Grid Scheduling and Multithreading

Contents

- Introduction to Multithreading Programming
- Data Race Problem
- Critical Section

Text & Reference

- The Art of Multiprocessor Programming, Morgan Kaufmann, 2012.

- N. Jung, Multi-threaded Parallel Programming.

Related Programming

- C programming / C++ Programming
- Data Structure / Computer Algorithm
- Computer Architecture
- Operating System

Programming Environments

- OS: Windows
- Compiler: Visual Studio
- Machine: Intel x86-based PC

Parallel Programming

- Shared Memory Model
- Distribued Memory Model
  - Message Passing Method

Multicores

- Multiple cores / Multiple CPUs
- Examples: Intel Core 2, Samsung Exynos

Multithreaded Programming

- C++ programming language
- Windows API

Lab #0: Start

- Basic program makes 100 millions

```
#include <stdio.h>

int main()
 {
    int acc = 0; int i;

    for(i = 0; i<50000000; i++) acc = acc + 2;

    printf(“%d”, acc);
 }
```

Windows API

- CreateThread

```
HANDLE WINAPI CreateThread(
        __in_opt LPSECURITY_ATTRIBUTES lpThreadAttributes,
        __in SIZE_T dwStackSize,
        __in LPTHREAD_START_ROUTINE lpStartAddress,
        __in_opt LPVOID lpParameter,
        __in DWORD dwCreationFlags,
        __out_opt LPDWORD lpThreadId );
```

  - lpStartAddress : function name
  - lpThreadId : thread ID

멀티 쓰레딩

- WaitForSingleObject

```
DWORD WINAPI WaitForSingleObject(
        __in HANDLE hHandle,
        __in DWORD dwMilliseconds );
```

  - hHandle : thread handle
  - dwMilliseconds : waiting time

Program outlines

- 2 threades
  - Global variable acc
  - Each thread executes a line,
    - acc = acc + 2;
    - 25 million times

Lab #1: 2 Threads

```
#include <windows.h>
#include <stdio.h>

int acc = 0;

DWORD WINAPI Bignumber(LPVOID lpVoid)
{ int I;
                for (i=1; i<=25000000; i++)  acc  = acc + 2;
                return 0;
}

int main()
{
DWORD addr;   HANDLE hd_th1, hd_th2;

                hd_th1 = CreateThread(NULL, 0, Bignumber, NULL, 0, &addr);
                hd_th2 = CreateThread(NULL, 0, Bignumber, NULL, 0, &addr);
                WaitForSingleObject(hd_th1, INFINITE); CloseHandle(hd_th1);
                WaitForSingleObject(hd_th2, INFINITE); CloseHandle(hd_th2);

                printf("Bignumber is %d\n", acc);
                getchar();
}
```

Data Race

- Disassemble
  - mov eax, [sum]
  - add eax, 2
  - mov [sum], eax

Locking & Unlocking

- Lock과 Unlock
  - 3 Windows apis

    - InitializeCriticalSection()
    - EnterCriticalSection()
    - LeaveCriticalSection()

api

```
#include <windows.h>

CRITICAL_SECTION  lock_var;

InitializeCriticalSection(CRITICAL_SECTION * lock_var);
```

api

```
#include <windows.h>

CRITICAL_SECTION  lock_var;

EnterCriticalSection(CRITICAL_SECTION * lock_var);
```

api

```
#include <windows.h>

CRITICAL_SECTION  lock_var;

LeaveCriticalSection(CRITICAL_SECTION * lock_var);
```

Lab #2: critical section

- make a correct program using critical section and the program in Lab #1

Debug

- Any problem ?
  - Program Speed !

Lab #3: atomic operation

```
DWORD WINAPI ThreadFunc(LPVOID lpVoid)
 {
  for (int i=0;i<50000000 / num_threads;i++)
     InterlockedExchangeAdd(&sum, 2);
  return 0;
 }
```

Lab #4: snooping

```
void *hd_th1(LPVOID lpVoid)
{
        while(flag != true)   ;         // waiting
                get_token = token;
}
```

```
void * hd_th2(LPVOID lpVoid)
{
      token = 123;
      flag = true;
}
```

Compiler

- disassemble
  - do you have some problem ?

Lab #5: compiler

- make a correct program using volatile and the program in Lab #4

Summary

- Critical section
- Mutual exclusion
- Lock / Unlock