Assignment 1 – Producer / Consumer

ส่งไฟล์ code และรายงานภายใน วันที่ 3 พฤศจิกายน 2562 เวลา 18:00 น.

ตรวจในชั้นเรียน วันที่ 3 พฤศจิกายน 2562

คะแนนเต็ม 100 คะแนน

---

1. ปัญหา

สมมติว่านศ.มีบัฟเฟอร์แบบวนกลับ (circular buffer) ใช้สำหรับการทำงานแบบคิวหรือ First-in-first-out โดยมีขนาดที่เก็บข้อมูลได้ N รายการ (1 <= N <= 1000) ที่ถูกใช้งานร่วมกันด้วยเทรด(thread) มากกว่า 1 เทรด

1.1 **Basic Operations**: ให้เขียนฟังก์ชัน 2 ฟังก์ชันต่อไปนี้ เพื่อรองรับการทำงานแบบ circular buffer โดยสมมติว่า ณ เวลาใดเวลาหนึ่งจะมีเทรด (thread) ที่เรียกฟังก์ชันนี้ได้เพียงเทรดเดียวเท่านั้น

add_item – เพิ่มรายการเข้าไปในท้ายแถวของบัฟเฟอร์
remove_item – ลบรายการหัวแถวออกจากบัฟเฟอร์

1.2 **Concurrent Operations**: ให้นักศึกษาเขียนฟังก์ชัน **append** และ **remove** ซึ่งเรียกใช้ฟังก์ชันในข้อ 1.1 แต่รองรับการทำงานแบบหลายเทรดพร้อม ๆ กัน โดยจะมีคุณสมบัติดังต่อไปนี้

- 1.1.1 Mutually exclusive access to buffer: ณ เวลาใดเวลาหนึ่ง จะมีเพียงเทรดหนึ่งเทรดเท่านั้นที่สามารถเพิ่มหรือลบรายการในบัฟเฟอร์ได้
- 1.1.2 No buffer overflow: เทรดสำหรับ append จะสามารถเรียกใช้ add_item ได้ก็ต่อเมื่อบัฟเฟอร์ไม่เต็ม (ให้รอจนกว่าบัฟเฟอร์จะมีพื้นที่เหลือก่อนที่จะเรียก add_item)
- 1.1.3 No buffer underflow: เทรดสำหรับ remove จะสามารถเรียกใช้ remove_item ได้ก็ต่อเมื่อบัฟเฟอร์ไม่ว่าง (ถ้าว่างให้รอจนกว่าจะมีรายการในบัฟเฟอร์ก่อนเรียก remove_item)
- 1.1.4 No busy waiting: ห้ามใช้สปินล็อก(spinlock) ในการรอตรวจสอบเงื่อนไขของบัฟเฟอร์
- 1.1.5 No producer starvation: เทรดที่จะเรียก append มีเวลารอคอยที่จำกัด
- 1.1.6 No consumer starvation: เทรดที่จะเรียก remove มีเวลารอคอยที่จำกัด

1.2 **Buffer Benchmark**: ให้นักศึกษาเขียนโปรแกรม **buff** ที่ทำงานได้หลายเทรดพร้อม ๆ กัน เพื่อทดสอบประสิทธิภาพของ append และ remove โดยให้เทรดของการผลิตทั้งหมดสร้างรายการตามจำนวนที่ระบุใส่ในบัฟเฟอร์ ในขณะที่เทรดของการเรียกใช้ดึงรายการออกจากบัฟเฟอร์ และให้โปรแกรม buff นับจำนวนรายการที่ถูกดึงออกมาและเวลาที่ใช้ในการทำงาน โดย buff จะรับพารามิเตอร์ต่อไปนี้

- 1.2.1 จำนวนเทรดของการผลิต (producer thread - เทรดที่เรียก append)
- 1.2.2 จำนวนเทรดของการเรียกใช้ (consumer thread - เทรดที่เรียก remove)
- 1.2.3 ขนาดของบัฟเฟอร์ (N)
- 1.2.4 จำนวนรายการที่จะผลิตทั้งหมด (ไม่เกิน 1,000 ล้าน requests)

ตัวอย่าง ทดสอบโปรแกรมโดยให้มีเทรดผลิต 20 เทรด, มีเทรดเรียกใช้ 30 เทรด, บัฟเฟอร์ขนาด 1000 รายการ และผลิตรายการทั้งหมด 100000 รายการ

```
# buff 20 30 1000 100000

Producers 20, Consumers 30
Buffer size 1000
Requests 100000

Successfully consumed 95401 requests (95.4%)
Elapsed Time: 31.40 s
Throughput: 3038.25 successful requests/s
```

โดยงานนี้มีจุดประสงค์หลักให้อัตรา Throughput มีค่าสูงสุด

Assignment นี้จะเป็นงานกลุ่มโดยกลุ่มหนึ่งจะมีจำนวน 11-12 คน (4 กลุ่ม)

2. สิ่งที่ต้องส่ง

a. Design Document: รายงานการออกแบบ
   i. การออกแบบ เงื่อนไข วิธีการทำงาน และการพิสูจน์คุณสมบัติของ append
   ii. การออกแบบ เงื่อนไข และวิธีการทำงานของ remove
b. Program Source: URL เพื่อเข้าถึง Source code repository (e.g. GitHub) ของโปรแกรม
c. Presentation: การนำเสนอ/สาธิตการทำงานของโปรแกรม และการตอบคำถาม

3. เกณฑ์การให้คะแนน

a. 10 คะแนน รายงานที่มีผลการวิเคราะห์การออกแบบ และคำอธิบายแสดงถึงความถูกต้องของโปรแกรม
b. 50 คะแนน – การพัฒนาโปรแกรม (Implementation): Completion, Performance, Reliability
   i. Completion พิจารณาจากโปรแกรมทำงานตามที่กำหนดและมีผลลัพธ์ตามที่กำหนดในงาน
   ii. Performance จะคิดจากจำนวน Successful requests/ s เรียงตามลำดับแต่ละกลุ่มในชั้นเรียน โดยคิดฐานการคำนวณจาก 1000 ล้าน requests บน CPU ที่มี 4 logical cores
   iii. Reliability คิดจากอัตราส่วนจำนวน success requests ต่อจำนวน requests ทั้งหมด
c. 40 คะแนน – Presentation, Q&A (มีสิทธิถามใครก็ได้ในกลุ่มในการตอบคำถามหรือนำเสนอผลงาน)

4. อื่น ๆ

a. **Late submission** – นักศึกษาไม่สามารถส่งงานช้าเกินกว่าที่กำหนดในใบงานได้ หากส่งช้าเกินกำหนดจะได้คะแนนเป็นศูนย์สำหรับงานนี้
b. **Academic Integrity & Plagiarism** – สามารถพูดคุยแลกเปลี่ยนแนวทางต่างๆ ที่เหมาะสมกับการแก้ปัญหาได้ แต่ไม่สามารถแชร์โปรแกรมหรือคัดลอก source code จากเพื่อนหรือแหล่งข้อมูลภายนอกได้ หากมีเหตุการณ์หรือสถานการณ์ที่บ่งชี้ว่ากลุ่มใดไม่ได้พยายามทำงานด้วยตนเองจะได้คะแนนศูนย์ในงานนี้