01204223 OS Report Problem Set #1 (2023)

คำอธิบายทั่วไป

ให้นิสิตเลือกทำโจทย์ต่อไปนี้ โดยแต่ละโจทย์จะมีคะแนนเต็มไม่เท่ากัน ให้นิสิตเลือกทำกี่ข้อก็ได้จนคะแนนเต็มรวมครบ 100 คะแนน

ข้อที่	ลักษณะ	หัวข้อ	คะแนนสูงสุด
A	ปฏิบัติการ	OpenMP	40
B	ปฏิบัติการ	Memory Management	20
C	ปฏิบัติการ	Linux Kernel	20
D	ปฏิบัติการ	Timing I/O	20
E-H	ทฤษฎี	ตอบคำถาม	4 x 20

การตรวจและให้คะแนน (โดยรวม)

ผู้สอนจะตรวจไม่เกิน 6 ข้อ โดยเรียงลำดับจาก A ถึง H คะแนนที่ได้จะคำนวณโดยเอาข้อที่ "ถูกหักคะแนน" น้อยที่สุดมารวมกันตามลำดับ จนคะแนนเต็มรวมมากกว่าหรือเท่ากับ 100 คะแนน

กรณีที่ทำตามข้างต้นแล้วคะแนนที่นิสิตได้มากกว่า 100 จะถูกปัดลงเหลือ 100 คะแนน

คะแนนที่ได้นี้จะถูกเทียบเป็นคะแนนรายวิชาไม่เกินร้อยละ 10

การส่งงาน (โดยรวม)

ให้นำงานทั้งหมดที่อยู่ใน “ส่วนรายงาน” ใส่ในไฟล์ PDF เพียงไฟล์เดียวเท่านั้น ไฟล์ PDF นี้เป็นไฟล์เดียวที่สามารถแนบส่งบนระบบ Google Classroom ได้ (ห้ามส่งไฟล์อื่นใดผ่าน Google Classroom โดยเด็ดขาด)

ในส่วนของโปรแกรม ให้ทำตามคำสั่งของแต่ละข้อ และแนบ URL ของ repository หรือวิดีโอทั้งหมดลงในตัวรายงานด้วย (PDF) ทั้งนี้ ข้อมูลเพิ่มเติมใดที่ไม่มี URL แนบในรายงาน จะไม่ถือว่าเป็นส่วนหนึ่งของรายงาน และจะไม่ได้รับการพิจารณาตรวจ

กำหนดส่ง 2023-08-20 23:59:00 (ICT)

นิสิตควรตรวจสอบไฟล์ที่ส่งมาแล้ว ว่าสามารถดาวน์โหลดและเปิดได้อย่างถูกต้องหรือไม่

A: การใช้ OpenMP (40 คะแนน) (C language only)

สื่อเพิ่มเติมที่ต้องใช้: GitHub Classroom https://classroom.github.com/a/hkUUYcc_ (URL ลงท้ายด้วย underscore)

ให้นิสิตเขียนโปรแกรมหาค่าเฉลี่ยของตัวเลข โดยข้อมูลนำเข้าจะประกอบด้วยจำนวนเต็ม $100 \leq N \leq \{500, 50000, 500000\}$ ในบรรทัดแรก และในบรรทัดต่อไปจำนวน N บรรทัด จะมีจำนวนเต็ม 1 จำนวน $0 \leq n_i \leq 255$

นิสิตจะต้องใช้ OpenMP เพื่อแบ่งงานระหว่างเธรด (threads) คำนวณและแสดงค่าเฉลี่ยของจำนวนทั้งหมดที่นำเข้า (ปัดเศษลง) โดยในส่วนของการรวบรวมตัวเลขขั้นต้นจะต้องใช้สภาพแวดล้อม `#pragma omp parallel` เท่านั้น **แนะนำให้รวมเลขทั้งหมดก่อนแล้วค่อยหารในขั้นตอนสุดท้าย**

นอกจากนี้ นิสิต**ต้องเขียนรายงาน**ประมาณ 150-200 คำ อธิบายแนวคิดและอัลกอริทึมที่ใช้ให้ชัดเจนในรายงาน

ตัวอย่างข้อมูลนำเข้าและส่งออก

ข้อมูลนำเข้า	ข้อมูลส่งออก
5 1 2 3 4 5	3
15 8 5 34 72 23 75 0 0 0 23 111 66 45 44 44	36

มาตรฐานการให้คะแนน

ข้อ	คะแนน (จาก 40)	ประเด็น
1	Required	นิสิตใช้ OpenMP ทำขั้นตอน summation เบื้องต้น
2	Required	รายงานและโค้ดสอดคล้องกัน
3	20	ผ่าน Test Cases ทั้งสองเคสใน GitHub Classroom (ตรวจอัตโนมัติ)
4	10	ผ่าน Test Case พิเศษ (ลับ ตรวจด้วยมือ)
5	5	วินัยการใช้หน่วยความจำและคุณภาพโค้ด
6	5	คุณภาพการทำรายงาน
7	4 (Bonus)	เทคนิคการเขียนโปรแกรมยอดเยี่ยม

(คะแนนรวมจะได้ไม่เกิน 40 คะแนน)

คะแนน = Required หมายความว่าไม่มีคะแนนในตัวมันเอง แต่หากไม่ทำตามจะถูกหักคะแนนหรือไม่รับตรวจ

การส่งงาน

ส่วนโปรแกรม: ให้ใช้สภาพแวดล้อมการทำงาน GitHub Classroom ส่ง (push) ไฟล์ภาษาซีเพียงไฟล์เดียว ให้ตั้งชื่อไฟล์ว่า `averagepx.c` (ห้ามตั้งชื่ออื่น มิฉะนั้น autograder จะไม่ตรวจให้) ให้นิสิตทำงานทั้งหมดด้วยตัวเอง และมีการทำเวอร์ชั่นซอฟท์แวร์อย่างสม่ำเสมอ เพื่อเป็นหลักฐานว่าได้ทำงานเองจริง

ทั้งนี้ การคอมไพล์จะคอมไพล์ด้วยคำสั่ง `gcc -fopenmp averagepx.c -o averagepx`

ระบบจะไม่ตรวจไฟล์อื่น แต่ไม่แนะนำให้เพิ่มไฟล์ที่ไม่จำเป็นเข้ามาใน repository เพื่อป้องกันการเกิดปัญหาอื่นๆ

ส่วนรายงาน: ให้พิมพ์รายงานใส่ในไฟล์รายงาน รวมกับข้ออื่นๆ และในรายงานขอให้แนบลิงค์ GitHub Repository ที่นิสิตเก็บโค้ดไว้ด้วยเพื่อเป็นหลักฐานเพิ่มเติม โดยห้ามแก้ไข Repository นี้หลังจากหมดเวลาส่ง และห้ามลบ Repository นี้ก่อนเกรดออก

หมายเหตุ: กรณีนิสิตต้องการใช้ Fortran ขอให้ติดต่อผู้สอน

B: Memory Management (20 คะแนน) (ภาษาใดก็ได้ที่ใช้ pointer ได้)

ให้นิสิตเขียนโปรแกรม 1 โปรแกรม ที่สามารถพิสูจน์ได้ว่า stack โตลง และ heap โตขึ้น โดยในโปรแกรมจะต้องประกอบไปด้วย:

- Function call stack ที่ซ้อนกันอย่างน้อย 3 ชั้น (อาจใช้ recursive function หรือเรียกฟังก์ชันซ้อนกันก็ได้ แต่ต้องแสดงให้เห็น address ของบางสิ่งใน call stack นั้น เช่น local variables)
- `malloc` อย่างน้อย 3 ครั้ง (หรือการจองหน่วยความจำในภาษาอื่น)
- Global variables อย่างน้อย 3 ตัว
- BONUS: นิสิตสามารถทำให้โปรแกรมเกิด stack overflow ได้

ให้นิสิตแสดงให้เห็นผ่านทาง output ว่า ในแต่ละครั้งที่มีการเรียกใช้ฟังก์ชัน กำหนดตัวแปร หรือจองหน่วยความจำ แอดเดรสที่เกี่ยวข้องนั้นเปลี่ยนไปอย่างไรบ้าง

มาตรฐานการให้คะแนน

ข้อ	คะแนน (จาก 20)	ประเด็น
1	Required	ทำตามรายการความต้องการข้างต้นครบถ้วน
2	Required	รายงานและโค้ดสอดคล้องกัน
3	12	เขียนรายงานชัดเจนและรู้เรื่อง
4	8	สร้างแผนภาพแสดงโครงสร้างหน่วยความจำ
5	2 (Bonus)	โปรแกรมเกิด stack overflow และนิสิตสามารถอธิบายปรากฏการณ์ได้ว่าเกิดขึ้นเมื่อใด เพราะอะไร

(คะแนนรวมจะได้ไม่เกิน 20 คะแนน)

การส่งงาน

ส่วนโปรแกรม: ให้ทำ GitHub Repository หรือ Gist ไว้ในพื้นที่ส่วนตัวของนิสิต แล้วแนบ URL ไว้ในรายงาน หากนิสิตไม่ได้ใช้ภาษาซี ให้เขียนวิธีการคอมไพล์และรันโปรแกรมไว้ใน README ของ repository ด้วย

ส่วนรายงาน: ให้เขียนรายงานประมาณ 150 คำ อธิบายว่าโปรแกรมทำงานอย่างไร และแสดงให้เห็นถึงการทำงานของหน่วยความจำได้อย่างไร ให้แนบ URL ของ repository และ output ตัวอย่างด้วย

C: Linux Kernel (20 คะแนน)

คำเตือน: แนะนำให้สร้าง Virtual Machine ใหม่ เพื่อทำแล็บนี้โดยเฉพาะ ผู้สอนจะไม่รับผิดชอบต่อความเสียหายใดๆ ที่เกิดจากการไม่เชื่อฟังคำเตือนนี้

ให้นิสิตคอมไพล์ Linux Kernel ที่มีเวอร์ชันแตกต่างจากที่ใช้อยู่ แล้วบูตเข้าเคอร์เนลใหม่นั้น

ความต้องการพื้นฐานที่ต้องทำได้/สิ่งที่ต้องมีในรายงาน

- แสดงเวอร์ชันของเคอร์เนลที่มีอยู่เดิมด้วยคำสั่ง `uname -r`
- คอมไพล์เคอร์เนลได้อย่างถูกต้อง (ให้แสดงภาพบางส่วนจากขั้นตอนการทำ `menuconfig`)
- แสดงหน้าบูตโหลดเดอร์ (GRUB)
- แสดงเวอร์ชันของเคอร์เนลใหม่หลังจากบูตเข้าเคอร์เนลใหม่แล้วโดยใช้คำสั่งข้างต้น
- หน้าจอ shell ต้องเห็น username ด้วย และห้ามเป็น username พื้นฐานของระบบปฏิบัติการ หาก shell ที่ใช้ไม่แสดงชื่อผู้ใช้ ให้ใช้คำสั่ง `whoami` เพื่อแสดงชื่อผู้ใช้ด้วย

มาตรฐานการให้คะแนน

ข้อ	คะแนน (จาก 20)	ประเด็น
1	10	ทำตามความต้องการพื้นฐานได้ครบถ้วน
2	6	เขียนอธิบายครบถ้วน
3	4	ภาพประกอบชัดเจน

การส่งงาน

ส่วนโปรแกรม: ไม่มี แต่หากใช้ VM ขอให้เก็บไฟล์ image ของ VM นั้นไว้เป็นหลักฐาน

ส่วนรายงาน: เขียนเล่าประสบการณ์ของตัวเอง โดยใช้ภาษาตามความรู้สึกของตัวเอง บอกเล่าเหตุการณ์และสิ่งที่ทำ ตลอดจนขั้นตอนอย่างละเอียด (แต่ขอว่าทั้งเนื้อหาและภาพไม่ควรเกิน 3 หน้า)

D: Timing I/O (20 คะแนน) (ใช้ภาษาอะไรก็ได้)

ให้นิสิตแสดงให้เห็นว่า I/O เป็นการทำงานที่ช้า โดยอาศัยการเขียนโปรแกรมเพื่อแสดงเวลาในการทำงาน ควบคู่กับคำสั่งจับเวลา (เช่น `time` ในลินุกซ์) หรืออาจใช้เครื่องมือเช่น profiler ในการแสดงให้เห็นก็ได้

มาตรฐานการให้คะแนน

ข้อ	คะแนน (จาก 20)	ประเด็น
1	10	ทำตามความต้องการพื้นฐานได้ครบถ้วน
2	6	เขียนอธิบายครบถ้วน
3	4	ภาพประกอบชัดเจน

การส่งงาน

ส่วนโปรแกรม: ให้ทำ GitHub Repository หรือ Gist ไว้ในพื้นที่ส่วนตัวของนิสิต แล้วแนบ URL ไว้ในรายงาน หากนิสิตไม่ได้ใช้ภาษาซีหรือไพธอน ให้เขียนวิธีการคอมไพล์และรันโปรแกรมไว้ใน README ของ repository ด้วย

ส่วนรายงาน: เขียนรายงานประมาณ 100-150 คำ และให้แนบ URL ของ repository และ output ตัวอย่างด้วย

E-H: ตอบคำถาม (Up To 4 x 20 คะแนน)

ให้นิสิตตอบคำถามต่อไปนี้โดยละเอียด จะเลือกทำกี่ข้อก็ได้ คะแนนเต็มข้อละ 20 คะแนน คำตอบควรจะยาวประมาณข้อละ 200-250 คำ และให้ใส่ภาพประกอบได้เท่าที่จำเป็น กรณีใช้เนื้อหาหรือรูปภาพจากภายนอกต้องอ้างอิงด้วย

E: POSIX

POSIX คืออะไร? ระบบปฏิบัติการจะต้องเป็นอย่างไรจึงจะถือว่า POSIX-Compliant? มีวิธีการอย่างไรที่ผู้ใช้ระบบปฏิบัติการวินโดวส์จะเข้าถึงสภาพแวดล้อมการทำงานที่เป็น POSIX-Compliant? วิธีใดที่นิสิตรู้สึกว่าเหมาะสมสำหรับนิสิต (หากคุณใช้แมคหรือลินุกซ์อยู่แล้ว: ลองยกตัวอย่างคนใกล้ตัว)?

F: Memory Management

ให้ยกตัวอย่างโปรแกรมที่มาจากภาษาโปรแกรมอื่น (ที่ไม่ใช่ภาษาซี) ภาษานี้มีการจัดการหน่วยความจำอย่างไร? มีโครงสร้างการทำงานอย่างไร? นิสิตคิดว่าภาษานี้เหมาะสมกับการทำงานแบบใด?

G: My First Operating System

หากนิสิตต้องทำโปรเจคส่งอาจารย์เป็นระบบปฏิบัติการของตัวเอง นิสิตจะทำอย่างไร? ใช้ภาษาอะไร? มีแนวคิดหรือหลักการออกแบบอย่างไร? อธิบายให้ละเอียด และอาจวาดแผนภาพสถาปัตยกรรมที่ต้องการใช้หรือสร้างด้วย

H: Web Environment Integrity (WEI)

WEI คืออะไร? WEI (จะ)มีประโยชน์และโทษ มีข้อดีข้อเสียอย่างไร? นิสิตรู้สึกอย่างไรกับ WEI?

มาตรฐานการให้คะแนน

ข้อ	คะแนน (จาก 20)	ประเด็น
แต่ละข้อ	10	ความถูกต้องและคุณภาพการอธิบาย
	6	ภาษามีความชัดเจน
	4	ภาพประกอบชัดเจน
	2 (Bonus)	ภาพประกอบวาดเอง มีความชัดเจน อธิบายเพิ่มความเข้าใจได้ดี

ทั้งนี้ คะแนนรวมแต่ละข้อจะได้ไม่เกิน 20 คะแนน

การส่งงาน

ให้พิมพ์เนื้อหาและแนบภาพทั้งหมดในไฟล์รายงาน