2018008331 컴퓨터소프트웨어학부 박민경

운영 체제 HW#4

제출 일자 : 2020/04/18

A. 과제 A

1. 프로그램 설명

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

#define ARGUMENT_NUMBER 20

//Variable to store results for all threads
long long result=0;

void *ThreadFunc(void *n)
{
    long long i;
    long long number = *((long long *)n);

    //Variable to store results for individual threads
    long long tResult = 0;
    printf("number=%lld\n", number);

    //Store values in different tResult for each thread
    for (i=0;i<25000000;i++)
        tResult +=number;
    }

    //Store the sum of the results of all threads
    result +=tResult;
}

int main(void)
{

    pthread_t threadID[ARGUMENT_NUMBER];
    long long argument[ARGUMENT_NUMBER];
    long long i;

    //Create ARGUMENT_NUMBER threads
    for (i=0;i<ARGUMENT_NUMBER;i++)
    {
        argument[i] = i;
        pthread_create(&(threadID[i]), NULL, ThreadFunc, (void *)&argument[i]);
    }
    printf("Main Thread is waiting for Child Thread!\n");

    //Wait for ARGUMENT_NUMBER threads to terminate
    for (i=0;i<ARGUMENT_NUMBER;i++)
    {
        pthread_join(threadID[i], NULL);
    }
    printf("result=%lld\n", result);
    return 0;
}
```

Ubuntu Software

- 함수

  ThreadFunc 함수는 각 스레드가 실행하는 스레드 함수이다. 인자로 void* 형 변수 n을 전달받아 long long으로 형변환하여 number에 저장하고, 스레드마다 number값을 25000000번 더해 지역변수 tResult에 저장한다. 그리고 그 결과를 전역변수 result에 더해 저장한다.

  pthread_create 함수는 스레드를 생성한다. 스레드 식별자, 스레드 특성, 실행할 스레드 함수, 스레드 함수의 매개변수를 순서대로 인자로 갖는다. 이 프로그램에서는 pthread_t 배열로 threadID 변수를 선언하고, for문을 돌며 threadID의 i번째 값을 첫번째 인자로 넘겨주었다. ThreadFunc 함수를 분기시켜 실행하고, ThreadFunc 함수의 매개변수로 (void*)&argument[i] 값을 넘겨준다.

  pthread_join 함수는 스레드 종료를 기다린다. 이 프로그램에서는 for문을 돌며 threadID 변수의 i번째 값을 인자로 넘겨주어 모든 스레드의 종료를 기다리며, join이 되는 순간 스레드는 모든 자원을 반납한다.

- 프로그램 구조

  20개의 스레드를 새로 만들고, 스레드 함수가 실행되면 각각의 스레드가 생성된 순서에 따라 그 숫자를 number에 저장하고, print한다. 스레드는 병렬적으로 실행되기 때문에, 각 스레드 별로 연산 결과를 담기 위해 지역변수 tResult를 선언한 후, for문을 돌며 tResult에 number값을 25000000 더한 결과를 최종적으로 전역변수인 result에 더해 저장한다. 최종적으로 모든 스레드의 결과값의 합이 result에 저장되고, main문에서 모든 스레드의 종료를 기다린 후 result를 출력한다.

2. Time 결과값 설명

```
mingyeong@mingyeong-VirtualBox:~/operatingSystem/HW#4$ time ./multithread_practi
ce_solution
Main Thread is waiting for Child Thread!
number=6
number=7
number=8
number=5
number=9
number=10
number=4
number=11
number=12
number=13
number=3
number=14
number=15
number=16
number=17
number=18
number=19
number=2
number=1
number=0
result=4750000000

real    0m0.992s
user    0m0.808s
sys     0m0.000s
```

<multithread_practice_solution>

real: 실제 경과 시간 (총 수행 시간)

user: 프로세스 내 user mode에서 소비된 CPU 시간

sys: 프로세스 내 kernel mode에서 소비된 CPU 시간

보통 OS가 해당 프로그램뿐만 아니라 다른 작업도 수행하므로, real>=user+sys이다. 하지만 멀티스레드에서는 user time과 sys time에 스레드 개수를 곱하여 출력하므로, real time보다 더 커지는 경우가 있을 수 있다.

3. 시간 차이가 나는 이유 설명

multithread_practice (주어진 코드)에서는 프로세스 내에서 ThreadFunc을 순차적으로 20번 호출하여 실행했다. 하지만 multithread_practice_solution (수정코드)에서는 멀티스레드를 적용하였기 때문에 프로세스 내에서 ThreadFunc 함수가 병렬적으로 20번 실행될 수 있어, 멀티스레드가 적용된 프로그램의 수행시간이 짧았다.

B. 과제 B

1. Mutex 예비레포트

mutex: mutual exclusion(상호 배제)

-> critical section을 가진 스레드들의 running time이 서로 겹치지 않게, 각각 단독으로 실행되게 하는 기술. 잠금 형식으로 이루어진다. 스레드가 잠금을 얻어야 critical section에 진입할 수 있고, critical section에서 빠져나오면 잠금을 되돌려줘서 다른 스레드가 그것을 얻을 수 있도록 한다.

- critical section: 프로그램 상에서 동시에 실행될 경우 문제를 일으킬 수 있는 부분

만약 어느 스레드에서 critical section을 실행하고 있으면 다른 스레드들은 그 critical section에 접근할 수 없고, 앞의 스레드가 critical section을 벗어나기를 기다려야 한다.

- mutex 메커니즘의 특징

- **Atomicity** - mutex 잠금(lock)는 최소단위 연적(atomic operation)으로 작동한다. 이 말의 뜻은 하나의 쓰레드가 mutex 를 이용해서 잠금을 시도하는 도중에 다른 쓰레드가 mutex 잠금을 할 수 없도록 해준다는 뜻이다. 한 번에 하나의 mutex 잠금을 하도록 보증해준다.

- **Singularity** - 만약 스레드가 mutex 잠금을 했다면, 잠금을 한 쓰레드(:12)가 mutex 잠금을 해제하기 전까지 다른 어떠한 쓰레드도 mutex 잠금을 할 수 없도록 보증해준다.
-
- **Non-Busy Wait** – 바쁜 대기 상태에 놓이지 않는다는 뜻으로, 하나의 쓰레드가 mutex 잠금을 시도하는데 이미 다른 쓰레드가 mutex 잠금을 사용하고 있다면 이 쓰레드는 다른 쓰레드가 lock 을 해제하기전까지 해당 지점에 머물러 있으며 이동안은 어떠한 CPU 자원도 소비하지 않는다.