



# 고급프로그래밍

## 병렬프로그래밍

Professor Jeong, Mun-Ho

Robot Vision & Intelligence Laboratory  
Kwangwoon University  
(02-940-5625, mhjeong@kw.ac.kr)

# Schedule

주차	주제		과제	퀴즈
1	과목소개	교과목 소개 (1), C++ 시작 (2)		
2	C++	C++ 프로그래밍의 기본 (3), 클래스와 객체 (4)	1	1
3		객체생성과 사용 (5)	2	2
4		함수와 참조 (6, 3/26), 복사 생성자와 함수중복(7)	3	3
5		static, friend, 연산자중복 (8, 4/2), 연산자중복 상속(9)	4	4
6		상속 (10, 4/9), 가상함수와 추상클래스 (11)		5
7		템플릿과 STL (12, 4/16), 입출력(13)	5	
8		중간고사		
9		파일 입출력(14), 예외처리 및 C 사용(15)		6
10		람다식(16, 5/7) , 멀티스레딩(17, 5/9)	6	7
11		멀티스레딩(18, 5/14), 고급문법(19, 5/16)		8
12		고급문법 2(20, 5/21), 고급문법 3(21, 5/23)		
13	병렬프로그래밍	병렬프로그래밍(22, 5/28)	7	9
14		병렬프로그래밍(23, 6/4), 병렬프로그래밍		
15		기말고사		

# 오늘의 학습내용

- 병렬프로그래밍

# Task 병렬화

# Task 병렬화

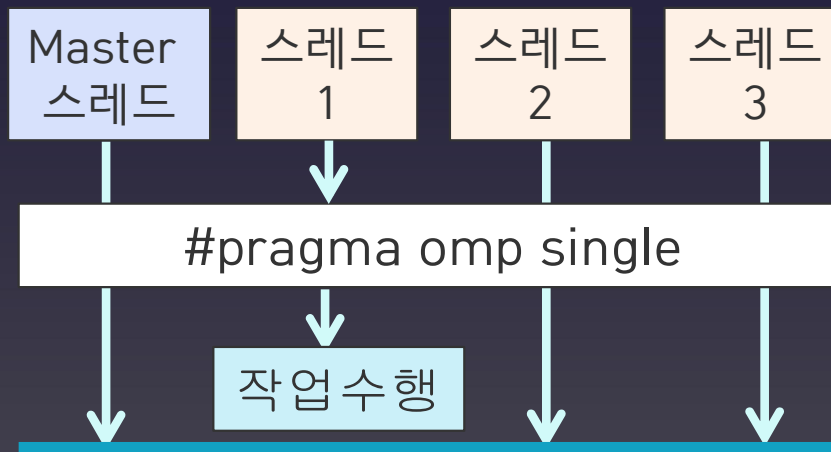
- 작업분할은 작업의 종류에 따라 비효율성 발생 가능 → Task 병렬화
- Task 병렬화 순서
  1. 병렬처리 할 작업 전부를 태스크 단위로 나눈다
  2. N개의 태스크를 하나의 스레드가 작업 큐에 넣는다
  3. 준비된 스레드부터 태스크를 가져와 실행한다
    - 하나의 태스크를 완료하면 다시 준비 상태가 된다
    - 작업 큐가 빌 때까지 반복한다

```
#pragma omp parallel
{
    #pragma omp single
        for(i=0; i<MAX; i++) //루프 회수만큼 태스크 생성
        {
            #pragma omp task //태스크 지정
            {
                :
            }
        }
}
```

# single 지시어

```
#pragma omp single  
{  
  :  
}
```

- 하나의 스레드에게만 해당 코드의 실행 지정
- 다른 스레드는 **single** 영역의 마지막 지점에서 암시적 동기화로 대기



암시적 동기화로 대기

# single 지시어

## ■ Example

```
9 int main()
10 {
11     #pragma omp parallel
12     {
13
14         #pragma omp single
15         {
16             printf("parallel region using single, thread %d\n\n", omp_get_thread_num());
17         }
18
19         printf("parallel region not using single, thread %d\n", omp_get_thread_num());
20
21     }
22
23     return 0;
24 }
```

C:\Qt\Tools\QtCreator\bin\qtcreator\_process\_stub.exe

```
parallel region using single, thread 2
parallel region not using single, thread 3
parallel region not using single, thread 6
parallel region not using single, thread 5
parallel region not using single, thread 4
parallel region not using single, thread 0
parallel region not using single, thread 7
parallel region not using single, thread 1
parallel region not using single, thread 2
Press <RETURN> to close this window...
```

# single 지시어

## ■ Example

```
9 int main()
10 {
11 #pragma omp parallel
12 {
13     printf("parallel region not using single, thread %d\n", omp_get_thread_num());
14
15     #pragma omp single
16     {
17         printf("\nparallel region using single, thread %d\n\n", omp_get_thread_num());
18     }
19 }
20
21 return 0;
22 }
```

C:\Qt\Tools\QtCreator\bin\qtcreator\_process\_stub.exe

parallel region not using single, thread 2

parallel region using single, thread 2

parallel region not using single, thread 4

parallel region not using single, thread 5

parallel region not using single, thread 1

parallel region not using single, thread 3

parallel region not using single, thread 0

parallel region not using single, thread 7

parallel region not using single, thread 6

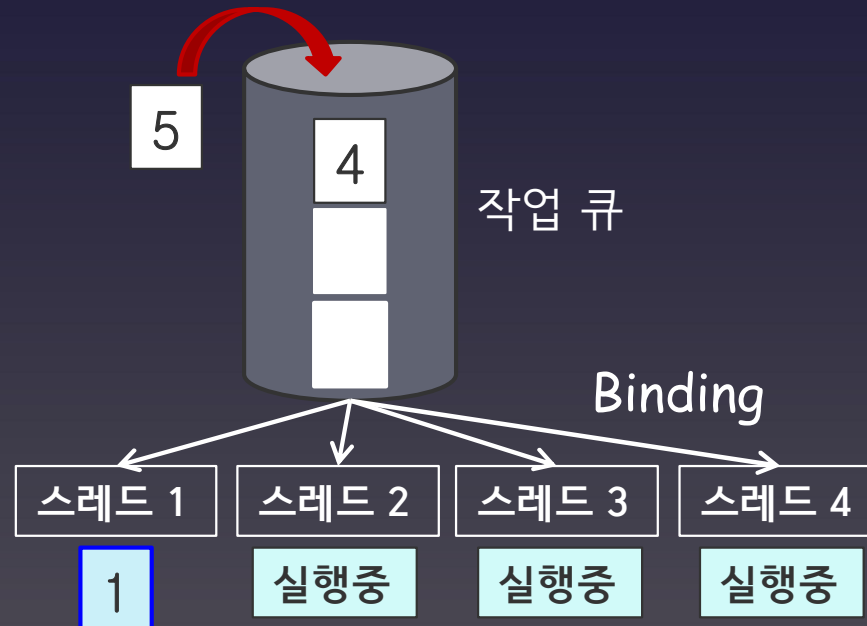
Press <RETURN> to close this window...



# task 지시어

```
#pragma omp task
{
  :
}
```

- 암시적 동기화 없음 -> taskwait을 이용한 대기 가능
- while 루프, c++ iterator, 재귀함수의 병렬화에 적용가능
- single 지시어를 사용하여 태스크를 작업 큐에 넣음



# task 지시어

## ■ Example

```
#include <iostream>
#include <omp.h>
#include <math.h>
#include <chrono>

using namespace std;

int main(int argc, char *argv[])
{
    int i = 0;

    #pragma omp parallel firstprivate(i)
    {
        #pragma omp single ←
        while(i++ < 100)
        {
            #pragma omp task ←
            printf("index %d, thread %d\n", i, omp_get_thread_num());
        }
    }
}
```

## ■ 보조 지시어

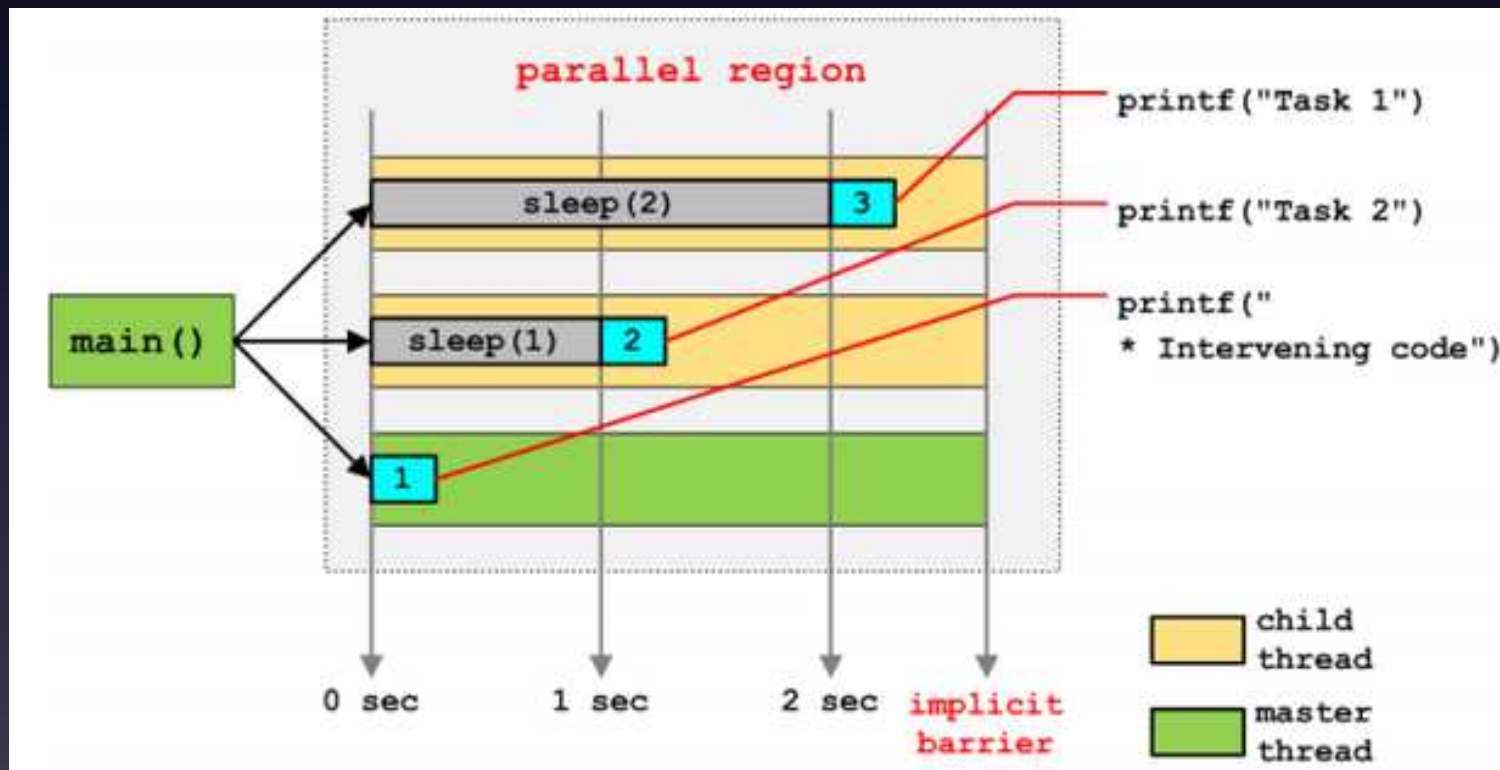
- private: thread 마다 독립적인 변수 선언
- firstprivate : private과 같으나 순차 영역에서 가져온 값으로 초기화

# task 지시어

- 아래 코드를 실행하여 예측한 결과와 비교 하시오

```
:
:
int main()
{
#pragma omp parallel
{
    #pragma omp single
    {
        printf("OMP parallel region, thread No. = %d\n", omp_get_num_threads());
        #pragma omp task
        {
            this_thread::sleep_for(chrono::seconds(2));
            printf("Task 1\n");
        }
        printf("* Intervening code\n");
        #pragma omp task
        {
            this_thread::sleep_for(chrono::seconds(1));
            printf("Task 2\n");
        }
    }
}
```

# task 지시어

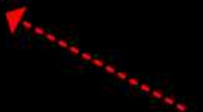


# task 지시어

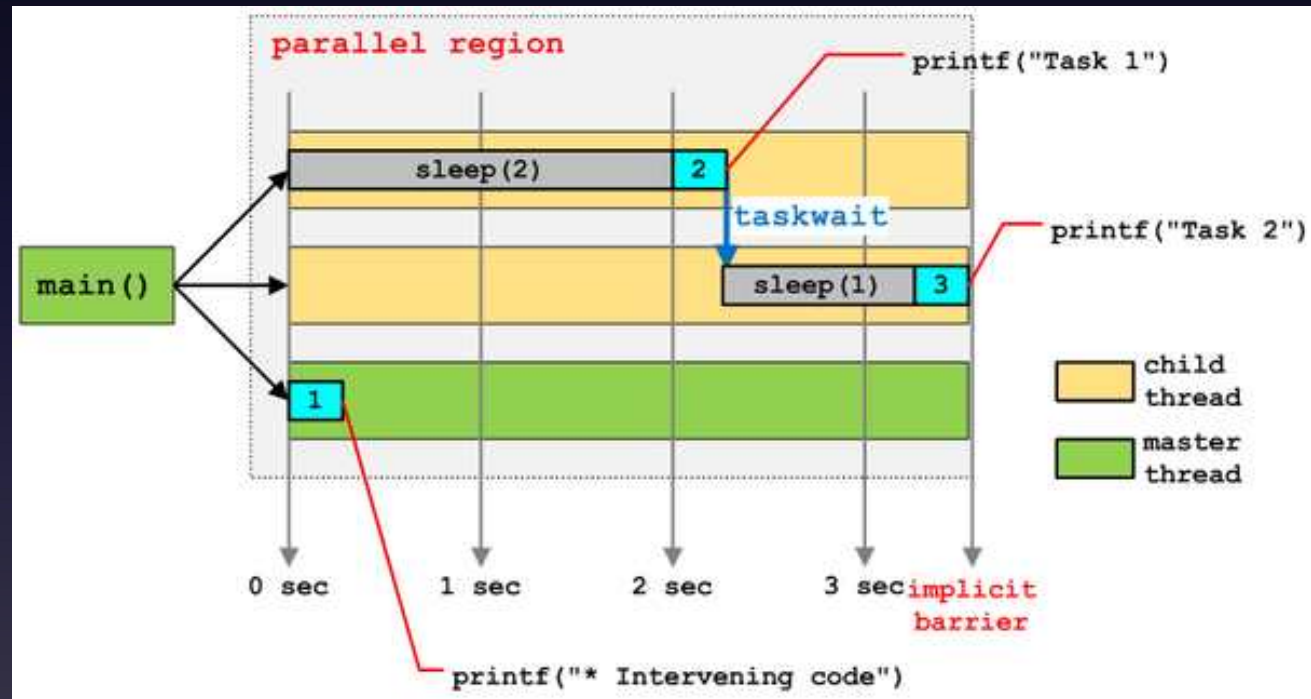
- 아래 코드를 실행하여 예측한 결과와 비교 하시오

```
int main()
{
#pragma omp parallel
{
    #pragma omp single
    {
        printf("OMP Parallel region, thread No. = %d\n", omp_get_num_threads());
        #pragma omp task
        {
            this_thread::sleep_for(chrono::seconds(2));
            printf("Task 1\n");
        }
        printf("* Intervening code\n");
        #pragma omp taskwait
    }

#pragma omp task
    {
        this_thread::sleep_for(chrono::seconds(1));
        printf("Task 2\n");
    }
}
}
```



# task 지시어



# Task 병렬화: 피보나치 수열

- for 루프 병렬화 적용 - static, dynamic

```
int main()
{
    const int nMAX = 41;
    int i = 0, nFibNumber[nMAX] = { 0 };
    timespec oStart, oEnd;
    clock_gettime(CLOCK_MONOTONIC, &oStart);
    #pragma omp parallel
    {
        #pragma omp for
        for(i=1; i<nMAX; i++) { nFibNumber[i] = Fibonacci(i); }
    }
    clock_gettime(CLOCK_MONOTONIC, &oEnd);
    //시간 출력
    cout << "Time : " << 1e3*(oEnd.tv_sec - oStart.tv_sec) + (oEnd.tv_nsec - oStart.tv_nsec) / 1e6 << " ms" << endl;

    //피보나치 수열 출력
    cout << "Fibonacci No : ";
    for(int i=1; i<nMAX; i++)
        cout << nFibNumber[i] << " ";
}
```

# Task 병렬화: 피보나치 수열

- 태스크 병렬화 적용 - 불균형 작업 분할

```
int main()
{
    const int nMAX = 41;
    int i = 0, nFibNumber[nMAX] = { 0 };
    timespec oStart, oEnd;

    clock_gettime(CLOCK_MONOTONIC, &oStart);

#pragma omp parallel
{
    #pragma omp single private(i)
    for(i=1; i<nMAX; i++)
    #pragma omp task
    {
        nFibNumber[i] = Fibonacci(i);
    }
}

    clock_gettime(CLOCK_MONOTONIC, &oEnd);

    //시간 출력
    cout << "Ellapsed Time : " << 1e3*(oEnd.tv_sec - oStart.tv_sec) +
        (oEnd.tv_nsec - oStart.tv_nsec) / 1e6 << " ms" << endl;

    //피보나치 수열 출력
    cout << "Fibonacci No : ";
    for(int i=1; i<nMAX; i++)
        cout << nFibNumber[i] << " ";
}
```



# Task 병렬화: 피보나치 수열

- 태스크 병렬화 적용 - 균형 작업 분할
  - shared(변수) : 외부에 선언된 변수와 thread에서 공유함

```
int Fibonacci(int n)
{
    int x,y;

    if(n<2)
        return n;
    else
    {
        x = Fibonacci(n-1);
        y = Fibonacci(n-2);

        return (x+y);
    }
}
```

```
int FibonacciTask(int n)
{
    int x,y;

    if(n<2)
        return n;
    else
    {
        #pragma omp task shared(x)
        x = Fibonacci(n-1);

        #pragma omp task shared(y)
        y = Fibonacci(n-2);

        #pragma omp taskwait
        return (x+y);
    }
}
```

# Task 병렬화: 피보나치 수열

- 태스크 병렬화 적용 - 균형 작업 분할

```
int main()
{
    const int nMAX = 41;
    int      i = 0, nFibNumber[nMAX] = { 0 };
    timespec oStart, oEnd;

    clock_gettime(CLOCK_MONOTONIC, &oStart);

#pragma omp parallel
    {
#pragma omp single private(i)
        for(i=1; i<nMAX; i++)
        {
            nFibNumber[i] = FibonacciTask(i);
        }
    }

    clock_gettime(CLOCK_MONOTONIC, &oEnd);
```

# Thread Memory Space

# Execution Environment Function

- `omp_set_num_threads` : parallel region에서 운용할 thread 개수 지정
- `omp_get_num_threads` : parallel region내에서 생성된 thread 개수 반환
- `omp_get_max_threads` : 생성 가능한 최대 thread 개수 지정
- `omp_get_thread_num` : thread별 고유번호 반환

# 예제

```
#include <iostream>
#include <omp.h>
int main()
{
#pragma omp parallel num_threads(2)
{
    printf("omp_get_thread_num() = %d\n", omp_get_thread_num());
    #pragma omp for
    for(int i = 0 ; i < 10 ; i++)
        printf("i = %d, thread num = %d\n", i, omp_get_thread_num());
}
```

# Thread Local Variables

- Thread 영역 내에서 선언된 변수

```
:
int main()
{
    int x=0;
    cout << "main area x: " << x << endl;
    cout.sync_with_stdio(true);

    #pragma omp parallel num_threads(3)
    {
        if(omp_get_thread_num() == 0)
            x = 1; //0번 스레드는 1로 설정
        else if(omp_get_thread_num() == 1)
            x = 2; //1번 스레드는 1로 설정
        else
            x = 3;

        //스레드 번호와 x 값을 출력
        cout << "No." << omp_get_thread_num() << " thread x:" << x << endl;
    }

    cout << "main area x: " << x << endl;
}
```

# Thread Private Variables

- OpenMP가 지원하는 Thread local memory

```
int main()
{
    int x=0;

    cout << "main area x: " << x << endl;
    cout.sync_with_stdio(true);

    #pragma omp parallel num_threads(3) private(x)
    {
        if(omp_get_thread_num() == 0)
            x = 1; //0번 스레드는 1로 설정
        else if(omp_get_thread_num() == 1)
            x = 2; //1번 스레드는 1로 설정
        else
            x = 3;

        //스레드 번호와 x 값을 출력
        cout << "No." << omp_get_thread_num() << " thread x:" << x << endl;
    }

    cout << "main area x: " << x << endl;
    return 0;
}
```

