**แบบฝึกหัดที่ 4**

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

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

User-Level threads

ชนิดการทำงานของเธรดใน User-Level threads จะเป็นแบบ Many-to-one หมายถึง ระบบปฏิบัติการจะวางเธรดทั้งหมดของ Multithread process ไว้ใน Execution context เพียงตัวเดียว ทำงานอยู่ภายใต้พื้นที่ของผู้ใช้งาน ไม่เกี่ยวกับพื้นที่ของเคอร์เนล ทำให้ไม่สามารถเข้าใช้งานเคอร์เนลได้

Kernel-level threads

ชนิดการทำงานของเธรดใน Kernel-level threads จะเป็นแบบ one-to-one หมายถึง ระบบปฎิบัติการจะวางเธรดแตละตัวไว้ใน Execution context ของตัวเอง ดังนั้น ในการวางเธรดแต่ละตัวจะต้องให้ระบปฎิบัติการช่วยในเรื่องการ Mapping ตัว User-level threads แต่ละตัวเข้ากับ Kernel-level threads

**- 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

**- สถานการณ์ใดที่เหมาะสมกับการใช้ Kernel-Level Thread**

การทำงานของเธรดแบบ one-to-one คือ เคอร์เนลสามารถส่งเธรดของโปรเซสไปยังโปรเซสเซอร์หลายๆตัวได้ในเวลสเดียวกัน ทำให้ช่วยปรับปรุงประสิทธิภาพของการประมวลผลแบบขนานให้ดีขึ้นได้ เนื่องจากเมื่อระบบปฏิบัติการบล็อกการร้องขอใช้งานอุปกรณ์อินพุต/เอาท์พุตของเธรดใดแล้ว จะไม่ส่งผลให้เธรดทั้งหมดถูกบล็อกการทำงานไปด้วยแต่เธรดอื่นที่พร้อมสำหรับการประมวลผลจะถูกเลือกให้ได้รับการประมวลผลแทน

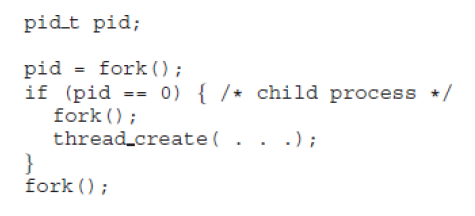
**2. สถานการณ์ใดที่ Multithreaded Solution ที่ใช้ Multiple Kernel Threads ทำให้ระบบมีประสิทธิภาพดีกว่าการใช้ Single-Threaded Solution บนระบบที่มีโพรเซสเซอร์เดียว จงยกตัวอย่างพร้อมคำอธิบาย**

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**

Threadที่ถูกแชร์คือ Heap Memory และ Global Variables

**4. จากส่วนของโค้ดต่อไปนี้ มีโปรเซสที่สร้างขึ้นกี่โปรเซสมี Thread ที่สร้างขึ้นกี่ Thread**



มี 6 โปรเซสและมี 8 Thread

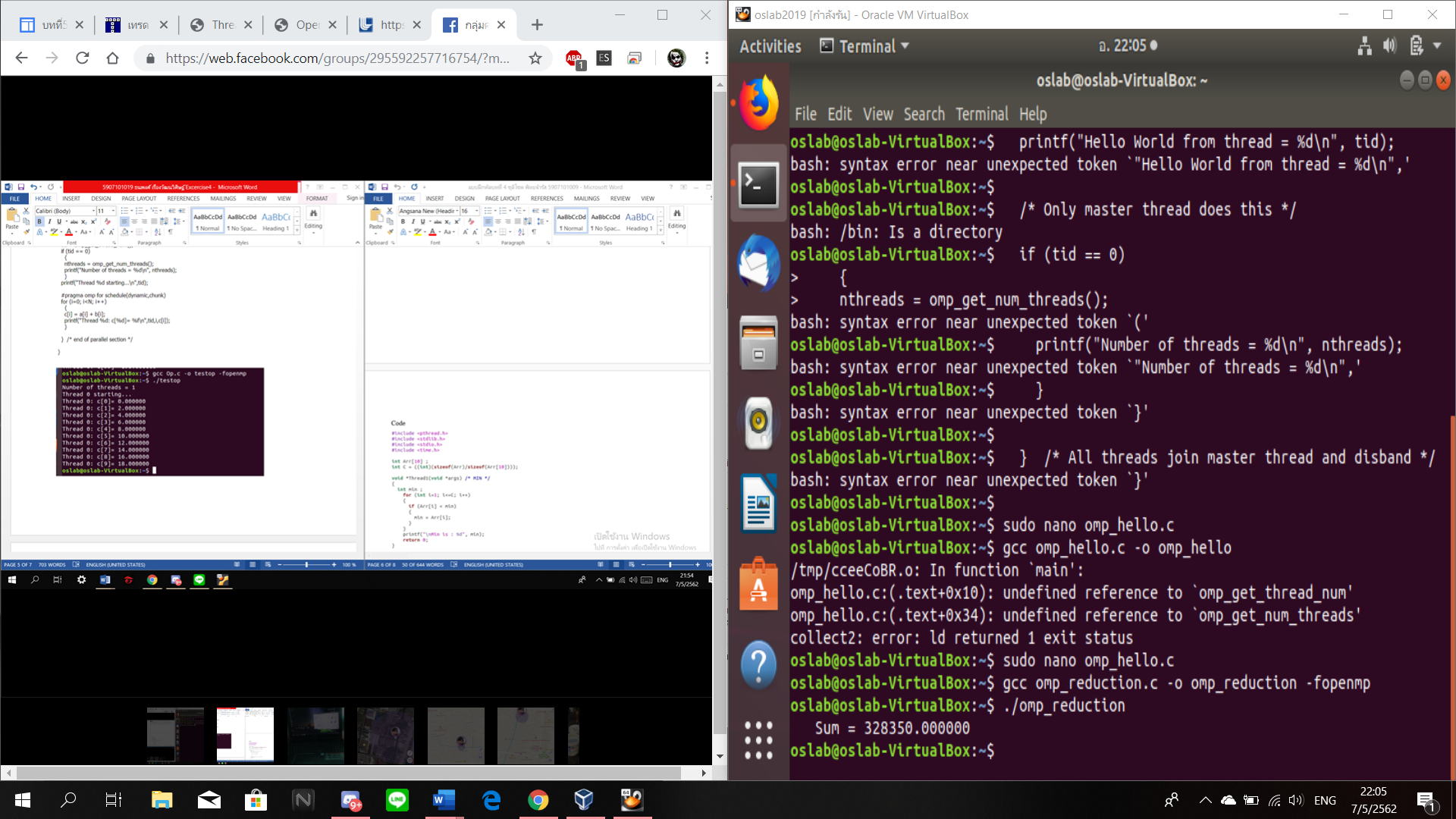
**5. Thread Pool คืออะไร จงอธิบาย**

Thread Pool เป็น Class ที่ .NET Framework เตรียมเอาไว้ให้สำหรับการทำงานแบบ Multi-thread ที่ง่ายๆ ไม่ซับซ้อน รวมถึงการจัดการกับการทำงานของ Method ที่ Run แบบ Asynchronous ด้วย

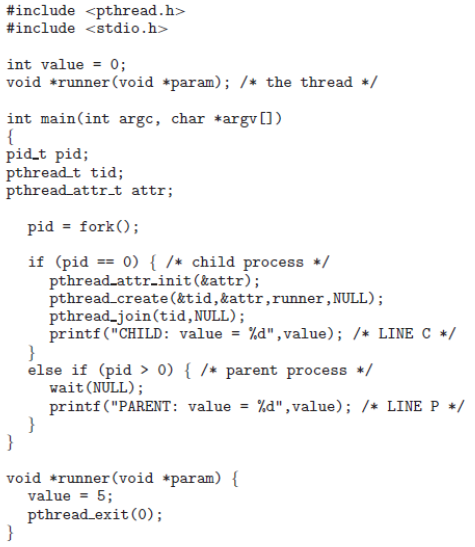
Thread Pool นั้น Run แบบ Background Thread ดังนั้น Application สามารถหยุดการทำงานได้ทันทีถ้า Foreground Thread อื่นๆ สิ้นสุดการทำงานลง

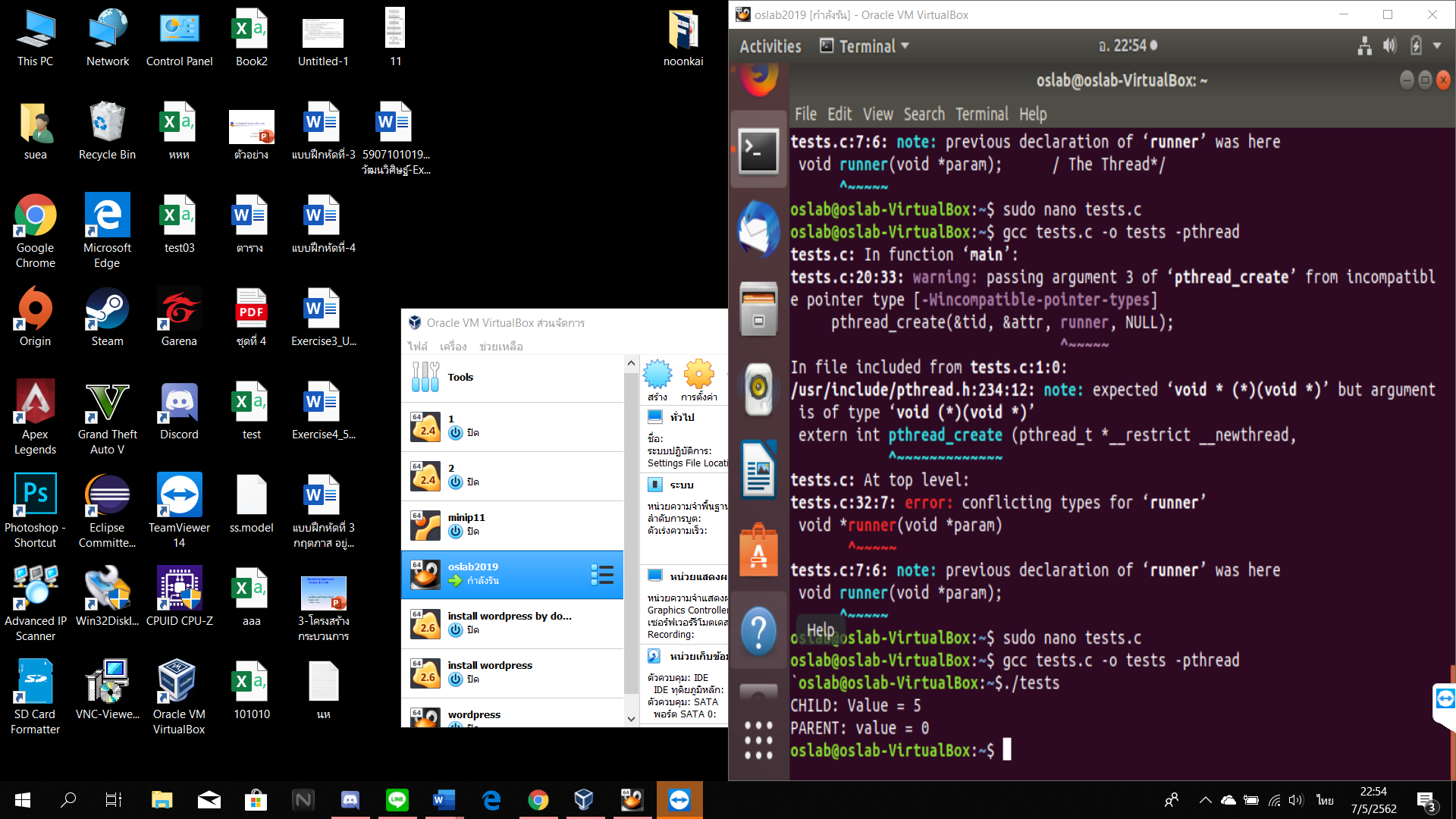
Thread Pool นั้นเหมาะสำหรับที่จะใช้จัดการ Thread ที่ทำงานเล็กๆ ไม่ยุ่งยากซับซ้อนและใช้เวลาในการประมวลผลไม่นานนัก รวมทั้งไม่คำนึงถึงลำดับความสำคัญของ Thread อีกด้วย

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

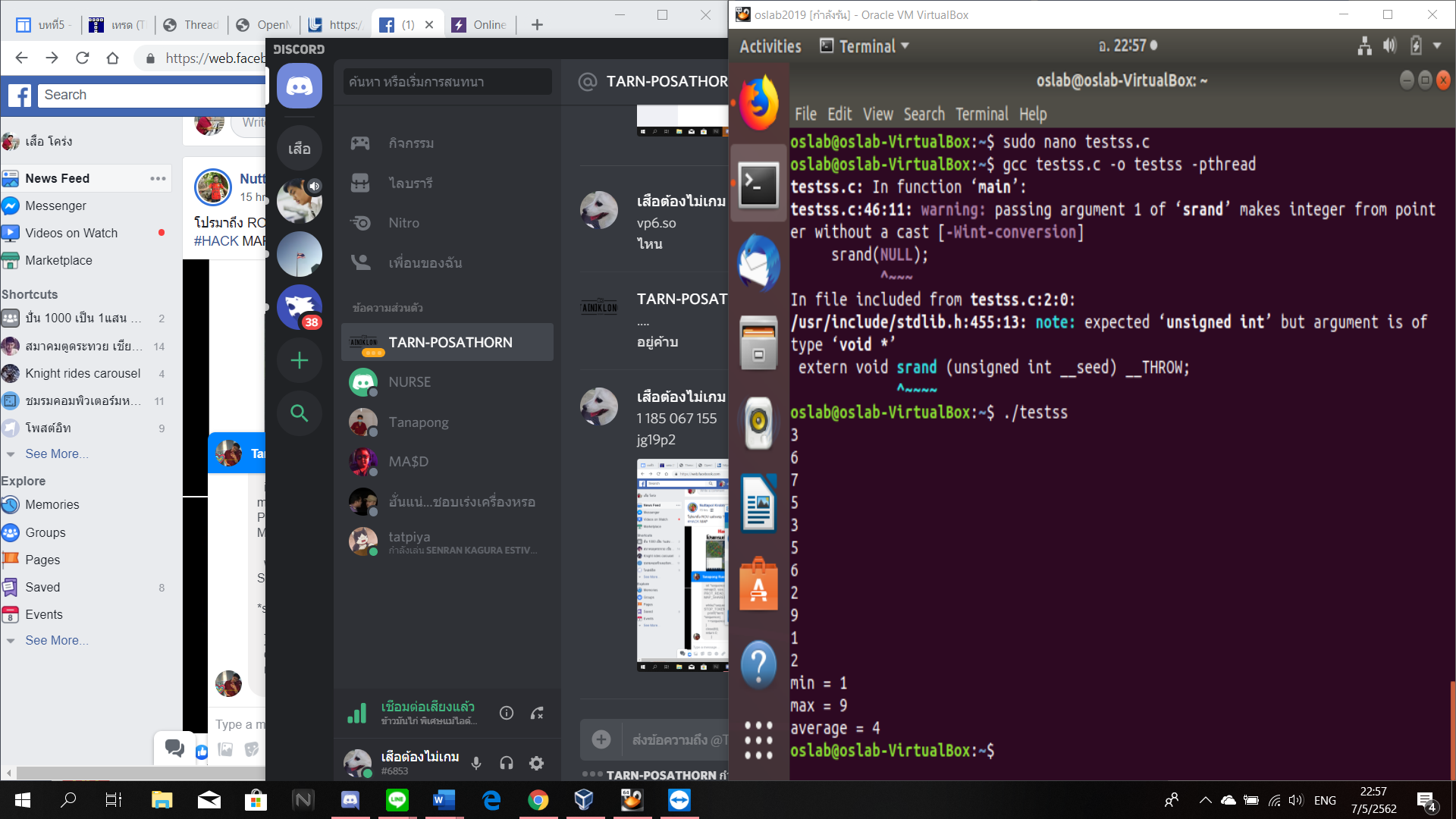


**7. จงทดลองโค้ดภาษาซีต่อไปนี้ ผลลัพธ์ที่ได้จาก Line C และ Line P เป็นอย่างไร**





**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