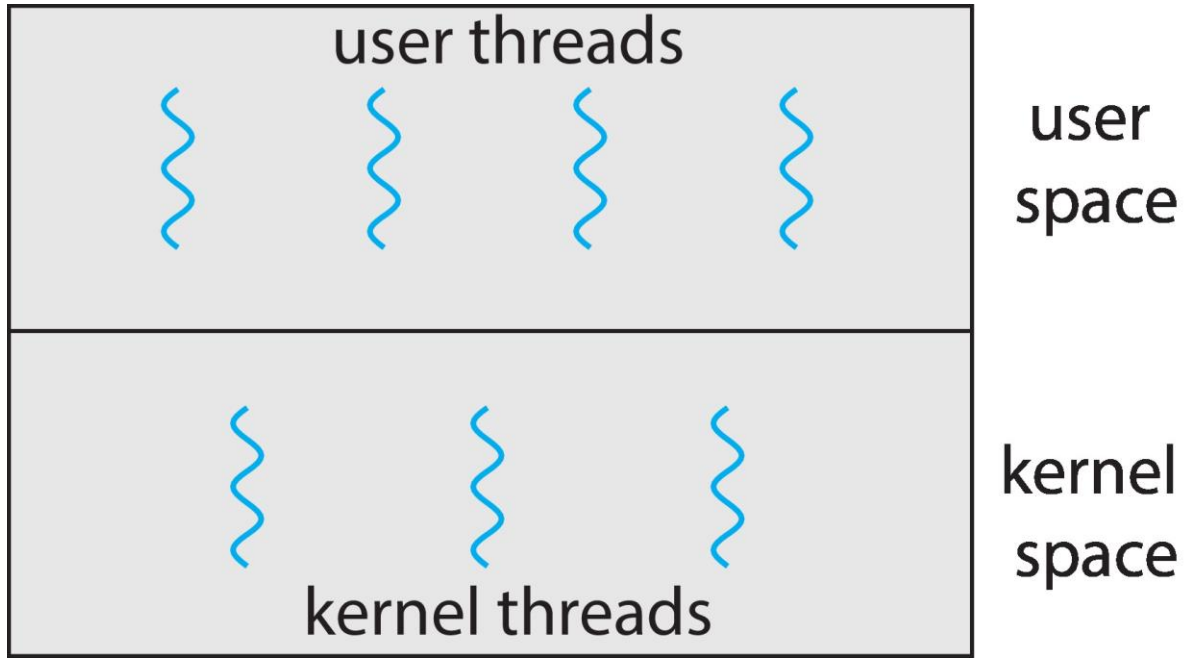
บทที่ 4 Thread (2)

Thread สำหรับผู้ใช้และ Thread สำหรับระบบปฏิบัติการ

เธรดอาจจะแบ่งตามระดับการสนับสนุนได้ 2 แบบที่สัมพันธ์กัน คือ

1. เธรดสำหรับผู้ใช้ - ง่ายที่จะถูกสร้างและอาจถูกยกเลิกก่อนเข้าเธรดสำหรับระบบปฏิบัติการได้และ
2. เธรดสำหรับระบบปฏิบัติการ - รองรับเธรดสำหรับผู้ใช้และการปฏิบัติงาน

Thread สำหรับผู้ใช้และ Thread สำหรับระบบปฏิบัติการ



Thread สำหรับผู้ใช้

การสร้างเธรดและการจัดเวลา เธรดทั้งหมดจะกระทำเสร็จสิ้นภายในพื้นที่ของผู้ใช้โดยไม่จำเป็นต้องใช้ Kernel ดังนั้นเธรดในระดับผู้ใช้สามารถสร้างและจัดการได้อย่างรวดเร็ว

อย่างไรก็ตามถ้า Kernel เป็น Single thread แล้ว เธรดระดับผู้ใช้จะบล็อก System call จนเป็นเหตุให้ทุกโพรเซสถูกบล็อก ถึงแม้ว่าเธรดอื่นจะยังคงรันอยู่ในแอปพลิเคชั่นก็ตาม

Thread สำหรับระบบปฏิบัติการ

โดย Kernel จะสร้าง จัดเวลา และจัดการเธรดภายในพื้นที่ของ Kernel เอง เนื่องจากระบบปฏิบัติการเป็นผู้จัดการเกี่ยวกับการสร้างและจัดการเธรดเอง จึงทำให้เธรดสำหรับระบบปฏิบัติการจะสร้างและจัดการได้ช้ากว่าเธรดสำหรับผู้ใช้

อย่างไรก็ตาม เพราะ Kernel จัดการเกี่ยวกับเธรด ดังนั้นถ้าเธรดเกิดการบล็อก System call จะทำให้ Kernel จัดการนำเอาเธรดอื่นในแอปพลิเคชั่นเข้ามาดำเนินการแทนได้ เช่นเดียวกับในสภาวะมัลติโพรเซสเซอร์ที่ Kernel สามารถจัดเธรดลงในโพรเซสเซอร์อื่นได้

รูปแบบของ Thread

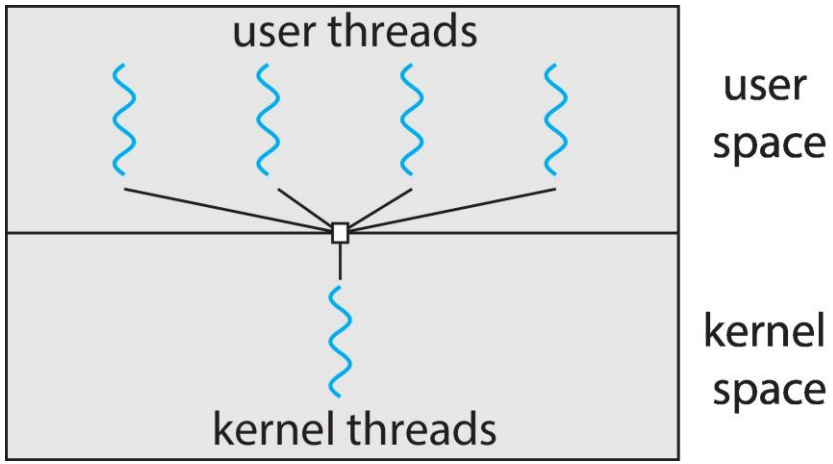
- การสนับสนุนการทำงานของเธรดจะขึ้นอยู่กับระดับของผู้ใช้จากเธรดของผู้ใช้ หรือจาก Kernel
- แต่เธรดของผู้ใช้จะสนับสนุนมากกว่า Kernel และสามารถควบคุมโดยไม่ต้องใช้การสนับสนุนจาก Kernel
- ส่วนเธรดของ Kernel นั้นจะสนับสนุนและควบคุมโดยตรงจากระบบปฏิบัติการ
- ในที่สุดแล้วเธรดของผู้ใช้และเธรดของ Kernel ก็ยังเชื่อมโยงกันอยู่ดี

Thread แบบ Many-to-One

- รูปแบบ Many-to-One เป็นรูปแบบที่ใช้เธรดสำหรับระบบปฏิบัติการ 1 หน่วย กับเธรดสำหรับผู้ใช้หลายหน่วย (Thread ผู้ใช้ผลัดกันทำงานใน Kernel)
- การจัดการเธรดจะอยู่ในพื้นที่ของผู้ใช้ซึ่งมีประสิทธิภาพ แต่ถ้าเธรดบล็อก System call โพรเซสทั้งหมดจะถูกบล็อกไปด้วย เนื่องจากจะมีเพียงเธรดเดียวเท่านั้นที่เข้าถึง Kernel ในเวลาหนึ่ง ๆ

Thread แบบ Many-to-One (2)

- เธรดหลาย ๆ เธรด ไม่สามารถรันขนานกันในระบบมัลติโพรเซสเซอร์ได้ ระบบที่ใช้รูปแบบนี้เช่น Green thread ซึ่งเป็นไลบรารีในโซลาริสทู (Solaris 2)

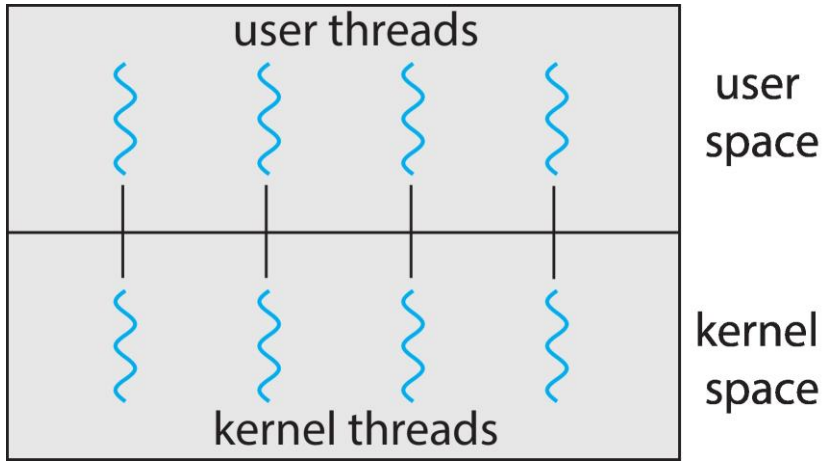


Thread แบบ One-to-One

- รูปแบบ One-to-One เป็นรูปแบบที่แต่ละเธรดสำหรับผู้ใช้ จะจับคู่กับเธรดสำหรับระบบปฏิบัติการ ในลักษณะ 1 ต่อ 1
- ทำให้สามารถทำงานพร้อมกันดีกว่าแบบ Many-to-One โดยยอมให้เธรดอื่นรันได้เมื่อเธรดบล็อก System call

Thread แบบ One-to-One (2)

- นอกจากนี้โมเดลนี้ยังยอมให้หลาย ๆ เธรดทำงานแบบขนานกันได้ในระบบมัลติโพรเซสเซอร์ได้อีกด้วย
- การสร้างเธรดสำหรับผู้ใช้ จำเป็นต้องสร้างเธรดสำหรับระบบปฏิบัติการที่สัมพันธ์กัน ระบบที่โมเดลมีข้อจำกัดที่จำนวน เธรดที่สนับสนุนในระบบได้ โมเดลนี้นำมาใช้ในระบบ เช่น ในระบบปฏิบัติการวินโดว์กับ Linux



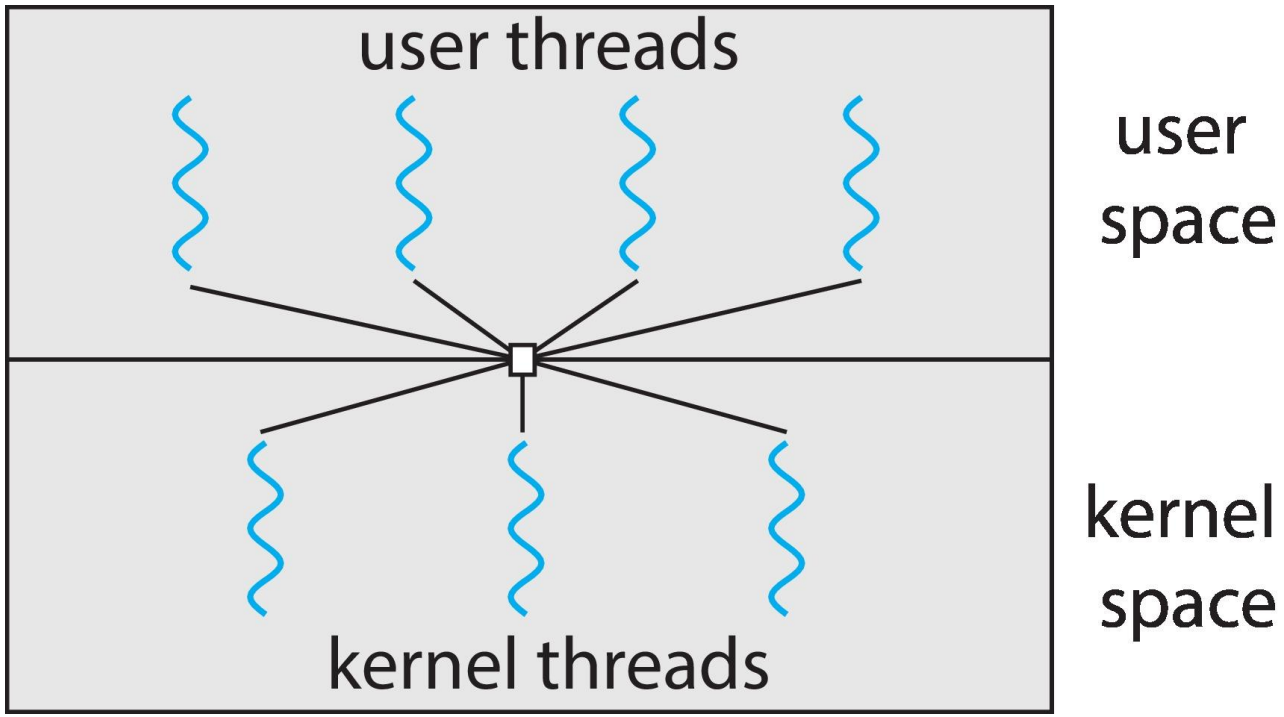
Thread แบบ Many-to-Many

- รูปแบบ Many-to-Many เป็นรูปแบบที่อาจจะมีจำนวนเธรดสำหรับผู้ใช้มากกว่าหรือเท่ากับจำนวนเธรดสำหรับระบบปฏิบัติการ
- ผู้พัฒนาสร้างเธรดสำหรับผู้ใช้ ได้ตามที่เขาต้องการ แต่จะไม่สามารถทำงานได้พร้อมกัน

Thread แบบ Many-to-Many (2)

- จำนวนเธรดสำหรับระบบปฏิบัติการ อาจจะเป็นตัวกำหนดแอปพลิเคชันเฉพาะหรือเครื่องเฉพาะ - ระบบปฏิบัติการสามารถเลือกจำนวนเธรดเคอร์เนลที่จะสร้างได้ ในจำนวนที่ตนเห็นว่าเหมาะสม ไม่ถูกบังคับให้ต้องสร้างเท่ากับจำนวนเธรดผู้ใช้
- เมื่อเธรดเกิดการบล็อก System call แล้ว Kernel จะจัดเวลาเพื่อนำเธรดอื่นขึ้นมารันก่อนก็ได้

Thread แบบ Many-to-Many (2)



Thread libraries

Thread Library 3 อย่างหลัก ๆ ที่ใช้ในปัจจุบันคือ

1) POSIX Pthreads
2) Win32
3) Java

Pthreads

Pthreads เป็นตัวพื้นฐานของ POSIX (IEEE 103.1C) เรียกได้ว่าเป็น API สำหรับการสร้างเธรดและสิ่งที่เกิดขึ้นในเวลาเดียวกัน เป็นตัวบ่งบอกถึงพฤติกรรมของเธรดโดยไม่ใช้เครื่องมือ การออกแบบระบบปฏิบัติการมักจะใช้เครื่องมือในงานที่ต้องการ ระบบปฏิบัติการส่วนใหญ่ใช้ Pthreads เป็นเครื่องมือ ไม่ว่าจะเป็นระบบปฏิบัติการโซราริส ลินุกส์ แมคโอเอส และยูนิกส์

Pthreads

1. pthreads_create() - สร้าง Thread
2. pthread_join( ) - เธรดพ่อจะรอคำสั่งสิ้นสุดการทำงาน
3. pthread_exit( ) - เธรดลูกจะสิ้นสุดการทำงานเมื่อมันเรียกฟังก์ชัน

การยกเลิก Thread

- เธรตที่จะถูกยกเลิกเรียกว่าเธรตเป้าหมาย (target thread)
- การยกเลิกทันที (Asynchronous Cancellation) หยุดการทำงานของเธรตทันที ผู้ใช้ไม่ต้องรอ (asynchronous) แต่เธรตอาจจะไม่มีโอกาสได้คืนทรัพยากรที่สำคัญ ไม่แนะนำให้ใช้
- การยกเลิกแบบถ่วงเวลา (Deferred Cancellation) เธรตจะตรวจเป็นระยะว่าผู้ใช้ต้องการให้ตนหยุดทำงานหรือไม่ เธรตจึงมีโอกาสที่จะคืนทรัพยากรที่สำคัญก่อนหยุดทำงาน แต่ในระหว่างนี้ผู้ใช้อาจจะต้องรอ