ใบงานที่ 5

เรื่อง Threads Creation and Execution

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

จัดทำโดย

นาย กวีวัธน์ กาญจน์สุพัฒนากุล 65543206003-7

ใบงานนี้เป็นส่วนหนึ่งของรายวิชา ระบบปฏิบัติการ

หลักสูตรวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ประจำภาคที่ 2 ปีการศึกษา 2566

Lab 5 : Threads Creation and Execution

Due December 20, 2023 11:59 PM

Instructions

ปฏิบัติการทดลองตามใบงาน

- แสดงผลการทำงานโปรแกรมในเวลาคาบปฏิบัติ
- จัดทำเอกสารใบงานการทดลองส่งตามกำหนด

Reference materials

LAB_5_Threads Creation and Execution.pdf ···

ตัวอย่างที่ 1

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
void *print_message_function( void *ptr );
/* struct to hold data to be passed to a thread this shows how
multiple data items can be passed to a thread */
main()
{
        pthread_t thread1, thread2; /* thread variables */
        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); /* Start waiting for
thread1. */
        pthread_join( thread2, NULL); /* Start waiting for
thread2. */

        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);
}
```

ผลลัพธ์

```
[Kaweewat@localhost lab5]$ ./ex1
Thread 1
Thread 2
Thread 1 returns: 0
Thread 2 returns: 0
```

ขั้นตอนการทำงาน

1. โปรแกรมจะพิมพ์ข้อความเข้าไปใน Print_message_function ที่ threadแต่ละตัวเรียกใช้
2. Pthread_create จะทำงานที่ฟังก์ชั่น print_message_function
3. pthread รอการทำงานของ thread ทั้ง 2 ทำงานจนเสร็จสิ้น ก่อนที่จะประมวลผลใน main
4. นอกจากนั้น มีการเรียกใช้ฟักค์ชั่น print_message_function เพื่อพิมพ์ข้อความ message

ตัวอย่างที่ 2

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
void *print_message_function( void *ptr );
int count = 1;
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 = 1;
       for (i=0; i<10; i++){
               sleep (1);
               printf("%u -> %d count = %d\n",(long)pthread_self(),i,count);
               count++;
       }
}
```

ผลลัพธ์

```
[Kaweewat@localhost lab5]$ ./ex2
Thread 2 pid = 5101 tid = 593372928
Thread 3 pid = 5101 tid = 584980224
Thread 1 pid = 5101 tid = 601765632
593372928 -> 0 count = 0
584980224 -> 0 count = 1
601765632 -> 0 count = 2
593372928 -> 1 count = 3
584980224 -> 1 count = 4
601765632 -> 1 count = 5
593372928 -> 2 count = 6
584980224 -> 2 count = 7
601765632 -> 2 count = 8
593372928 -> 3 count = 9
584980224 -> 3 count = 10
601765632 -> 3 count = 11
593372928 -> 4 count = 12
584980224 -> 4 count = 13
601765632 -> 4 count = 14
593372928 -> 5 count = 15
584980224 -> 5 count = 16
601765632 -> 5 count = 17
593372928 -> 6 count = 18
584980224 -> 6 count = 19
601765632 -> 6 count = 20
593372928 -> 7 count = 21
2937042688 -> 7 count = 22
2928649984 -> 7 count = 23
2920257280 -> 7 count = 24
2937042688 -> 8 count = 25
2928649984 -> 8 count = 26
2920257280 -> 8 count = 27
2937042688 -> 9 count = 28
2928649984 -> 9 count = 29
2920257280 -> 9 count = 30
```

ขั้นตอนการทำงาน

1. สร้างสาม threads ด้วย pthread_create, แต่ละ thread จะทำงานใน print_message_function ด้วยข้อความที่ต่างกัน ("Thread 1", "Thread 2", "Thread 3")
2. รอให้ทุก thread ทำงานเสร็จสิ้นด้วย pthread_join
3. print_message_function ทำหน้าที่พิมพ์ข้อความพร้อมกับข้อมูลอื่น ๆ
4. ในลูป for ภายใน print_message_function, แต่ละ thread จะทำงานเป็นเวลา 10 รอบ (1 รอบต่อวินาที), แสดงผลลัพธ์พร้อมกับการนับตัวแปร count ที่เพิ่มขึ้นทุกครั้งที่ทำงาน

สรุปผลการทดลอง

Threads คือการแบ่งหน้าที่ให้การทำงานให้กับ cpu เพื่อทำให้การทำงานได้ไวยิ่งขึ้น การใช้ Threads มีข้อดีในด้านประสิทธิภาพในด้านการทำงานและการจัดการทรัพยากร ได้ดีกว่า Process และ multitasking หรือการทำงานแบบหลาย Process