ใบงานที่ 5

เรื่อง Threads Creation and Execution

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

จัดทำโดย

นางสาวรัชนีกร เชื้อดี 65543206077-1

ใบงานนี้เป็นส่วนหนึ่งของรายวิชา ระบบปฏิบัติการ

หลักสูตรวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ประจำภาคที่ 2 ปีการศึกษา 2566

ใบงานที่ 5

Threads Creation and Execution

ลำดับขั้นการทดลอง

1. ให้เขียนโปรแกรมตามโค้ดตัวอย่าง บนระบบปฏิบัติการ CentOS และแสดงผลทดลอง
2. อธิบายการทำงานของโปรแกรม
3. สรุปผลการทดลอง
4. ส่งงานใน Microsoft Team เป็นไฟล์ PDF

บันทึกผลการทดลอง

ข้อ 1. ทำการสร้าง Thread ขึ้นมา จากโปรแกรมคำสั่งที่กำหนดให้ ต่อไปนี้ พร้อมกับอธิบายคำสั่งที่กำหนดไว้

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *print_message_function( void *ptr );
main()
{
    pthread_t thread1, thread2;
    char *message1 = "Thread 1";
    char *message2 = "Thread 2";
    int iret1, iret2;

    iret1 = pthread_create( &thread1, NULL, print_message_function, (void*) message1);
    iret2 = pthread_create( &thread2, NULL, print_message_function, (void*) message2);

    pthread_join ( thread1, NULL);
    pthread_join ( thread2, NULL);

    printf("Thread 1 returns: %d\n",iret1);
    printf("Thread 2 returns: %d\n",iret2);
    exit(0);
}
void *print_message_function( void *ptr )
{
    char *message;
    message = (char *) ptr;
    printf("%s \n",message);
}
```

บรรทัดที่ 1: โหลดไลบรารีจำเป็นสำหรับการทำงานของโปรแกรม

บรรทัดที่ 2-4: ประกาศฟังก์ชัน print_message_function ที่รับตัวแปร ptr เป็นอาร์กิวเมนต์และส่งคืนค่าเป็นตัวชี้แบบ void *

บรรทัดที่ 5-12: บล็อคโค้ดหลักของโปรแกรม

บรรทัดที่ 13-17: บล็อคโค้ดของฟังก์ชันภายในเธรด

ผลลัพธ์

```
[lookplarc@localhost ~]$ ./pthread1
Thread 1
Thread 2
Thread 1 returns: 0
Thread 2 returns: 0
```

โปรแกรมนี้สร้างสองเธรด โดยแต่ละเธรดจะพิมพ์ข้อความของตัวเอง จากนั้นรอให้ทั้งสองเธรดเสร็จสิ้นจึงพิมพ์ผลลัพธ์ของการสร้างเธรดและออกจากโปรแกรม

ข้อ 2. นำ Example: pthread1.c มาทำการ เพิ่ม ฟังก์ชั่นการทำงาน เข้าไป ใน pthread1.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *print_message_function( void *ptr );
int count = 0;
int main()
{
    pthread_t thread1, thread2, thread3;
    char *message1 = "Thread 1";
    char *message2 = "Thread 2";
    char *message3 = "Thread 3";
    int iret1, iret2, iret3;

    iret1 = pthread_create( &thread1, NULL,print_message_function, (void*) message1);
    iret2 = pthread_create( &thread2, NULL,print_message_function, (void*) message2);
    iret3 = pthread_create( &thread3, NULL,print_message_function, (void*) message3);

    pthread_join ( thread1, NULL);
    pthread_join ( thread2, NULL);
    pthread_join ( thread3, NULL);
    return 0;
}
void *print_message_function( void *ptr )
{
    char *message;
    message = (char *) ptr;
    printf("%s pid = %d tid = %u\n", message,getpid(),(unsigned int)pthread_self());

    int i = 0;
    for (i=0; i<10; i++) {
        sleep (1);
        printf("%u -> %d count = %d\n",(long)pthread_self(),i,count);
        count++;
    }
}
```

บรรทัดที่ 1-3: นำเข้าไลบรารีที่จำเป็น: stdio.h สำหรับการพิมพ์ข้อความ, stdlib.h สำหรับการจัดการหน่วยความจำ, pthread.h สำหรับการสร้างและควบคุมเธรด

บรรทัดที่ 4: ประกาศฟังก์ชัน print_message_function ที่รับ void* ptr เป็นอาร์กิวเมนต์และส่งคืน void*

บรรทัดที่ 5: นิยามตัวแปร count แบบ int เริ่มต้นเป็น 0 (ใช้สำหรับนับรอบการวนลูปภายในเธรด)

บรรทัดที่ 6-12: บล็อคโค้ดของฟังก์ชัน main

- `pthread_t thread1, thread2, thread3`: ประกาศตัวแปร `pthread_t` สามตัวเพื่อเก็บ ID ของเธรดที่จะสร้าง
- `char *message1, message2, message3`: ประกาศตัวแปรเก็บข้อความสำหรับเธรดแต่ละตัว
- `int iret1, iret2, iret3`: ประกาศตัวแปรเก็บผลลัพธ์ของการสร้างเธรด
- `pthread_create`: สร้าง 3 เธรดโดยส่ง `print_message_function` เป็นฟังก์ชันที่จะเรียกใช้ภายในเธรด
- `&threadX`: ส่งตัวแปรเก็บ ID ของเธรดใหม่ไปที่ `pthread_create`
- `NULL`: ไม่มีค่าเริ่มต้นสำหรับชื่อเธรด
- `(void*) messageX`: ส่งข้อความของแต่ละเธรดเป็นอาร์กิวเมนต์
- `pthread_join`: รอจนกระทั่งเธรดทั้งสามเสร็จสิ้นการทำงาน
- `return 0`: ออกจากโปรแกรมโดยส่งคืนค่า 0

บรรทัดที่ 13-20: บล็อคโค้ดของฟังก์ชัน print_message_function

- `char *message`: ประกาศตัวแปรเก็บข้อความที่จะพิมพ์
- `message = (char *) ptr`: นำข้อมูลจากตัวแปร `ptr` ซึ่งเป็นอาร์กิวเมนต์มาเก็บไว้ใน `message`
- `printf`: พิมพ์ข้อความ, pid ของกระบวนการ, และ tid ของเธรดปัจจุบัน
- `for` loop: วนลูป 10 รอบ
    - `sleep(1)`: รอ 1 วินาทีในแต่ละรอบ
    - `printf`: พิมพ์ tid ของเธรด, หมายเลขของรอบ, และค่าของ `count`
    - `count++`: เพิ่มค่า count 1 หลังจากพิมพ์แต่ละรอบ

ผลลัพธ์

```
Thread 1 pid = 5274 tid = 3506591488
Thread 2 pid = 5274 tid = 3498198784
Thread 3 pid = 5274 tid = 3489806080
3506591488 -> 0 count = 0
3498198784 -> 0 count = 1
3489806080 -> 0 count = 2
3506591488 -> 1 count = 3
3498198784 -> 1 count = 4
3489806080 -> 1 count = 5
3506591488 -> 2 count = 6
3498198784 -> 2 count = 7
3489806080 -> 2 count = 8
3506591488 -> 3 count = 9
3498198784 -> 3 count = 10
3489806080 -> 3 count = 11
3506591488 -> 4 count = 12
3498198784 -> 4 count = 13
3489806080 -> 4 count = 14
3506591488 -> 5 count = 15
3498198784 -> 5 count = 16
3489806080 -> 5 count = 17
3506591488 -> 6 count = 18
3498198784 -> 6 count = 19
3489806080 -> 6 count = 20
3506591488 -> 7 count = 21
3498198784 -> 7 count = 22
3489806080 -> 7 count = 23
3506591488 -> 8 count = 24

3498198784 -> 8 count = 25
3489806080 -> 8 count = 26
3506591488 -> 9 count = 27
3498198784 -> 9 count = 28
3489806080 -> 9 count = 29
```

โปรแกรมนี้สร้าง 3 เธรด แต่ละเธรดจะพิมพ์ข้อความของตัวเอง, pid และ tid ของตัวเอง วนลูป 10 รอบ พิมพ์หมายเลขของรอบและ count ที่เพิ่มขึ้นทุกวินาที รอจนกระทั่งเธรดทั้งสามเสร็จสิ้นจึงออกจากโปรแกรม

สรุปผลการทดลอง

จากผลการทดลองทั้ง 2 โปรแกรม สรุปได้ว่าโปรแกรมที่สร้างเธรดหลายตัวสามารถทำงานสำเร็จได้โปรแกรมที่มีการแข่งขันข้อมูล (data race) อาจส่งผลให้ผลลัพธ์ไม่แน่นอน ดังนั้น จึงควรระมัดระวังในการจัดการข้อมูลร่วมกันระหว่างเธรด โดยเฉพาะอย่างยิ่งข้อมูลที่สามารถเปลี่ยนแปลงได้

สื่อ / เอกสารอ้างอิง

อาจารย์ปิยพล ยืนยงสถาวร: เอกสารประกอบการสอน 5 Threads (เธรด)