แบบฝึกหัดที่ 4

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

• จงอธิบายความแตกต่างระหว่าง User-Level Thread และ Kernel-Level Thread

ชนิดการทำงานของเธรดใน **User-Level threads** จะเป็นแบบ **many-to-one** (เป็นรูปแบบที่ใช้เธรดสำหรับระบบปฏิบัติการ 1 หน่วย กับเธรดสำหรับผู้ใช้หลายหน่วย การจัดการเธรดจะอยู่ในพื้นที่ของผู้ใช้ซึ่งมีประสิทธิภาพ แต่ถ้าเธรดบล็อก System call โพรเซสทั้งหมดจะถูกบล็อกไปด้วย)  
 ชนิดการทำงานของเธรดใน **Kernel-level threads** จะเป็นแบบ **one-to-one** (เป็นรูปแบบที่แต่ละเธรดสำหรับผู้ใช้ จะจับคู่กับเธรดสำหรับระบบปฏิบัติการ ในลักษณะ 1 ต่อ 1 ทำให้สามารถทำงานพร้อมกันดีกว่าแบบ Many-to-One โดยยอมให้เธรดอื่นรันได้เมื่อเธรดบล็อก System call นอกจากนี้โมเดลนี้ยังยอมให้หลายๆเธรดทำงานแบบขนานกันได้ในระบบมัลติโพรเซสเซอร์ได้อีกด้วย)

• สถานการณ์ใดที่เหมาะสมกับการใช้ User-Level Thread และ Kernel-Level Thread

เป็นรูปแบบของเธรดที่พยายามจะเชื่อมช่องว่างระหว่างการทำงานของเธรดในรูปแบบ Many-to-one กับ One-to-one เข้าด้วยกันโดยนำ User-level threads กับ Kernel-level threads มาพัฒนารวมกันเกิดเป็นการทำงานของเธรดที่เรียกว่า Many-to-many หลักการทำงานคือ จะ Mapping ตัว User-level threads หลายๆตัวเข้ากับชุดของ Kernel-level threads ตัวอย่างของรูปแบบนี้คือ ระบบปฏิบัติการ ยูนิกส์

• สถานการณ์ใดที่เหมาะสมกับการใช้ Kernel-Level Thread

ชนิดการทำงานของเธรดใน Kernel-level threads จะเป็นแบบ one-to-one หมายถึง ระบบปฎิบัติการจะวางเธรดแต่ละตัวไว้ใน Execution context ของตัวเอง ดังนั้น ในการวางเธรดแต่ละตัวจะต้องให้ระบบปฎิบัติการช่วยในเรื่องการ Mapping ตัว User-level threads แต่ละตัวเข้ากับ Kernel-level threads ตัวอย่างของรูปแบบนี้ คือ ในระบบปฏิบัติการวินโดว์

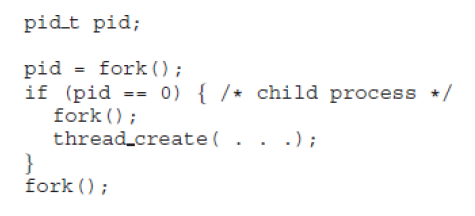
2. สถานการณ์ใดที่ Multithreaded Solution ที่ใช้ Multiple Kernel Threads ทำให้ระบบมีประสิทธิภาพดีกว่าการใช้ Single-Threaded Solution บนระบบที่มีโพรเซสเซอร์เดียว จงยกตัวอย่างพร้อมคำอธิบาย

เมื่อ kernel thread ประสบปัญหาความผิดพลาด สามารถเปลี่ยนเป็น kernel thread อื่นได้ ในขณะเดียวกันกระบวนการแบบ Single-Threaded จะไม่สามารถทำงานที่ได้เมื่อเกิดข้อผิดพลาด ดังนั้นในสถานการณ์ที่โปรแกรมอาจประสบกับความผิดพลาดบ่อย วิธีแก้ปัญหาแบบ Multithreaded Solution จะทำงานได้ดีกว่าแม้ในระบบ Single-Threaded

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

เธรดที่ถูกแชร์คือ **Heap Memory** และ **Global Variables**

4. จากส่วนของโค้ดต่อไปนี้



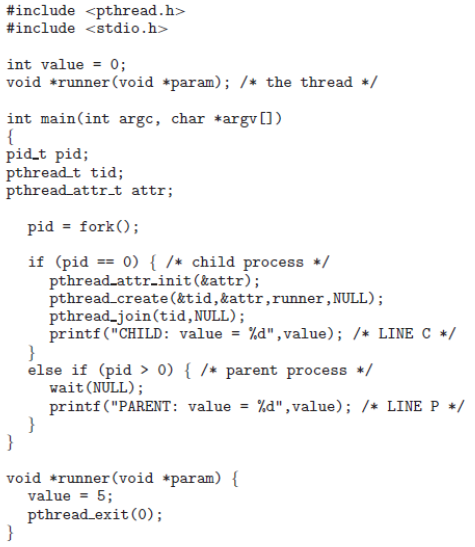
• มีโปรเซสที่สร้างขึ้นกี่โปรเซส   
มี 6 โปรเซส  
• มี Thread ที่สร้างขึ้นกี่ Thread   
มี 8 Thread

5. Thread Pool คืออะไร จงอธิบาย

Thread Pool นั้นเหมาะสำหรับที่จะใช้จัดการ Thread ที่ทำงานเล็กๆ ไม่ยุ่งยากซับซ้อนและใช้เวลาในการประมวลผลไม่นานนัก รวมทั้งไม่คำนึงถึงลำดับความสำคัญของ Thread อีกด้วยThread Pool เป็น Class ที่ .NET Framework เตรียมเอาไว้ให้สำหรับการทำงานแบบ Multi-thread ที่ง่ายๆ ไม่ซับซ้อน รวมถึงการจัดการกับการทำงานของ Method ที่ Run แบบ Asynchronous ด้วย Thread Pool นั้น Run แบบ Background Thread ดังนั้น Application สามารถหยุดการทำงานได้ทันทีถ้า Foreground Thread อื่นๆ สิ้นสุดการทำงานลง

6. จงหาตัวอย่างของโปรแกรมที่เขียนด้วย Java Thread และ OpenMP แสดงโค้ดพร้อมผลลัพธ์ของการรันที่ได้ และอธิบายพฤติกรรมของโปรแกรม

7. จงทดลองโค้ดภาษาซีต่อไปนี้ ผลลัพธ์ที่ได้จาก Line C และ Line P เป็นอย่างไร



รูปภาพประกอบด้วย ภาพหน้าจอ

คำอธิบายที่สร้างขึ้นโดยอัตโนมัติ

(undefinded reference pthread)รูปภาพประกอบด้วย ภาพหน้าจอ

คำอธิบายที่สร้างขึ้นโดยอัตโนมัติ

(ตั้งค่าใหม่ โดย เข้าไปที่โปรเจคที่เราสร้าง ตั้งค่าตามในรูป)

รูปภาพประกอบด้วย ภาพหน้าจอ

คำอธิบายที่สร้างขึ้นโดยอัตโนมัติ

(ผลลัพธ์)

CHILD value = 5

PARENT value = 0

8. จงเขียนโปรแกรมแบบ Multithread โดยมีการทำงานดังนี้

• Main Thread สร้างเลขจำนวนเต็มแบบสุ่ม 10 ค่า แล้วแสดงออกทางจอภาพ

• Main Thread เตรียมตัวแปร min, max, และ avg สำหรับเก็บค่าต่ำสุด, สูงสุด, และค่าเฉลี่ยตามลาดับ

• สร้าง Thread ย่อย 3 Thread มีหน้าที่ดังนี้ จากเลขจานวนเต็มที่สุ่มมาจาก Main Thread

i. Thread #1: ค้นหาค่าต่ำสุด เก็บลงใน min

ii. Thread #2: ค้นหาค่าสูงสุด เก็บลงใน max

iii. Thread #3: คำนวณค่าเฉลี่ย เก็บลงใน avg

• Main Thread แสดงผลลัพธ์ของ min, max, avg ออกทางจอภาพ