แผนบริหารการสอนประจำบทที่ 4

มัลติธเร็ดโปรแกรมมิง (Multithread Programming)

วัตถุประสงค์

1. สามารถอธิบายหลักการพื้นฐานของธเร็ด
2. สามารถอธิบายโครงสร้างพื้นฐานของการใช้งาน ซีพียู ที่สนับสนุนหลักการของ multithreaded computer system
3. สามารถอธิบาย APIs สำหรับ Pthreads, Win32 และ Java thread libraries
4. เข้าใจประเด็นปัญหาที่เกี่ยวข้องกับ multithread programming

เนื้อหา

1. หลักการพื้นฐานของธเร็ด (Principle of Threads)
2. โมเดลของการทำมัลติธเร็ด (Multithreading Models)
3. คลังของธเร็ด (Thread Library)

กิจกรรมการเรียนการสอน

1. บรรยาย
2. ฝึกปฏิบัติการระบบลีนุกซ์
3. วิดีโออนิเมชั่นการทำงานคอมพิวเตอร์
4. ค้นคว้าด้วยตัวเอง
5. รายงานหน้าชั้นเรียน
6. ฝึกปฏิบัติ
7. สอบกลางภาค

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน
2. สไลด์การสอน
3. โปรแกรม google class room

4. โปรแกรม facebook
5. โปรแกรม VMware
6. โปรแกรมระบบปฏิบัติการลีนุกซ์ Ubuntu

การวัดผลและการประเมินผล

1. แบบฝึกหัดท้ายบท
2. การถามตอบในชั้นเรียน
3. ใบงาน
4. สอบเก็บคะแนน

บทที่ 4

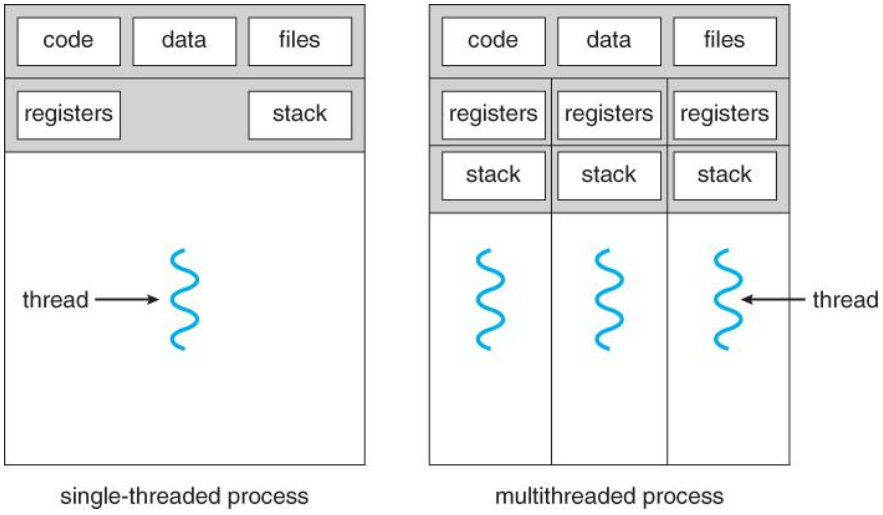
มัลติธเร็ดโปรแกรมมิง (Multithreaded Programming)

ในบทก่อนหน้า ได้อธิบายรายละเอียดของโปรเซสโดยตั้งอยู่บนสมมุติฐานว่า หนึ่งโปรเซสต่อหนึ่งธเร็ด อย่างไรก็ตาม หากมีโปรเซสใดที่ต้องบริการผู้ใช้จำนวนมาก เช่น โปรเซสของผู้ให้บริการไฟล์ จะทำให้ระบบปฏิบัติการต้องสร้างโปรเซสลูกขึ้นมาตามจำนวนผู้ร้องขอใช้บริการซึ่งต้องใช้ทรัพยากรมาก ในความเป็นจริงโปรเซสสามารถมีหน่วยทำงานย่อย คือ ธเร็ด ซึ่งถูกออกแบบให้มีการทำงานที่เหมาะสมภายใต้ระบบที่มีหน่วยประมวลผลหลายชุด (Naghibzadeh, 2011)

ในบทนี้เริ่มต้นด้วยการอธิบายหลักการพื้นฐานของธเร็ด ซึ่งจะกล่าวถึงคุณลักษณะของธเร็ด ประโยชน์ต่อการพัฒนาโปรแกรม และประเด็นที่นักพัฒนาโปรแกรมต้องคำนึงถึงเมื่อมีการใช้งานธเร็ด จากนั้นจะกล่าวถึงกลุ่มของธเร็ดและความสัมพันธ์ระหว่างกลุ่มของธเร็ด และสุดท้ายเป็นการแนะนำให้รู้จักการคลังของธเร็ดจากระบบปฏิบัติการต่าง ๆ

4.1 หลักการพื้นฐานของธเร็ด (Principle of Thread)

ธเร็ด (Thread) คือหน่วยพื้นฐานที่เล็กที่สุดที่ใช้งานซีพียู ในการประมวลผล (CPU Utilization) โดยธเร็ด 1 ชุดประกอบด้วย รหัสธเร็ด (Thread ID), รีจีสเตอร์ชนิด Program Counter (PC), ชุดรีจีสเตอร์ และสแตก (stack) ทั้งนี้หนึ่งโปรเซสอาจมีเพียงธเร็ดเดียวหรืออาจมีหลายธเร็ด (multiple threads) โดยใช้



รูปที่ 4-1: Single-threaded and multithreaded processes

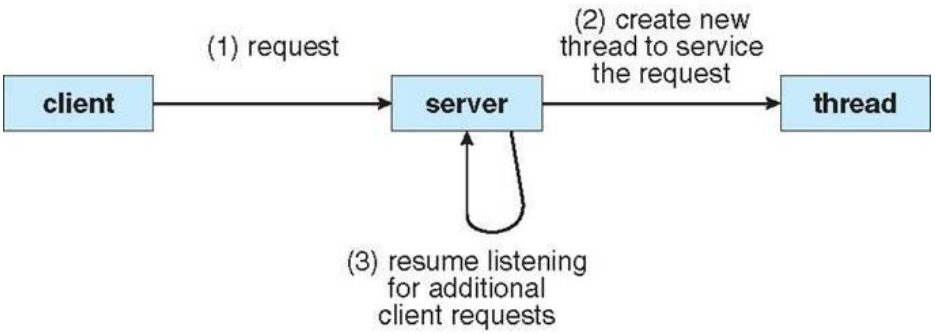
ที่มา : (Silberschatz, Galvin, & Gagne, 2010)

ทรัพยากรร่วมกัน ทรัพยากรดังกล่าวคือ ชุดคำสั่งซึ่งเป็นส่วนของเนื้อโปรแกรม (program code), ข้อมูลต่าง ๆ ที่โปรแกรมใช้งาน (data) และไฟล์ซึ่งเปิดขึ้นมาใช้งาน (opened files) รูปที่ 4-1 แสดงโปรเซสที่มีเพียงหนึ่งธเร็ด และโปรเซสที่มีหลายธเร็ด จะเห็นได้ว่ามีทรัพยากรบางส่วนที่ธเร็ดใช้ร่วมกัน และบางส่วนมีทรัพยกรเป็นของตัวเอง

4.1.1 แนวความคิดการใช้ธเร็ดในระบบปฏิบัติการ (Concept of Using Thread in Operating Systems)

ในการพัฒนาโปรแกรมให้ทำงานบนคอมพิวเตอร์ยุคใหม่นั้น แต่ละโปรเซสสามารถมีธเร็ดหลากหลายแยกย้ายกันทำงานให้ ภายใต้การควบคุม เช่น web browser อาจมีธเร็ดจำนวนหนึ่ง ทำงานหลายอย่างในเวลาเดียวกัน เช่น แสดงภาพ แสดงข้อความ และไปโหลดข้อมูลเพิ่มเติมจากแม่ข่ายผู้ให้บริการเว็ป (web server) หรือแม้แต่ web server เองก็ต้องมีธเร็ดจำนวนหลาย ๆ ตัว เพื่อให้บริการแก่ผู้ขอใช้ (client) ที่เข้ามาขอใช้บริการ ซึ่งจะเห็นว่าแท้จริงแล้วในแต่ละธเร็ดมีการทำงานลักษณะเดียวกัน เพียงแต่รับผิดชอบ client คนละชุด ดังนั้นระบบปฏิบัติการเรียกใช้งาน หลายโปรเซส (multiprocesses) หรือ หลายธเร็ด (multithreads) จึงมีแตกต่างกันที่การใช้ทรัพยากรไม่เท่ากัน กล่าวคือ multithreads มีความต้องการใช้ทรัพยกรน้อยกว่า multiprocesses เพราะทรัพยากรบางส่วนใช้งานร่วมกันระหว่างธเร็ดซึ่งอยู่ในโปรเซสเดียวกัน นอกจากนั้น multithreads มีการทำงานที่รวดเร็วกว่า เนื่องจากมีการแลกเปลี่ยนข้อมูลในลักษณะใช้พื้นที่หน่วยความจำร่วมกัน (shared memory)

พิจารณารูปที่ 4-2 ผู้ขอใช้บริการ (client) ร้องขอข้อมูลไปยังแม่ข่ายผู้ให้บริการ (server) ในการตอบสนองต่อการร้องขอ server สร้างธเร็ดขึ้นใหม่เพื่อให้บริการสำหรับเฉพาะ client นี้ จากนั้นธเร็ดปฏิบัติหน้าที่และส่งกลับข้อมูลตามที่ client ร้องขอ ในขณะเดียวกันเมื่อ server ได้แบ่งงานให้ธเร็ดได้ปฏิบัติแล้ว ก็กลับเข้าสู่การเตรียมพร้อมเพื่อรับการร้องขอใช้บริการจาก client อื่น ๆ ต่อไป



รูปที่ 4-2: Multithread on Client-Server System

ที่มา: (Silberschatz, Galvin, & Gagne, 2010)

หาก server มีการสร้างโปรเซสใหม่ซ้ำ ๆ กันทุกครั้งเมื่อมีการร้องขอจาก client จะทำให้ระบบปฏิบัติการต้องจัดสรรทรัพยากรตามความต้องการของโปรเซสที่สร้างขึ้นใหม่เหล่านั้น จึงเป็นการสิ้นเปลืองทรัพยากรโดยเฉพาะหน่วยความหลักที่มีจำกัด และใช้เวลาในการสร้างโปรเซสใหม่อีกด้วย ดังนั้น ระบบปฏิบัติการของ server เพียงสร้างธเร็ดใหม่ขึ้นมาใช้งาน เพื่อบริการให้แก่บรรดา clients ที่ร้องขอใช้บริการ โดยสามารถใช้ทรัพยากรบางส่วนร่วมกันอย่างคุ้มค่า

4.1.2 ประโยชน์จากการใช้งานมัลติธเร็ด (Benefits of Multithread)

ประโยชน์ของการใช้งาน multithread สามารถอธิบายได้ 4 ประการคือ

1) Responsiveness: ทำให้การตอบสนองต่อผู้ใช้มีคุณภาพ เมื่อภาระงานถูกประมวลโดยธเร็ดคู่ขนานไปกับการทำตามคำสั่งอื่น ๆ ของผู้ใช้

2) Resource sharing: การแบ่งการใช้งานทรัพยากรด้านหน่วยความจำ เช่น program code และ data

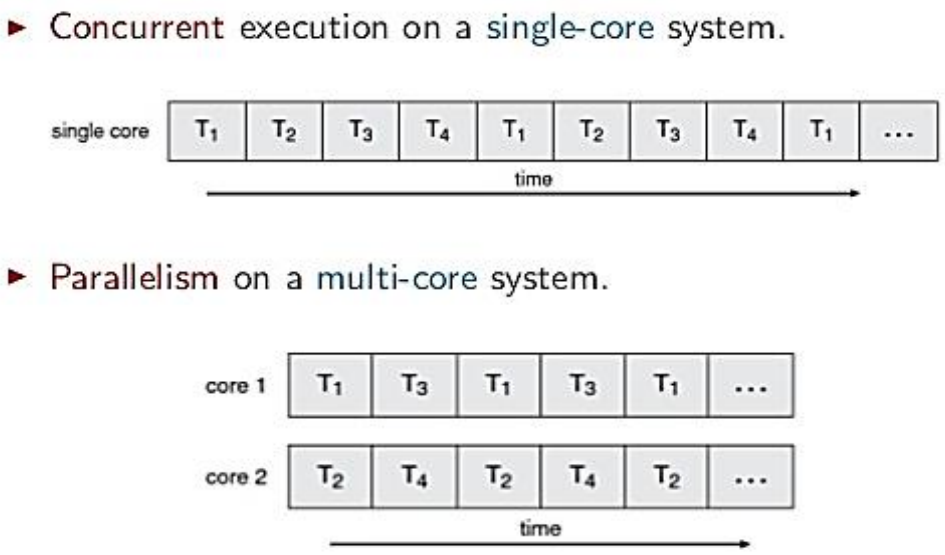
3) Economy: ประหยัดทั้งในด้านทรัพยากรและเวลา เนื่องจากการใช้ธเร็ดสามารถเข้าถึงและควบคุมการทำงานได้ดีกว่าการใช้โปรเซสใหม่ เช่นการสร้างธเร็ดใหม่ทำได้เร็วกว่าการสร้างโปรเซสใหม่ถึง 30 เท่า และการสลับเนื้อหา (context switch) สามารถทำได้เร็วกว่า 5 เท่าในระบบปฏิบัติการ Solaris

4) Scalability: multithread เหมาะสมกับโครงสร้างคอมพิวเตอร์แบบมีหลายหน่วยประมวลผล (multi-CPU machine) ซึ่งเป็นโครงสร้างระบบคอมพิวเตอร์ยุคใหม่

4.1.3 ธเร็ดบนระบบหลายหน่วยประมวลผล (Thread in Multiple-Processor System)

พัฒนาการทางด้านฮาร์ดแวร์ของระบบคอมพิวเตอร์ยุคใหม่ ได้เพิ่มความสามารถในการประมวลผลไม่เพียงด้านความเร็ว แต่รวมถึงการประมวลผลคู่ขนาน ด้วยการออกแบบชิบให้มีหน่วยประมวลผลหลายชุด (multiple core system) ซึ่งสอดรับกับโปรแกรมที่ถูกออกแบบให้มีการทำงานในลักษณะ multithreaded โปรแกรมด้วยคุณลักษณะดังกล่าวจะสามารถใช้ประโยชน์จากโครงสร้างทางฮาร์ดแวร์ระบบคอมพิวเตอร์ยุคใหม่ได้อย่างมีประสิทธิภาพโดยการทำงานคู่ขนานกันอย่างสมบูรณ์ (Bauer, 1992)

พิจารณารูปที่ 4-3 แสดงการทำงานของธเร็ดจำนวน 4 ธเร็ดบนระบบหน่วยประมวลผลเดี่ยว (single-core system) และระบบหลายหน่วยประมวลผล (multicore system) ในระบบหน่วยประมวลผลเดี่ยวนั้น ธเร็ดถูกนำเข้าสู่การประมวลตามลำดับ ซึ่งแตกต่างจากระบบหลายหน่วยประมวลผล จากรูประบบมีหน่วยประมวลผล 2 ชุด ทำให้ระบบปฏิบัติการสามารถบรรจุธเร็ดให้ได้รับการประมวลผลคู่ขนานกัน ดังนั้นเวลาที่ใช้ในการประมวลผลในงานหนึ่ง ๆ จะกระทำได้รวดเร็ว



รูปที่ 4-3: การประมวลผลในระบบหน่วยประมวลผลเดี่ยว (single-core system) และระบบหลายหน่วยประมวลผล (multicore system)

ที่มา : (Stallings, 2017)

4.1.4 ประเด็นในการพัฒนาโปรแกรมด้วยคุณลักษณะมัลติธเร็ด (Issues of Software Developemnt with Multithread)

เมื่อโครงสร้างทางด้านฮาร์ดแวร์ของคอมพิวเตอร์พัฒนาไปสู่ระบบประมวลผลแบบหลายหน่วย (multiple cores) ทำให้การออกแบบและการเขียนโปรแกรมต้องคำนึงถึงการประมวลผลแบบขนาน (parallel execution) เพื่อให้คอมพิวเตอร์ทำงานเต็มประสิทธิภาพ อย่างไรก็ตามการพัฒนาโปรแกรมในลักษณะนี้จำเป็นต้องพิจารณาสิ่งต่อไปนี้

1) Dividing activities: การแบ่งงานออกเป็นส่วน ๆ เพื่อให้งานต่าง ๆ ประมวลผลคู่ขนานกันได้โดยไม่เกิดข้อผิดพลาด

2) Balance: ความสมดุลย์ในการจัดสรรภาระงานให้แต่ละหน่วยประมวลผลอย่างเท่าเทียมทั้งด้านการใช้ทรัพยากรและด้านการประมวลผล

3) Data splitting: เนื่องจากงานได้ถูกแบ่งออกเป็นงานย่อย ๆ เพื่อให้ธเร็ดได้นำไปปฏิบัติและมีความจำเป็นในการเข้าถึงข้อมูล ดังนั้นการจัดแบ่งหรือจัดสัดส่วนข้อมูล เพื่อรองรับการทำงานจากหลายหน่วยประมวลผลจึงเป็นเรื่องสำคัญ

4) Data dependency: ข้อมูลจำเป็นต้องถูกควบคุมให้มีความถูกต้องจากการเข้าใช้งานโดยธเร็ดหลาย ๆ ชุด เนื่องจากธเร็ดมีการใช้ข้อมูลร่วมกัน และการทำงานของธเร็ดอาจทำงานในเวลาเดียวกัน ดังนั้นความสอดคล้องกันในการอ่านและเขียนข้อมูลให้ถูกต้องจึงมีความสำคัญ

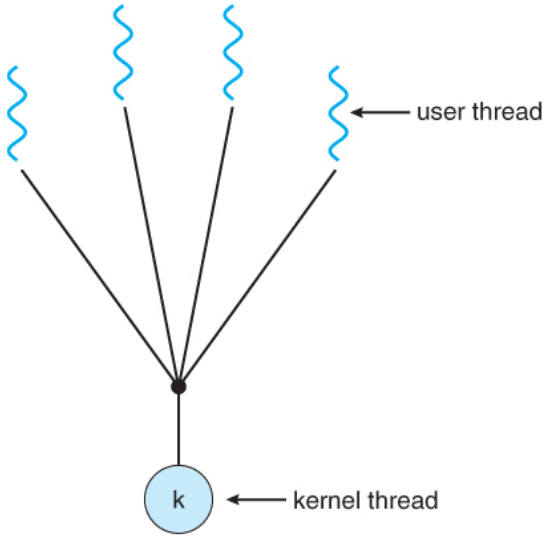
5) Testing and debugging: การทดสอบและตรวจสอบการทำงานของโปรแกรมอาจทำได้อยากในระบบที่มีการประมวลผลแบบคู่ขนาน เนื่องจากการค้นหาจุดบกพร่องของโปรแกรมซึ่งเกิดจากการทำงานของธเร็ดในหน่วยประมวลผลหนึ่ง อาจมีสาเหตุมาจากการทำงานที่ผิดพลาดของธเร็ดที่อยู่ในอีกหน่วยประมวลผลก็เป็นได้

4.2 โมเดลของการทำมัลติธเร็ด (Multithreading Models)

ธเร็ดสามารถแบ่งออกเป็น 2 ประเภทคือ ธเร็ดผู้ใช้ (user thread) และ ธเร็ดแก่น (kernel thread) ธเร็ดผู้ใช้คือธเร็ดที่ทำงานและถูกจัดการในระดับของผู้ใช้ซึ่งมีคลังของธเร็ด (thread library) เป็นแหล่งรองรับการใช้งาน โดยไม่เกี่ยวข้องกับการเรียกใช้บริการของระบบ ในทางกลับกันธเร็ดแก่นนั้นระบบปฏิบัติการเป็นผู้จัดการโดยตรง อย่างไรก็ตามความสัมพันธ์ระหว่างธเร็ดทั้งสองกลุ่มจำเป็นต้องมีเพื่อผู้ใช้สามารถใช้ถึงทรัพยากรของระบบได้ โดยความสัมพันธ์ระหว่างธเร็วผู้ใช้และธเร็ดแก่นสามารถอธิบายเป็นโมเดลได้ใน 3 ลักษณะ คือ 1) กลุ่ม-ต่อ-หนึ่ง (many-to-one model) 2) หนึ่ง-ต่อ-หนึ่ง (one-to-one model) และ 3) กลุ่ม-ต่อ-กลุ่ม (many-to-many model)

4.2.1 โมเดล กลุ่ม-ต่อ-หนึ่ง (many-to-one model)

ความสัมพันธ์ระหว่างธเร็ดผู้ใช้ (user thread) และธเร็ดแก่น (kernel thread) เป็นแบบกลุ่ม-ต่อ-หนึ่ง ดังรูปที่ 4-4 โดยในชั้นของผู้ใช้ประกอบด้วยธเร็ดหลายชุด ซึ่งมีการทำงานประสานกับธเร็ดแก่นในระดับระบบปฏิบัติการเพียงหนึ่ง ธเร็ด ในลักษณะความสัมพันธ์เช่นนี้ทำให้การจัดการธเร็ดอยู่ในส่วนพื้นที่ของผู้ใช้เป็นส่วนใหญ่ โดยเรียกใช้งานจากคลังของธเร็ด (thread library) การทำงานจึงมีประสิทธิภาพ อย่างไรก็ตามกลุ่มของธเร็ดผู้ใช้เมื่อต้องการใช้เข้าถึงทรัพยากรผ่านระบบปฏิบัติการ มีเพียงธเร็ดผู้ใช้เพียง 1 ชุดเท่านั้นที่สามารถเข้าใช้ธเร็ดแก่นได้ในเวลาหนึ่ง ๆ จึงเป็นสาเหตุให้กลุ่มของธเร็ด (multiple threads) ไม่สามารถทำงานพร้อมกันได้ และไม่สามารถใช้ความสามารถของหน่วยประมวลผลหลายชุด (multiple cores) ได้ (Godbole, 2008)

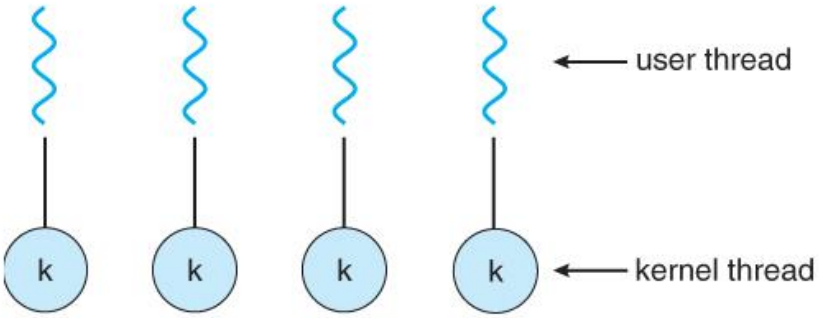


รูปที่ 4-4: ความสัมพันธ์ แบบ กลุ่ม-ต่อ-หนึ่ง

ที่มา: (Silberschatz, Galvin, & Gagne, 2010)

4.2.2 โมเดล หนึ่ง-ต่อ-หนึ่ง (one-to-one model)

โมเดลแบบ หนึ่ง-ต่อ-หนึ่ง (One-to-one model) เป็นลักษณะที่ธเร็ดผู้ใช้เข้าถึงธเร็ดแก่นในแบบ หนึ่ง-ต่อ-หนึ่ง ดังรูปที่ 4-5 ความสัมพันธ์ในลักษณะนี้ทำให้การประมวลผลสามารถใช้งานหน่วยประมวลผลได้หลายหน่วย ไปพร้อม ๆ กัน (Dhotre, Operating System, 2007) ทั้งนี้เนื่องจากมีธเร็ดแก่นรองรับการทำงานของธเร็ดผู้ใช้อย่างทั่วถึง อย่างไรก็ตามการใช้งานธเร็ดแก่นจำนวนหลายตัวในเวลาเดียวกันนั้น เป็นเหตุการสิ้นเปลื่องทรัพยากร และทำให้ประสิทธิภาพการทำงานของการประมวลผลลดลง ดังนั้นจำนวนธเร็ดแก่นจะต้องมีความเหมาะสมซึ่งขึ้นอยู่กับระบบปฏิบัติการนั้น ๆ ระบบปฏิบัติการ Windows และ Linux ใช้โมเดลนี้

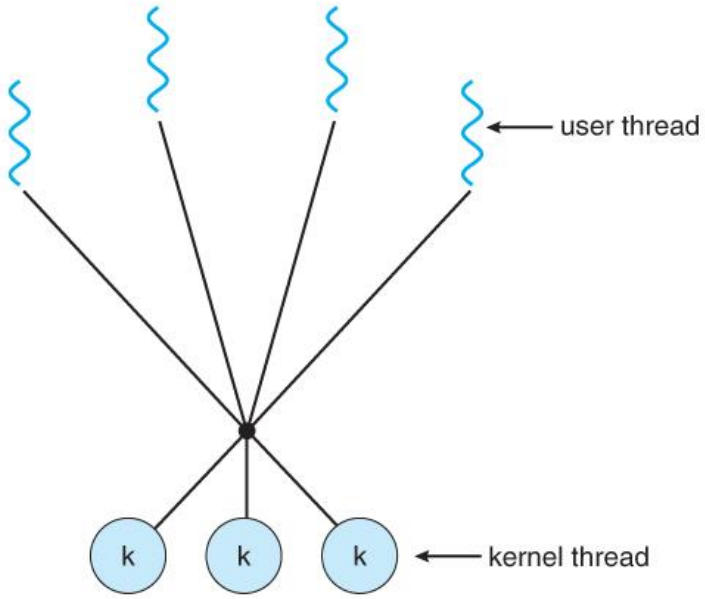


รูปที่ 4-5: ความสัมพันธ์ แบบ หนึ่ง-ต่อ-หนึ่ง

ที่มา: (Silberschatz, Galvin, & Gagne, 2010)

4.2.3 โมเดล กลุ่ม-ต่อ-กลุ่ม (many-to-many)

โมเดล กลุ่ม-ต่อ-กลุ่ม (many-to-many) เป็นการประสานการทำงานระหว่างธเร็ดผู้ใช้จำนวนหลายชุด เข้ากับธเร็ดแก่นหลายชุด ดังรูปที่ 4-6 แต่มีจำนวนที่น้อยกว่าหรือเท่ากัน จำนวนของธเร็ดแก่นอาจจะกำหนดเฉพาะเจาะจงโปรแกรมหรืออาร์ดแวร์ได้ อย่างไรก็ตามความสัมพันธ์แบบนี้ค่อนข้างยุ่งยากในการจัดการ (Dhotre, Operating Systems, 2009)



รูปที่ 4-6: ความสัมพันธ์ แบบ กลุ่ม-ต่อ-กลุ่ม

4.3 คลังของธเร็ด (Thread Library)

คลังของธเร็ด (thread library) เป็นสิ่งสนับสนุนให้แก่นักพัฒนาโปรแกรมในการเรียกใช้ และจัดการธเร็ด ผ่าน API (Application Programming Interface) โดย คลังแบ่งออกเป็น 2 ประเภทคือ

1) ประเภทสำหรับธเร็ดผู้ใช้ โดย code และ data อยู่ในส่วนของผู้ใช้เท่านั้น ฟังก์ชันต่าง ๆ ที่ใช้งานอยู่ในระดับของผู้ใช้ ซึ่งไม่มีการเรียกใช้งาน system call ในชั้นแก่นของระบบปฏิบัติการแต่อย่างใด

2) ประเภทสำหรับธเร็ดแก่นเป็นคลังที่ให้บริการโดยชั้นแก่นของระบบปฏิบัติการ ดังนั้น code และ data จึงอยู่ในพื้นที่ของแก่น

ปัจจุบันมีคลังของธเร็ดอยู่ 3 กลุ่มหลัก คือ POSIX Pthreads, Win32 และ Java

4.3.1 พีธเร็ด (Pthreads)

Pthreads สามารถเรียกใช้ผ่าน API ที่อ้างอิงมาตรฐาน POSIX (IEEE 1003.1c) Pthreads เป็นคลังของธเร็ดซึ่งให้บริการในระบบปฏิบัติการ Linux, Solaris, Mac X โดยรองรับการใช้งานทั้งในระดับธเร็ดผู้ใช้ และธเร็ดแก่น นอกจากนี้ Pthreads พัฒนาด้วยภาษา C เมื่อต้องการใช้ธเร็ด โปรแกรมเมอร์ต้องประกาศการควบรวมธเร็ดจากคลังด้วยคำสั่ง include <pthread.h> โดยที่ Pthreads มีฟังก์ชันที่สำคัญ ๆ เช่น pthread_create(), pthread_join() และ pthread_exit()

4.3.2 วิน 32 (Win32)

คลังของธเร็ดสำหรับ Win32 ใช้งานในระดับธเร็ดแก่น โดยให้บริการในระบบปฏิบัติการ Windows ผ่าน API ซึ่งมีลักษณะคล้ายกับ Pthreads การใช้งานจำเป็นต้องเรียกใช้จากคลังด้วยชื่อคือ include <windows.h> ฟังก์ชันที่สำคัญ เช่น CreateThread() และ WaitForSingleObject()

4.3.3 จาวา (Java)

คลังของธเร็ดสำหรับ Java สามารถเรียกใช้งานผ่าน java programing บนระบบปฏิบัติการใด ๆ ที่สามารถทำงานกับ Java Virtual Machine (JVM) ซึ่งสามารถสร้างธเร็ดได้ด้วยฟังก์ชัน run(), start() และ runnable interface เป็นต้น

บทสรุป

หลักการพื้นฐานของธเร็ดสืบเนื่องจากความต้องการใช้ความสามารถของระบบคอมพิวเตอร์ในยุคปัจจุบันซึ่งพัฒนาโครงสร้างหน่วยประมวลผลหลายชุด การพัฒนาโปรแกรมจึงมีเป้าหมายให้โปรเซสสามารถสร้างธเร็ดหลายชุดขึ้นมารองรับการทำงาน โดยธเร็ดเป็นหน่วยพื้นฐานในการประมวลผล ซึ่งระบบปฏิบัติการสามารถแจกจ่ายภาระการประมวลผลให้แต่ละหน่วยประมวลผลได้ ธเร็ดมีการใช้ทรัพยากรร่วมกันกับธเร็ดอื่น ๆ ในโปรเซสเดียวกัน จึงทำให้ประหยัดการทรัพยากรของระบบ นอกจากนี้ยังมีผลการทำงานที่รวดเร็ว เมื่อเทียบกับการสร้างโปรเซสใหม่ขึ้นมา

ธเร็ดสามารถแบ่งออกได้ออกเป็น 2 กลุ่มคือ ธเร็ดในระดับผู้ใช้ และธเร็ดในระดับแก่นของระบบปฏิบัติการ ซึ่งทั้งสองกลุ่มต้องทำงานประสานกัน จึงมีความสัมพันธ์เกิดขึ้นในหลายรูปแบบคือ กลุ่ม-ต่อ-หนึ่ง กลุ่ม-ต่อ-กลุ่ม และหนึ่ง-ต่อ-หนึ่ง ซึ่งมีข้อดีข้อเสียต่างกัน นอกจากนี้ธเร็ดยังมีคลังที่โปรแกรมเมอร์สามารถ

เรียกใช้งานได้ ทำงานทั้งในระดับผู้ใช้และแก่นของระบบ เช่น คลังในระบบปฏิบัติการ Linux และ Solaris คือ Pthread คลังในระบบปฏิบัติการ Windows คือ Win32 และคลังในระบบที่มี JVM คือ Java

แบบฝึกหัดท้ายบท

4.1 จงอธิบายธเร็ดคืออะไร

4.2 จงอธิบายองค์ประกอบธเร็ดมีอะไรบ้าง

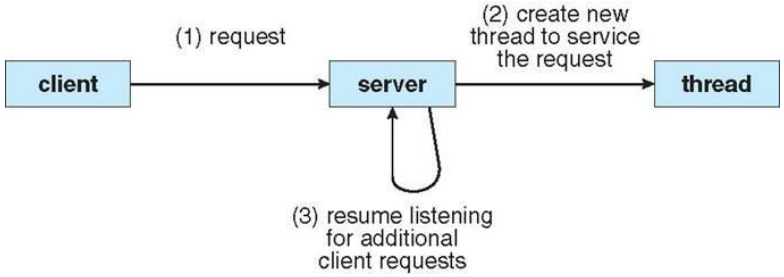
4.3 จงอธิบายเหตุผลเปรียบเทียบการทำ multiple processes และ multiple threads

4.4 จงอธิบายบทบาทของธเร็ดภายใต้ระบบคอมพิวเตอร์หน่วยประมวลผลเดี่ยวและระบบคอมพิวเตอร์หน่วยประมวลผลหลายชุด

4.5 จงอธิบายประเด็นที่จำเป็นต้องคำนึงในการใช้งานธเร็ดภายใต้ระบบคอมพิวเตอร์ที่มีหน่วยประมวลผลหลายชุด

4.6 จงบอกข้อดีของการทำมัลติธเร็ด (multithreading) มาอย่างน้อย 3 ประการ

4.7 จงอธิบายการทำงานของมัลติธเร็ด ตามรูปด้านล่างมาให้เข้าใจ



4.8 คลังของธเร็ดดังต่อไปนี้ Pthreads, Win32 และ Java ทำงานอยู่บนระบบปฏิบัติการใด

4.9 จงเปรียบเทียบข้อดีข้อเสียของโมเดลของการมัลติธเร็ดแบบ กลุ่ม-ต่อ-หนึ่ง (many-to-one) และแบบ หนึ่ง-ต่อ-หนึ่ง (one-to-one) มาให้เข้าใจ

4.10 จงแสดงตัวอย่างการเขียนโปรแกรมโดยเรียกใช้ธเร็ดจากคลังของธเร็ดใดก็ได้มา 1 ตัวอย่าง

บรรณานุกรม

Bauer, B. E. (1992). *Practical Parallel Programming.* Acadimic Press, Inc.

Dhotre, I. A. (2007). *Operating System.* Technical Publication Pune.

Dhotre, I. A. (2009). *Operating Systems* (seventh ed.). Technical Publication Pune.

Godbole, A. S. (2008). *Operating Systems.* Tata McGraw-Hill.

Naghibzadeh, M. (2011). *Operating System: Concepts and Techniques.* iUniverse, Inc.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2010). *Operating System Concepts* (8th ed.). Asia: John Wiley & Sons.

Stallings, W. (2017). *Operating Systems : Internals and Design Principles* (Fourth Edition ed.). Pearson.