

Lecture 4-1

Parallel Constructs

Introduction to OpenMP

Parallel Construction

```
#pragma omp parallel [clause list]
{/* structured block */}
```

- 병렬 영역(Parallel region) 지시어
 - Structured block 안의 코드가 병렬처리 됨을 지시
- fork()를 통해 team of threads가 생성 됨
- 별도의 Work-sharing 지시어가 없으면,
모든 스레드가 동일한 작업을 수행
 - Structured block 안의 코드 수행

Parallel Construction

```
#pragma omp parallel
```

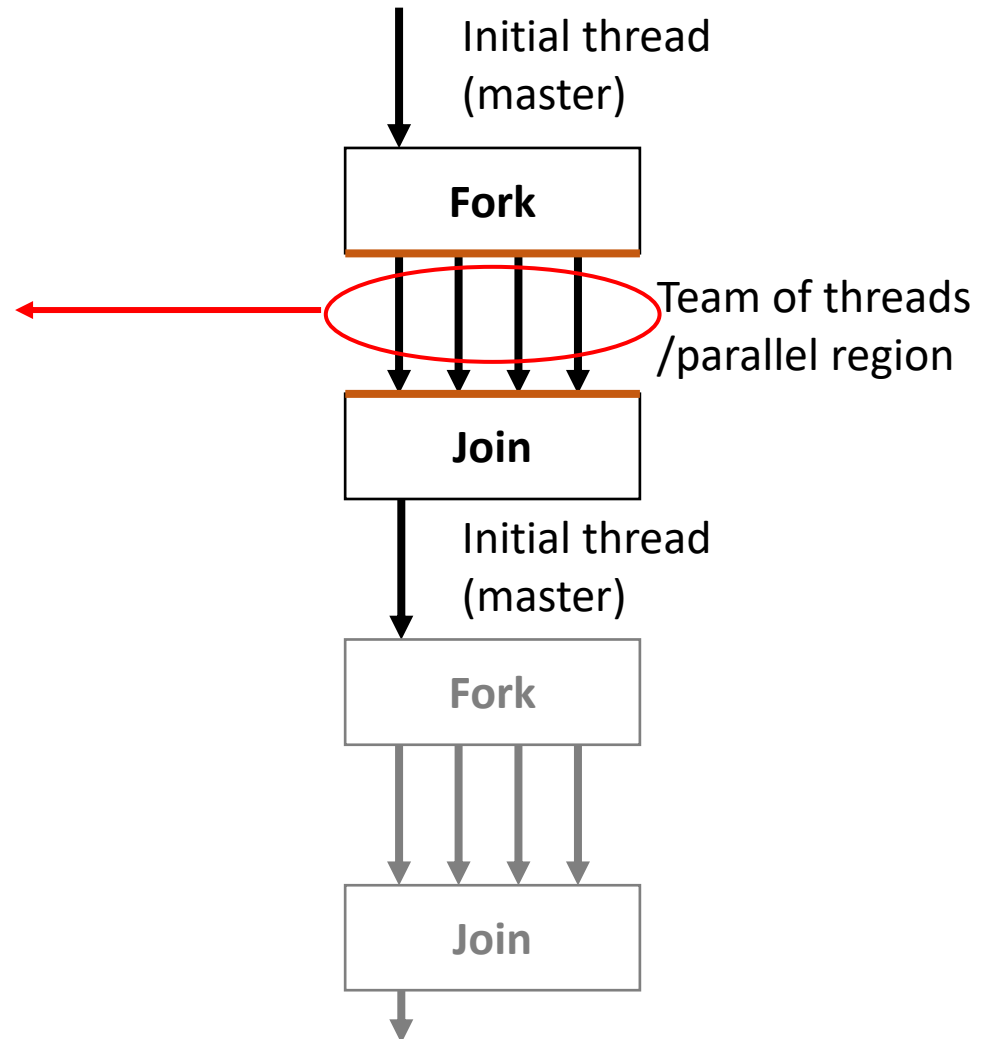
```
{
```

```
    // do something
```

```
    // 모든 스레드가
```

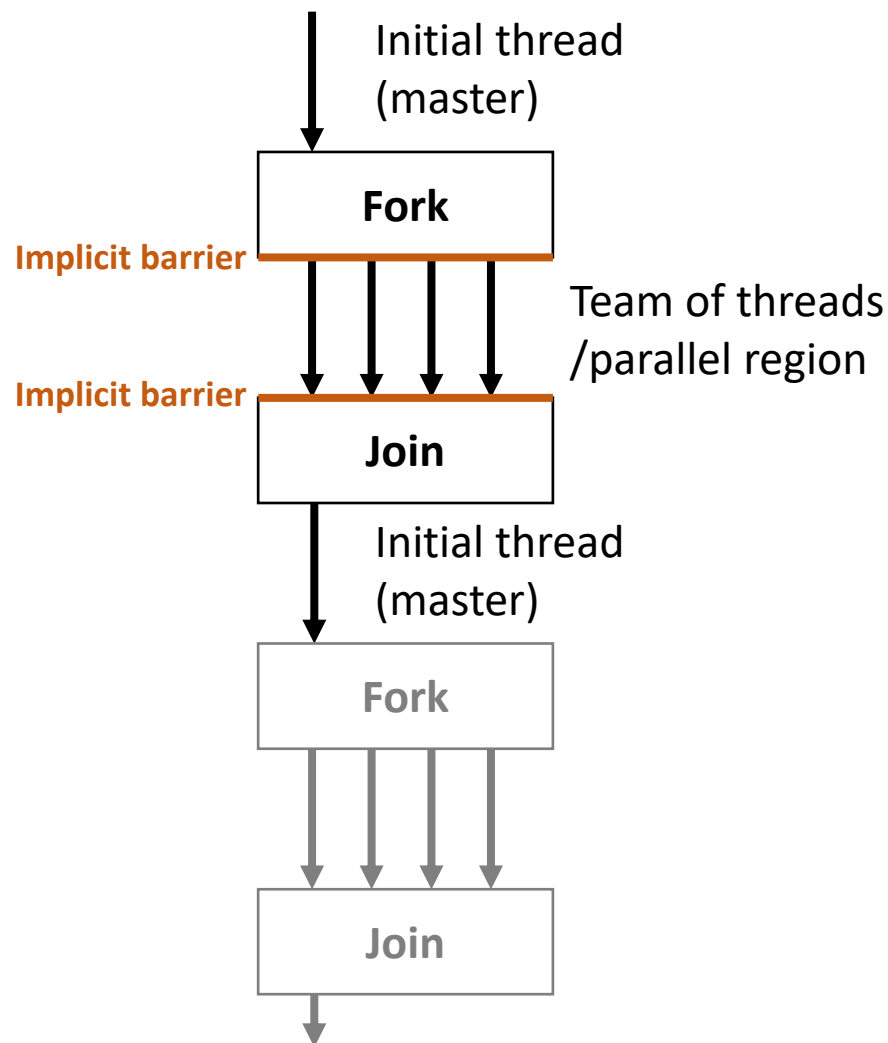
```
    // 동일한 코드를 실행
```

```
}
```



Barrier

- Thread 들이 동기화 되는 지점
 - 자신의 일이 끝나면, barrier에서 대기
 - 모든 thread가 일을 마치면 다음으로 이동



Key Functions & Clauses

- int **omp_get_num_threads()**

- return: 전체 스레드의 수

- int **omp_get_thread_num()**

- return: 자신의 스레드 ID

- 스레드의 수 지정

#pragma omp parallel **num_threads(n)**

Example – Vector Sum

```
int numElePerThread = VECTOR_SIZE / numThreads;
int *start = new int[numThreads];
int *end = new int[numThreads];

printf("Work decomposition\n");
for (int tID = 0; tID < numThreads; tID++) {
    start[tID] = numElePerThread * tID;
    end[tID] = numElePerThread * (tID + 1);

    if (tID == numThreads - 1) // for the last thread
        end[numThreads - 1] = VECTOR_SIZE;

    printf("\t[T%d] %d ~ %d\n", tID, start[tID], end[tID]);
}

#pragma omp parallel num_threads(numThreads)
{ // parallel region
    int tID = omp_get_thread_num();
    for (int i = start[tID]; i < end[tID]; i++)
        c[i] = a[i] + b[i];
}
```

if clause

```
#pragma omp parallel if(scalar-expression)  
{/* structured block */}
```

- 병렬화 할지를 결정 하는 조건문

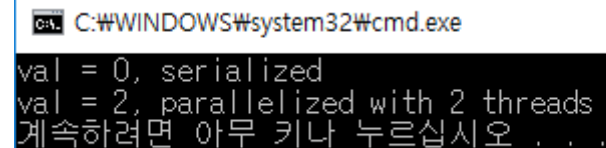
if clause

```

void test(int val)
{
    #pragma omp parallel if (val)
    if (omp_in_parallel())
    {
        #pragma omp single
        printf_s("val = %d, parallelized with %d threads\n",
            val, omp_get_num_threads());
    }
    else
    {
        printf_s("val = %d, serialized\n", val);
    }
}

int main()
{
    omp_set_num_threads(2);
    test(0);
    test(2);
}

```



```

C:\WINDOWS\system32\cmd.exe
val = 0, serialized
val = 2, parallelized with 2 threads
계속하려면 아무 키나 누르십시오 . . .

```


Lecture 4-2

Work-Sharing Constructs

Introduction to OpenMP

Work-Sharing Construct

- 일을 thread들에게 어떻게 나누어 줄지를 지시
 - Work(or computation) distribution
- 반드시 **parallel construct** 지시어와 함께 사용
 - E.g.,

```
#pragma omp parallel  
{  
    #pragma omp for  
}
```
- **Work-Sharing Construct**
 - Loop Construct
 - Sections Construct
 - Single Construct

Loop Construct

```
#pragma omp for [clause list]
```

```
for ( index = start; index < end ; index +=incr )  
{ /* do something */ }
```

- 가장 널리 사용되는 지시어 중 하나
- **for loop를 병렬화**
 - Index는 반드시 integer counter variable 이어야 함
 - 예) float는 안 됨
 - start, end, incr는 중간에 바뀌면 안됨
 - Loop 본문에 exit, return 등을 호출 하면 안됨

Loop Construct (example)

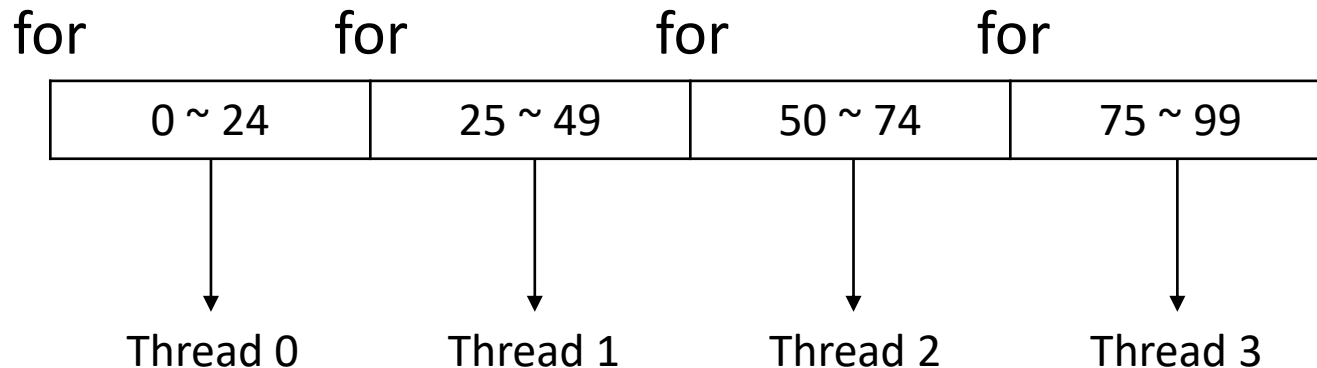
```
#pragma omp parallel
{
    #pragma omp for
    for (int i = 0; i < 16; i++) {
        printf("[Thread %d] executes loop iteration %d\n",
            omp_get_thread_num(), i);
    }
}
```

```
C:\WINDOWS\system32\cmd.exe
[Thread 1] executes loop iteration 2
[Thread 0] executes loop iteration 0
[Thread 3] executes loop iteration 6
[Thread 0] executes loop iteration 1
[Thread 6] executes loop iteration 12
[Thread 5] executes loop iteration 10
[Thread 7] executes loop iteration 14
[Thread 6] executes loop iteration 13
[Thread 5] executes loop iteration 11
[Thread 7] executes loop iteration 15
[Thread 1] executes loop iteration 3
[Thread 4] executes loop iteration 8
[Thread 4] executes loop iteration 9
[Thread 2] executes loop iteration 4
[Thread 2] executes loop iteration 5
[Thread 3] executes loop iteration 7
계속하려면 아무 키나 누르십시오 . . .
```

Loop Construct

- **Work decomposition (default)**

```
#pragma omp parallel num_threads(4)
{
    #pragma omp for
    for (int i = 0; i < 100; i++){
    }
```



* **schedule clause**를 이용하여 조절 가능 ➔ 다음 시간에

Loop Construct – Vector Sum

```
printf("* Using %d thread(s)\n", numThreads);
int numElePerThread = VECTOR_SIZE / numThreads;

int *start = new int[numThreads];
int *end = new int[numThreads];

printf("Work decomposition\n");
for (int tID = 0; tID < numThreads; tID++) {
    start[tID] = numElePerThread * tID;
```

```
#pragma omp parallel num_threads(numThreads)
{
    #pragma omp for
    for (int i = 0; i < VECTOR_SIZE; i++) {
        c[i] = a[i] + b[i];
    }
}
```

```
    // parallel region
    int tID = omp_get_thread_num();
    for (int i = start[tID]; i < end[tID]; i++) {
        c[i] = a[i] + b[i];
    }
}

timer.offTimer(numThreads);

delete start; delete end;
```

Loop Construct

- 주의: 데이터 의존성

```
int fibo[10];  
fibo[0] = fibo[1] = 1;  
  
#pragma omp parallel  
#pragma omp for  
for (int i = 2; i < 10; i++) {  
    fibo[i] = fibo[i - 1] + fibo[i - 2];  
}
```

CAL C:\WINDOWS\system32\cmd.exe

1 1 2 3 5 8 13 21 34 55

계속하려면 아무 키나 누르십시오 . . .

CAL C:\WINDOWS\system32\cmd.exe

1 1 2 124196602 124196604 248393206 31 1 32 1

계속하려면 아무 키나 누르십시오 . . .

Thread Safety

- 다중 thread가 문제를 일으키지 않고,
동시에 실행 될 수 있는 경우
Thread safety 하다고 함
- 알고리즘 설계 시, thread safety를 고려 해야 함

Loop Construct (Example)

```
#pragma omp parallel
{
    #pragma omp for
    for (int i = 0; i < SIZE; i++)
        a[i] = i;
    #pragma omp for
    for (int i = 0; i < SIZE; i++)
        b[i] = 2 * a[i];
}
```

← Implicit barrier

Loop Construct – 활용 Tip!

```
int main(void)
{
    int n = 4;
    #pragma omp parallel num_threads(4)
    {
        #pragma omp for
        for (int tID = 0; tID < n; tID++) {
            // tID를 사용하여 스레드 별 작업 정의
        }
    }
}
```

Sections Construct

```
#pragma omp sections [clause list]
```

```
{
```

```
    #pragma omp section
```

```
{
```

```
        // do task A
```

```
}
```

```
    #pragma omp section
```

```
{
```

```
        // do task B
```

```
}
```

```
}
```

← Implicit barrier

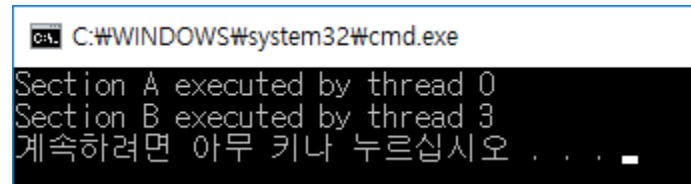
Sections Construct

- Thread 들이 서로 다른 일을 하도록 하는 지시어
 - Load-balancing을 직접 해주어야 함
- Section의 수 < thread의 수
 - 각각의 section이 하나의 thread에서 수행됨
 - 남은 thread는 idle 상태
- Section의 수 > thread의 수
 - 일부 thread가 여러 section 처리

Sections Construct (Example)

```
int main(void)
{
    #pragma omp parallel num_threads(8)
    {
        #pragma omp sections
        {
            #pragma omp section
            printf("Section A executed by thread %d\n"
                  , omp_get_thread_num());

            #pragma omp section
            printf("Section B executed by thread %d\n"
                  , omp_get_thread_num());
        }
    }
}
```



```
C:\WINDOWS\system32\cmd.exe
Section A executed by thread 0
Section B executed by thread 3
계속하려면 아무 키나 누르십시오 . . . .
```

Single Construct

```
#pragma omp single [clause list]  
{ /* do something */ }
```

- 하나의 thread만 수행하도록 하는 지시어
 - 어느 thread가 수행할 지는 비결정적

Single Construct (example)

```

int main(void)
{
    int a;
    int b[10] = { 0 };

    #pragma omp parallel
    {
        #pragma omp single
        {
            a = 10;
            printf("Single construct excuted by thread %d\n",
                omp_get_thread_num());
        }
        ← Implicit barrier
        #pragma omp for
        for (int i = 0; i < 10; i++)
            b[i] = a;
        ← Implicit barrier
        for (int i = 0; i < 10; i++)
            printf("b[%d] = %d\n", i, b[i]);
    }
}

```

C:\WINDOWS\system32\cmd.exe

Single construct excuted by thread 0

b[0] = 10
 b[1] = 10
 b[2] = 10
 b[3] = 10
 b[4] = 10
 b[5] = 10
 b[6] = 10
 b[7] = 10
 b[8] = 10
 b[9] = 10

계속하려면 아무 키나 누르십시오 . . .

Combined Parallel Work-Sharing Constructs

Full version	Combined construct
<pre>#pragma omp parallel { #pragma omp for for-loop }</pre>	<pre>#pragma omp parallel for for-loop</pre>
<pre>#pragma omp parallel { #pragma omp sections { [#pragma omp section] <i>structured block</i> [#pragma omp section <i>structured block</i>] ... } }</pre>	<pre>#pragma omp parallel sections { [#pragma omp section] <i>structured block</i> [#pragma omp section <i>structured block</i>] ... }</pre>

nowait clause

`#pragma omp WS-construct nowait`

- **Implicit barrier를 생성하지 않게 함**
 - 프로그램의 성능을 fine-tune 하기 위해 사용
- **Barrier가 필요 없는지, 정확히 판단 후 사용 해야 함**
 - 예) Thread 간 작업이 완전히 독립적일 때



nowait clause (example)

```
int main(void)
{
    int a; int b[10] = { 0 };

    #pragma omp parallel num_threads(8)
    {
        #pragma omp single nowait
        {
            Sleep(1);
            a = 10;
            printf("Single construct excuted by thread %d\n",
                omp_get_thread_num());
        }

        #pragma omp for
        for (int i = 0; i < 10; i++)
            b[i] = a;
    }
    ← Implicit barrier
    for (int i = 0; i < 10; i++)
        printf("b[%d] = %d\n", i, b[i]);
}
```

```
C:\WINDOWS\system32\cmd.exe
Single construct excuted by thread 0
b[0] = 10
b[1] = 10
b[2] = 2
b[3] = 2
b[4] = 2
b[5] = 2
b[6] = 2
b[7] = 2
b[8] = 2
b[9] = 2
계속하려면 아무 키나 누르십시오 . . .
```

Lecture 4-3

Scope of Variables

Introduction to OpenMP

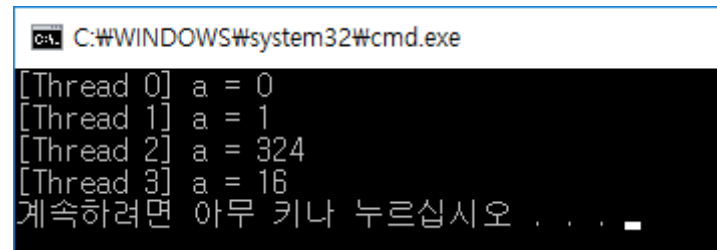
Scope of Variables

```
void main(void)
{
    int a = 0;
    int re[4] = { 0 };

    #pragma omp parallel for num_threads(4)
    LOOP_I(4)
    {
        a = a+i;
        a = a*a;

        re[i] = a;
    }

    LOOP_I(4)
    printf("[Thread %d] a = %d\n", i, re[i]);
}
```



```
cmd: C:\WINDOWS\system32\cmd.exe
[Thread 0] a = 0
[Thread 1] a = 1
[Thread 2] a = 324
[Thread 3] a = 16
계속하려면 아무 키나 누르십시오 . . . .
```

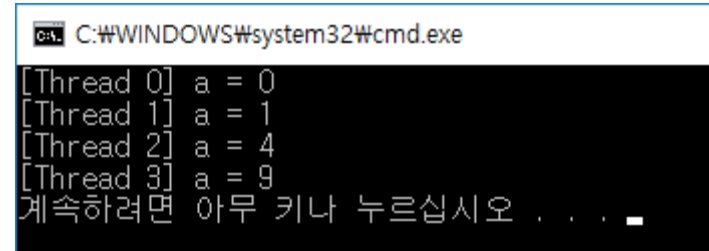
Scope of Variables

```
void main(void)
{
    int re[4] = { 0 };

    #pragma omp parallel for num_threads(4)
    LOOP_I(4)
    {
        int a = 0;
        a = a+i;
        a = a*a;

        re[i] = a;
    }

    LOOP_I(4)
    printf("[Thread %d] a = %d\n", i, re[i]);
}
```



```
C:\WINDOWS\system32\cmd.exe
[Thread 0] a = 0
[Thread 1] a = 1
[Thread 2] a = 4
[Thread 3] a = 9
계속하려면 아무 키나 누르십시오 . . .
```

Scope of Variables

- Parallel region
(structured block) 전에
선언된 변수는 모든
thread가 공유 함
 - 동시에 접근/수정 가능
 - 동기화(Synchronization)
필요
 - Next class!

```
int a; // shared
#pragma omp parallel
{
    int b; // Private
}
```

Scope of Variables (example)

```
#pragma omp parallel num_threads(4) //스레드 수 4개 지정
{
    int m = (NUM / omp_get_num_threads());
    // 스레드당 데이터 처리 개수

    int st = m*omp_get_thread_num(); //Tid 값에 따른 범위
    int end = m*(omp_get_thread_num() + 1);

    printf("%d ~ %d\n", st, end);
    //스레드 ID에 따른 처리할 데이터의 범위 출력

    timer.onTimer(1);

    for (int i = st; i<end; i++)
        C[i] = (A[i] + B[i]);

    timer.offTimer(1);
}
```

* DS timer 는 thread safety 하지 않음

```
C:\Windows\system32\cmd.exe
33554432 ~ 67108864
0 ~ 33554432
100663296 ~ 134217728
67108864 ~ 100663296

*          DS_timer Report          *
* The number of timer = 4, counter = 4
**** Timer report ****
Timer 0 : 10.54212 ms (10.54212 ms)
Timer 1 : 188.05416 ms (188.05416 ms)
**** Counter report ****
*          End of the report          *
계속하려면 아무 키나 누르십시오 . . .
```

OpenMP Clauses for Scope of Var.

- 변수의 적용 범위를 명시적으로 지정 가능
- shared
- private
- lastprivate
- firstprivate
- default

Shared Clause

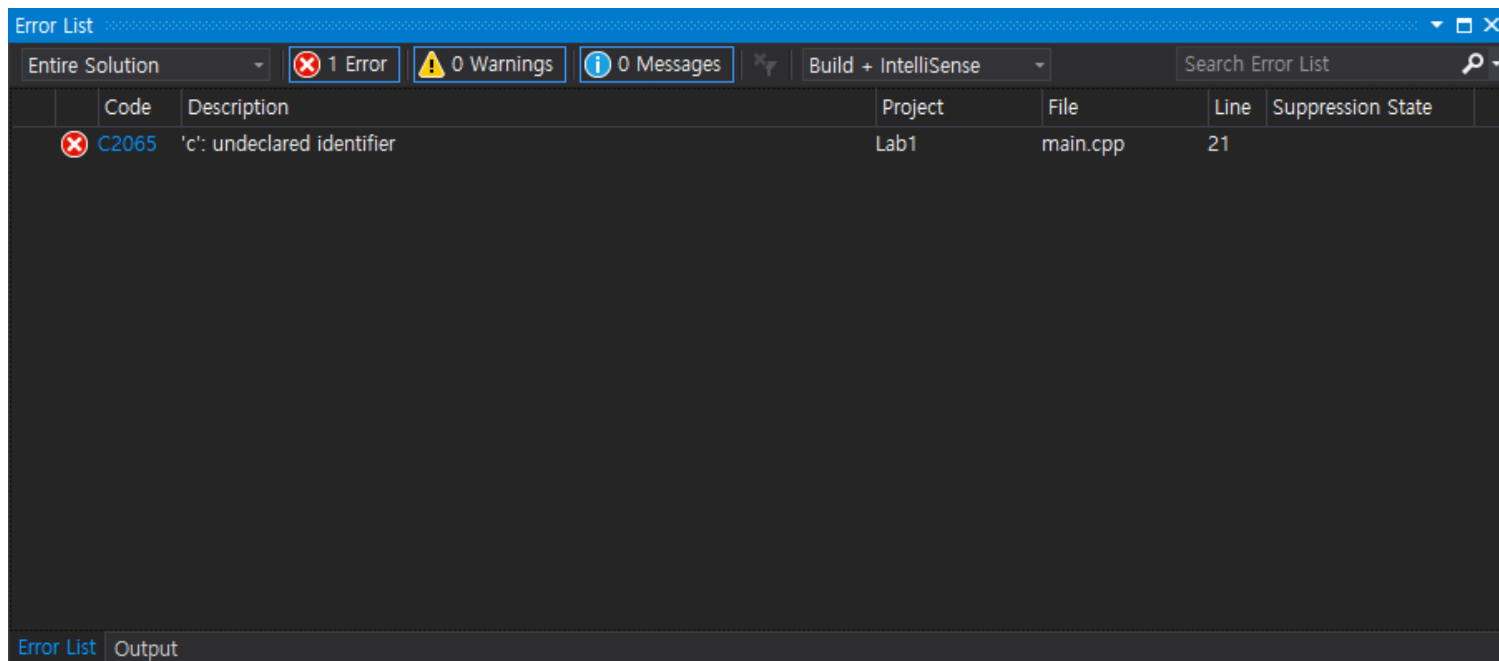
`#pragma omp parallel shared(var)`

- 주어진 변수를 shared variable로 사용
 - 모든 thread가 접근/수정 가능

```
#pragma omp parallel num_threads(numThreads) shared(a,b,c)
{
    #pragma omp for
    for (int i = 0; i < VECTOR_SIZE; i++) {
        c[i] = a[i] + b[i];
    }
}
```

Shared Clause

```
#pragma omp parallel shared(c)
{
    int c = omp_get_thread_num();
}
```



Private Clause

`#pragma omp WS-construct private(var)`

- 주어진 변수를 **private variable**로 사용
 - 각각의 thread는 자신만의 local variable을 생성
 - 자신에게 주어진 local variable 만 접근/수정 가능

```
int tid = 0;
int priVar = 10;
#pragma omp parallel for num_threads(4) private(tid, priVar)
LOOP_I(4)
{
    tid = omp_get_thread_num();
    priVar = tid * 10;
    printf("[Thread %d] priVar = %d\n", tid, priVar);
}
```

Private Clause (example)

```
int tid = 0;
int priVar;
#pragma omp parallel for num_threads(4) private(tid, priVar)
LOOP_I(4)
{
    tid = omp_get_thread_num();
    priVar = tid * 10;
    printf("[Thread %d] priVar = %d\n", tid, priVar);
}
```

C:\WINDOWS\system32\cmd.exe

```
[Thread 0] priVar = 0
[Thread 1] priVar = 10
[Thread 2] priVar = 20
[Thread 3] priVar = 30
계속하려면 아무 키나 누르십시오 . . .
```

Shared, Private Clause (example)

```

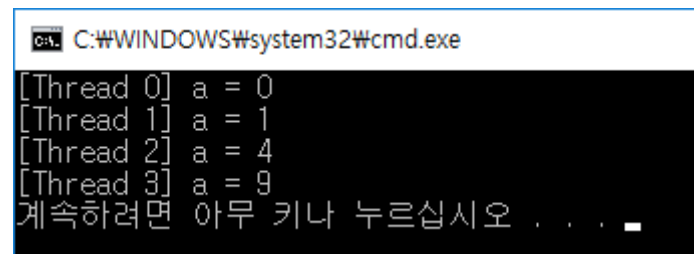
void main(void)
{
    int a = 0;
    int re[4] = { 0 };

    #pragma omp parallel for num_threads(4) shared(re) private(a)
    for (int i = 0; i < 4; i++)
    {
        a = i;
        a = a*a;

        re[i] = a;
    }

    LOOP_I(4)
        printf("[Thread %d] a = %d\n", i, re[i]);
}

```



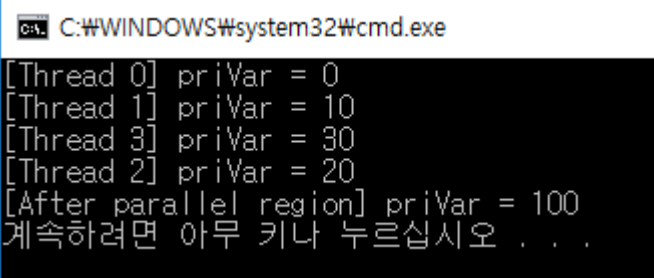
```

C:\WINDOWS\system32\cmd.exe
[Thread 0] a = 0
[Thread 1] a = 1
[Thread 2] a = 4
[Thread 3] a = 9
계속하려면 아무 키나 누르십시오 . . .

```

Private Clause (example)

```
int tid = 0;
int priVar = 100;
#pragma omp parallel for num_threads(4) private(tid, priVar)
LOOP_I(4)
{
    tid = omp_get_thread_num();
    priVar = tid * 10;
    printf("[Thread %d] priVar = %d\n", tid, priVar);
}
printf("[After parallel region] priVar = %d\n", priVar);
```



```
C:\WINDOWS\system32\cmd.exe
[Thread 0] priVar = 0
[Thread 1] priVar = 10
[Thread 3] priVar = 30
[Thread 2] priVar = 20
[After parallel region] priVar = 100
계속하려면 아무 키나 누르십시오 . . .
```

Lastprivate Clause

```
int tid = 0;
int priVar = 100;

#pragma omp parallel for num_threads(4) private(tid) lastprivate(priVar)
LOOP_I(4)
{
    tid = omp_get_thread_num();
    priVar = tid * 10;
    printf("[Thread %d] priVar = %d\n", tid, priVar);
}

printf("[After parallel region] priVar = %d\n", priVar);
```

C:\WINDOWS\system32\cmd.exe

```
[Thread 0] priVar = 0
[Thread 1] priVar = 10
[Thread 2] priVar = 20
[Thread 3] priVar = 30
[After parallel region] priVar = 30
계속하려면 아무 키나 누르십시오 . . .
```


Lastprivate Clause

- What is last?
 - **Loop construct**
 - Loop 의 마지막
 - **Sections construct**
 - 마지막 section의 code block

Lastprivate Clause (example)

```
int tid = 0;
int priVar;
#pragma omp parallel for num_threads(4) private(tid) lastprivate(priVar)
LOOP_I(16)
{
    tid = omp_get_thread_num();
    priVar = tid * 10 * i;
    printf("[Thread %d, loop %d] priVar = %d\n", i, tid, priVar);
}

printf("[After parallel region] priVar = %d\n", priVar);
```



```
C:\WINDOWS\system32\cmd.exe
[Thread 0, loop 0] priVar = 0
[Thread 4, loop 1] priVar = 40
[Thread 1, loop 0] priVar = 0
[Thread 8, loop 2] priVar = 160
[Thread 12, loop 3] priVar = 360
[Thread 2, loop 0] priVar = 0
[Thread 9, loop 2] priVar = 180
[Thread 13, loop 3] priVar = 390
[Thread 10, loop 2] priVar = 200
[Thread 5, loop 1] priVar = 50
[Thread 11, loop 2] priVar = 220
[Thread 6, loop 1] priVar = 60
[Thread 3, loop 0] priVar = 0
[Thread 7, loop 1] priVar = 70
[Thread 14, loop 3] priVar = 420
[Thread 15, loop 3] priVar = 450
[After parallel region] priVar = 450
계속하려면 아무 키나 누르십시오 . . .
```

Private Clause (example)

```
int main(void)
{
    int i = 10;

    #pragma omp parallel private(i)
    {
        printf("thread %d: i = %d\n", omp_get_thread_num(), i);
        i = 1000 + omp_get_thread_num();
    }

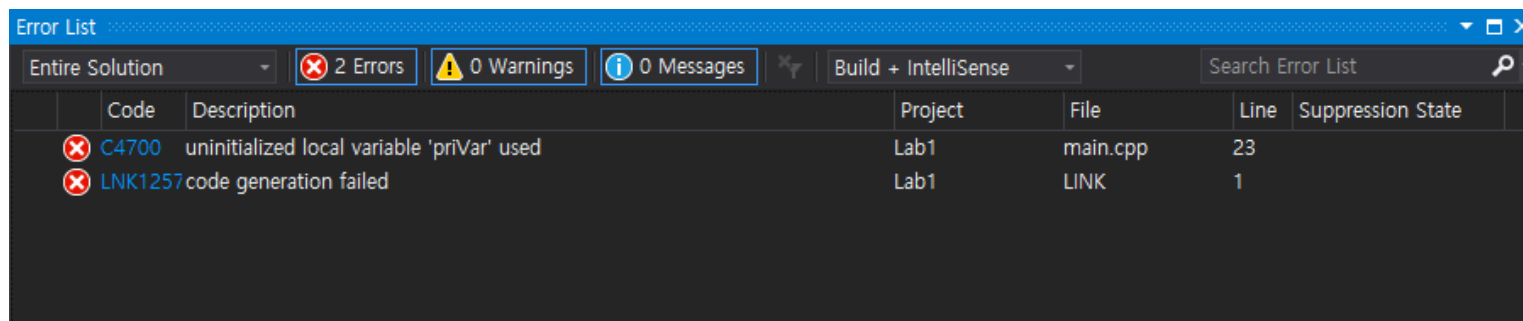
    printf("i = %d\n", i);

    return 0;
}
```

```
thread 0: i = 0
thread 3: i = 32717
thread 1: i = 32717
thread 2: i = 1
i = 10
```

Private Clause (example)

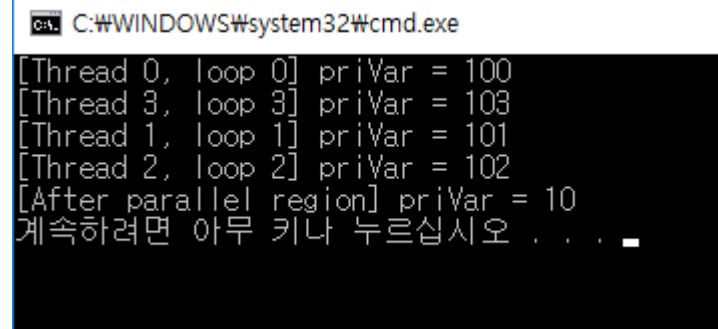
```
int tid = 0;
int priVar = 10;
#pragma omp parallel for num_threads(4) private(tid, priVar)
LOOP_I(4)
{
    tid = omp_get_thread_num();
    priVar = priVar * 10 + i;
    printf("[Thread %d] priVar = %d\n", tid, priVar);
}
```



Firstprivate Clause

- private variable을 parallel region 전에 주어진 값으로 초기화

```
int tid = 0;
int priVar = 10;
#pragma omp parallel for num_threads(4) private(tid) firstprivate(priVar)
LOOP_I(4)
{
    tid = omp_get_thread_num();
    priVar = priVar * 10 + i;
    printf("[Thread %d] priVar = %d\n", tid, priVar);
}
```



```
C:\WINDOWS\system32\cmd.exe
[Thread 0, loop 0] priVar = 100
[Thread 3, loop 3] priVar = 103
[Thread 1, loop 1] priVar = 101
[Thread 2, loop 2] priVar = 102
[After parallel region] priVar = 10
계속하려면 아무 키나 누르십시오 . . .
```

Default Clause

- Variable 공유의 기본 값 설정
 - default(none | shared)
 - C/C++은 두가지만 지원
 - default(none)
 - 모든 변수에 대해 사용자 직접 지정하도록 강제
 - 사용 권장

Default Clause

```
int tid = 0;
int priVar = 10 ;
#pragma omp parallel default(none)
LOOP_I(4)
{
    tid = omp_get_thread_num();
    priVar = priVar * 10 + i;
    printf("[Thread %d, loop %d] priVar = %d\n", i, tid, priVar);
}
```

Error List							
Entire Solution		2 Errors	0 Warnings	0 Messages	Build + IntelliSense		
	Code	Description	Project	File	Line	Suppression State	
✖	C3052	'tid': variable doesn't appear in a data-sharing clause under a default(none) clause	Lab1	main.cpp	23		
✖	C3052	'priVar': variable doesn't appear in a data-sharing clause under a default (none) clause	Lab1	main.cpp	24		

요약

- **Parallel construct**
- **Work-sharing construct**
 - Loop construct
 - Sections construct
- **Scope of Variables**
 - Shared, Private, Default clauses