5907101011 นายสิรวิชญ์ เตชะวณิชย์

**Exercise 4**

1. **เรื่อง User-Level Thread และ Kernel-Level Thread**

* **จงอธิบายความแตกต่างระหว่าง User-Level Thread และ Kernel-Level Thread**

User-level threads จะใด้รับการสนับสนุนจาก Kernel และอยู่ Library thread ในระดับของ ผู้ใช้ Library สนับสนุนการสร้าง thread การจัดเวลาและการจัดการ thread โดยไม่จำเป็นต้องใด้รับการ excute ด้วย kernel เนื่องจาก kernel ไม่ได้ยุ่งเกี่ยวกับ user level thread ถูกจัดการและสร้างทั้งหมดใน Library thread ทำให้การ context switch และการประมวลนั้นทำได้อย่างรวดเร็วถ้า Kernel เป็น Single thread แล้ว user thread level จะ block system call จนทำให้เกิดปัญหา process อื่นๆถูก block ไปด้วย

Kernel-level threads ที่ได้รับการสนับสนุนโดยตรงจากระบบปฎิบัติการ(OS) kernel จะสร้างการจัดเวลา และการจัดการ thread ภายในพื้นที่ของ kernel เอง เนื่องจากระบบปฎิบัติการเป็นผู้จัดการเกี่ยวกับการสร้าง และ จัดการ thread จึงทำให้ kernel thread ทำการสร้างการจัดการและ context switch ใด้ช้ากว่า user thread อย่างไรก็ตามถ้า thread เกิดการบล็อก system call จะทำให้ kernel จัดการนำเอา Thread อื่นๆในแอปพลิเคชั่นมาดำเนินการแทนก่อนได้

* **สถานการณ์ใดที่เหมาะสมกับการใช้ User-Level Thread และ Kernel-Level Thread**

สถานการณ์แบบ many to many และ Two level การนำเอา one to one ของ Kernel level รวมกับ many to one ของ User level thread มีการจัดการแบบแบ่ง thread ที่เกิดขึ้นกระจายตัวให้กับ kernel โดย kernel หนึ่งตัวสามารถรอบรับ thread ได้มากกว่า 1 หากเป็น Two level จะมีการเพิ่มในการใช้ one to one สำหรับงานที่มีความสำคัญสูงและยังต้องการให้ kernel ที่เหลือทำงานจาก thread อื่น ๆไปด้วย

* **สถานการณ์ใดที่เหมาะสมกับการใช้ Kernel-Level Thread**

สถานการณ์แบบ one to one Kernel ที่แต่ละ Kernel จะจับคู่กับ Thread แบบ 1 ต่อ 1

เหมาะกับการทำงานที่ต้องการความถูกต้องมีความสำคัญสูงแต่มีประสิทธิภาพในการทำงานช้า

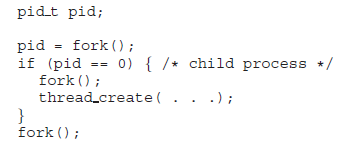
1. **สถานการณ์ใดที่ Multithreaded Solution ที่ใช้ Multiple Kernel Thread ทำให้ระบบมีประสิทธิภาพดีกว่าการใช้ Single-Threaded Solution บนระบบที่มีโพรเซสเซอร์เดียว จงยกตัวอย่างพร้อมคำอธิบาย**

เมื่อ kernel thread มีการทำงานผิดปกติทำงานผิดพลาดยังมี kernel thread อื่น ๆที่ยังสามารถสลับเพื่อมาทำงานแทนได้หากเป็น single thread นั้นไม่สามรถที่จะแก้ไขได้ไม่สามารถทำงานที่มีโอกาสที่จะเกิดความผิดพลาดบ่อยครั้ง เช่นการพัฒนาระบบหรือโปรแกรมที่มีความซับซ้อนสูง

1. **สิ่งใดต่อไปนี้ Register Values , Heap Memory , Global Variables , Stack Memory ที่ถูกแชร์ระหว่าง Multithreaded Process**

Global Variables และ Heap Memory ที่ถูกแชร์ระหว่าง Multithreaded Process

1. **จากส่วนของโค้ดต่อไปนี้**



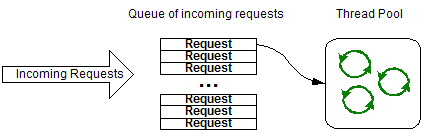
* **มีโปรเซสที่สร้างขึ้นกี่โปรเซส**

มี Process ที่สร้างขึ้น 6 Process

* **มี Thread ที่สร้างขึ้นกี่ Thread**

มี Thread ที่สร้างขึ้น 8 Thread

1. **Thread Pool คืออะไร จงอธิบาย**

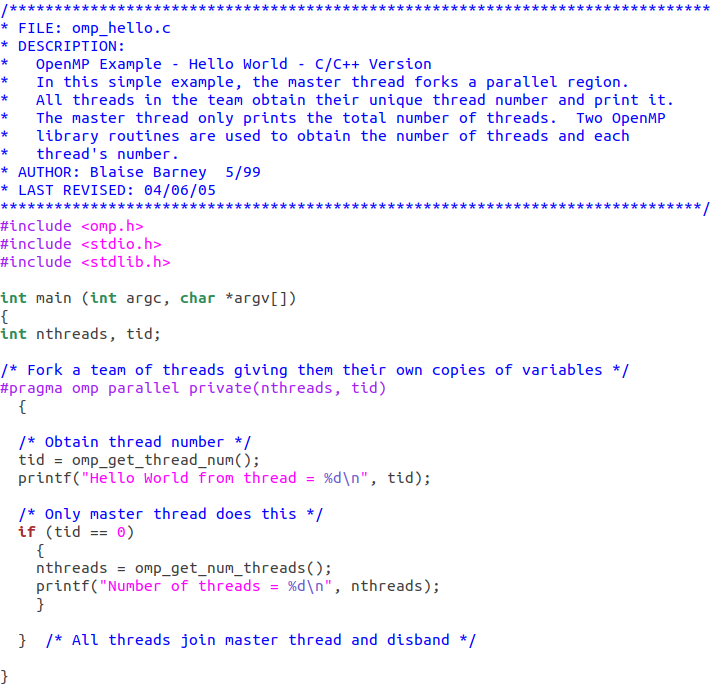


*( ภาพตัวอย่าง Diagram Thread Pool อย่างง่าย)*

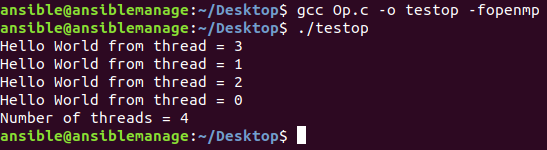
คือกลุ่มของ Thread ที่ไม่ได้ใช้งานอยู่ในสถานะ idle รอและพร้อมที่จะถูกเรียกมาใช้ Thread Pool คล้ายคลึงกับสระหรือบ่อที่รองรับเก็บหรือกัก Thread ที่ทำงานเล็ก ๆ ง่าย ๆ ไม่ยุ่งยากซับซ้อนมีความต้องในการประมวลผลอยู่บ่อย ๆ แค่ในช่วงเวลาสั้น ๆ รูปแบบ Thread Pool นั้น Run อยู่ในรูปแบบ Background Thread ซึ่ง Thread ที่อยู่ใน Thread Pool นั้นเมื่อมีความต้องการใช้งาน Thread นั้นถูกเรียกออกไปใช้โดย Thread Pool เมื่อ Thread นั้นถูกใช้งานเสร็จก็จะถูกเรียกกลับให้มาพักอยู่ใน Thread Pool เช่นเดิมการทำ Thread Pool นั้นช่วยลดระยะเวลาเพิ่มประสิทธิภาพให้แก่การทำงานแบบ Multithread ได้ดียิ่งขึ้นเนื่องจากไม่ต้องรอการสร้าง Thread สามารถนำ Thread ที่มีอยู่ใน Thread Pool มาใช้ได้ทันที

1. **จงหาตัวอย่างของโปรแกรมที่เขียนด้วย Java Thread และ OpenMP แสดงโค้ดพร้อมผลลัพธ์ของการรันที่ได้ และอธิบายพฤติกรรมของโปรแกรม**

OpenMP Code Sample code



*( ภาพประกอบ Sample code ของ OpenMP)*



*( ภาพประกอบจากผลการทดลอง )*

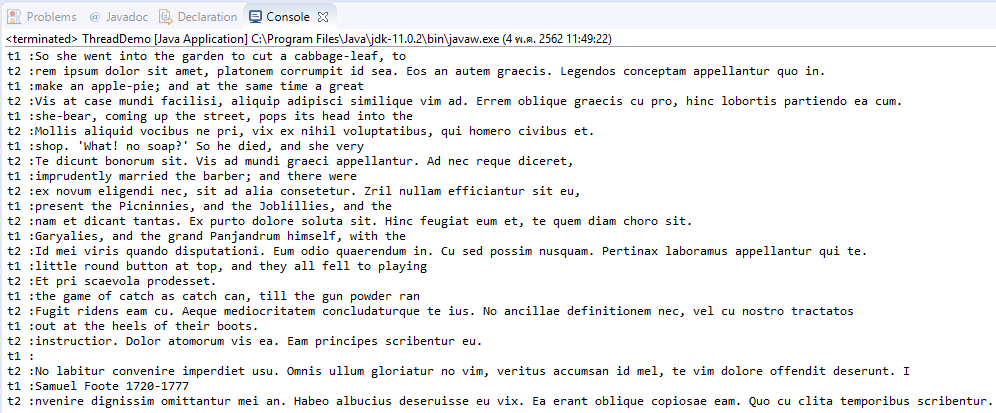
*พฤติกรรมของโปรแกรม*

โปรแกรมทำการรับ Thread ID ของ Thread มาเก็บที่ tid และหลังจากนั้นใช้ Thread ID ที่เก็บไว้ใน tid นั้นปริ้นข้อความออกมา ซึ่งจำนวนข้อความนั้นมีค่าเท่ากับตามจำนวนของ Thread ที่มีทั้งหมดและที่ Master Thread นั้นจะทำการนับค่าจำนวน Thread และเก็บไว้ใน nthreads และนำค่าในนั้นปริ้นแสดงผลออกมา

Java Thread Sample code



*( ภาพประกอบ Sample code ของ JAVA)*

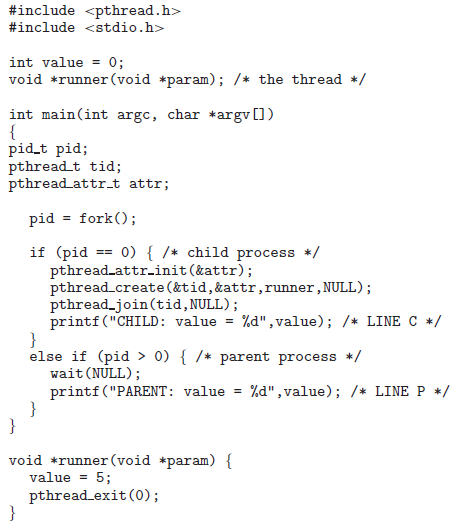


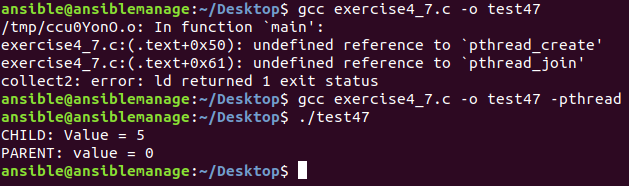
*( ภาพประกอบจากผลการทดลอง )*

พฤติกรรมของโปรแกรม

โปรแกรมทำการสร้าง Thread ขึ้นมา 2 Thread มีการเรียกใช้ Thread ทั้ง 2 Thread โดยตรงจาก Class Reader1 และ 2 ตามลำดับ Thread ทั้ง 2 มีการทำงานเหมือนกันคืออ่านค่าจากไฟล์ text แต่เป็น text ที่ต่างกันทั้งคู่ทำงานไปในเวลาพร้อม ๆกันและทำให้ผลลัพธ์ที่จะแสดงผลค่าที่อ่านได้ออกมาเป็นบรรทัดจะเห็นได้ว่ามีการสลับไปสลับมาอยู่เกือบตลอดเนื่องจาก Thread ไหนทำเสร็จก่อนก็จะปริ้นแสดงออกมาก่อน

1. **ทดลองโค้ดภาษาซีต่อไปนี้ ผลลัพธ์ที่ได้จาก Line C และ Line P เป็นอย่างไร**





*( ภาพประกอบจากผลการทดลอง )*

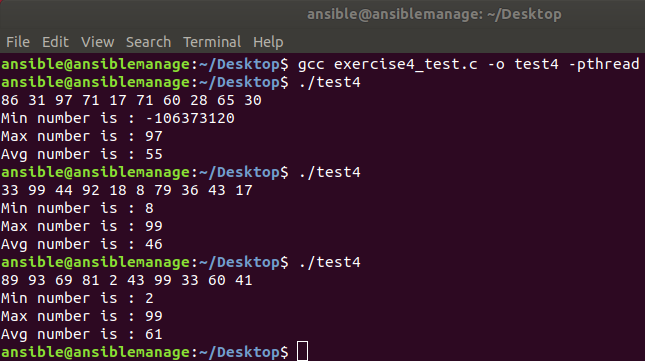
Child process ถูก fork โดย parent process หลังจาก fork แล้ว parent process จะรอ child process รันจนเสร็จสมบูรณ์มี Thread ใหม่ที่ถูกสร้างขึ้นสำหรับ child process และเรียกใช้ฟังก์ชัน runner() ซึ่งตั้งค่าของ value เป็น5หลังจากเสร็จสิ้นกระบวนการของ child process ค่าของ value ที่มีอยู่ใน parent process จะเป็น 0

1. **จงเขียนโปรแกรมแบบ Multithread โดยทีการทำงานดังนี้**

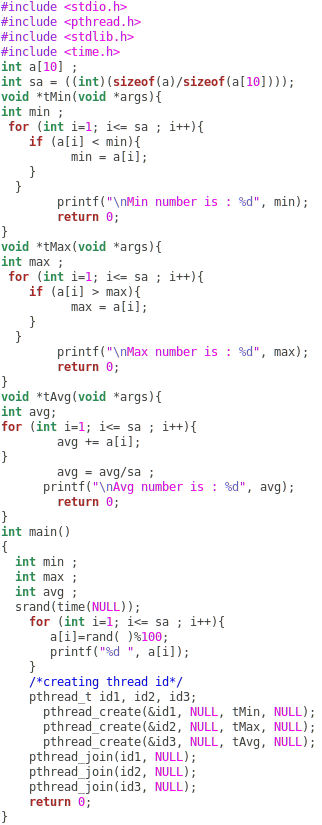
* **Main Thread สร้างเลขจำนวนเต็มแบบสุ่ม 10 ค่า แล้วแสดงออกทางจอภาพ**
* **Main Thread เตรียมตัวแปร min , max และ avg สำหรับเก็บค่าต่ำสุด , สูงสุด , และค่าเฉลี่ยตามลำดับ**
* **สร้าง Thread ย่อย 3 Thread มีหน้าที่ดังนี้ จากเลขจำนวนเต็มที่สุ่มมาจาก Main Thread**

1. **Thread#1: ค้นหาค่าต่ำสุด เก็บลงใน min**
2. **Thread#2: ค้นหาค่าสูงสุด เก็บลงใน max**
3. **Thread#3: ค้นหาค่าเฉลี่ย เก็บลงใน avg**

* **Main Thread แสดงผลลัพธ์ของ min , max , avg ออกทางจอภาพ**



*( ภาพประกอบจากผลการทดลอง )*

**