแต่ความเร็วในการทำงานช้าที่สุด เนื่องจากมีการเรียกใช้ฟังก์ชัน lock ทุกครั้งใน loop 1 รอบของทั้งฟังก์ชัน plus และ minus ทำให้ใช้เวลาในการทำงานมากขึ้น

- จากผลลัพธ์ของโปรแกรม Ex03 ที่ถูกนำมาแก้ไขจากEx02นั้นมีความรวดเร็วมากกว่า เพราะว่า มีการเรียกใช้ function lock น้อยกว่า Ex02

สรุปผลการทดลอง

จากการทดสอบโปรแกรมทั้งหมด พบว่าการทำงานของโปรแกรม Ex00 ทำงานได้ดีที่สุด (ความเร็วและความถูกต้อง) ทำให้ทราบว่า การใช้งานฟังก์ชัน Lock ทำให้การทำงานเกิดความล่าช้า แต่ช่วยให้การทำงานในรูปแบบ Muti-Thread ทำงานได้ถูกต้องไม่เกิดการชนกันของ Shared-Data

Modification Ex-04

- Modified Code & Results

```
{
    0 references
    class Program {
        7 references
        private static string x = "";
        3 references
        private static int exitflag = 0;

        1 reference
        static void ThReadX() {
            string xx;
            while(exitflag==0) {
                if (x == "") {
                    Console.Write("Input: ");
                    xx = Console.ReadLine();
                    x = xx;
                }
            }
        }

        1 reference
        static void ThWriteX() {
            while (exitflag == 0) {
                if (x != "") {
                    if (x == "exit") {
                        Console.WriteLine("X = {0}", x);
                        exitflag = 1;
                    }
                    else {
                        Console.WriteLine("X = {0}", x);
                        x = "";
                    }
                }
            }
        }

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

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

```
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
X = exit
```

Modification Ex-05

- Modified Code & Results

```
class Program {
        5 references
        private static string x = "";
        3 references
        private static int exitflag = 0;
        0 references
        private static int updateFlag = 0;
        2 references
        private static object _Lock = new object();
        1 reference
        static void ThReadX() {
            string xx;
            while (exitflag == 0) {
                lock (_Lock) {
                    Console.Write("Input: ");
                    xx = Console.ReadLine();
                    if (xx == "exit") {
                        exitflag = 1;
                    }
                    x = xx;
                }
            }
        }
        3 references
        static void ThWriteX(object i) {
            while(exitflag == 0) {
                lock (_Lock) {
                    if (x != "exit" && x != "") {
                        Console.WriteLine("***Thread {0} : x = {1}***", i, x);
                        x = "";
                    }
                }
            }
            Console.WriteLine("---Thread {0} exit---", i);
        }
        0 references
        static void Main(string[] args) {
            Thread A = new Thread(ThReadX);
            Thread B = new Thread(ThWriteX);
            Thread C = new Thread(ThWriteX);
            Thread D = new Thread(ThWriteX);
            A.Start();
            B.Start(1);
            C.Start(2);
            D.Start(3);
        }
```

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