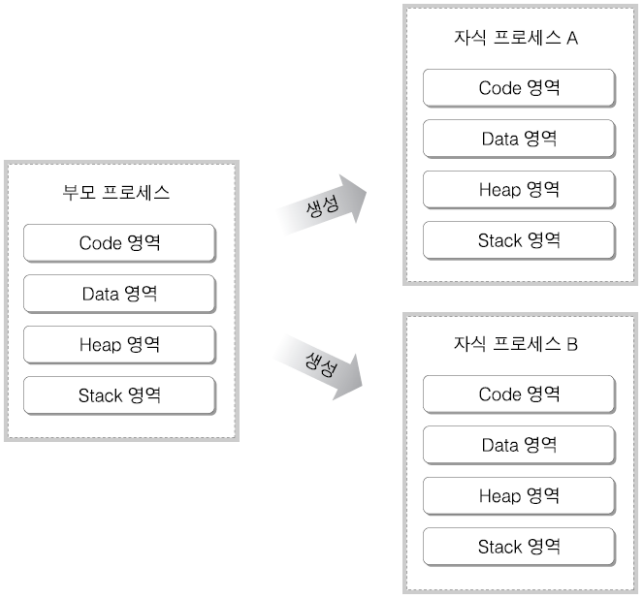
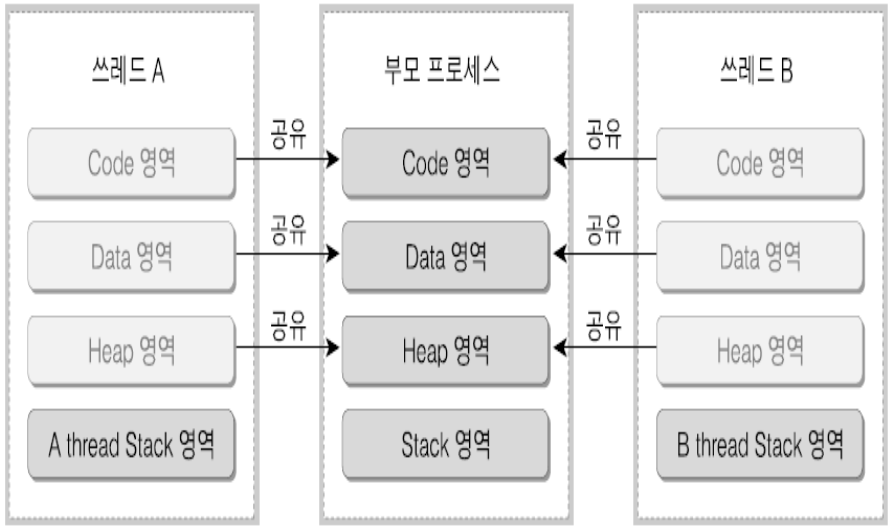
쓰래드

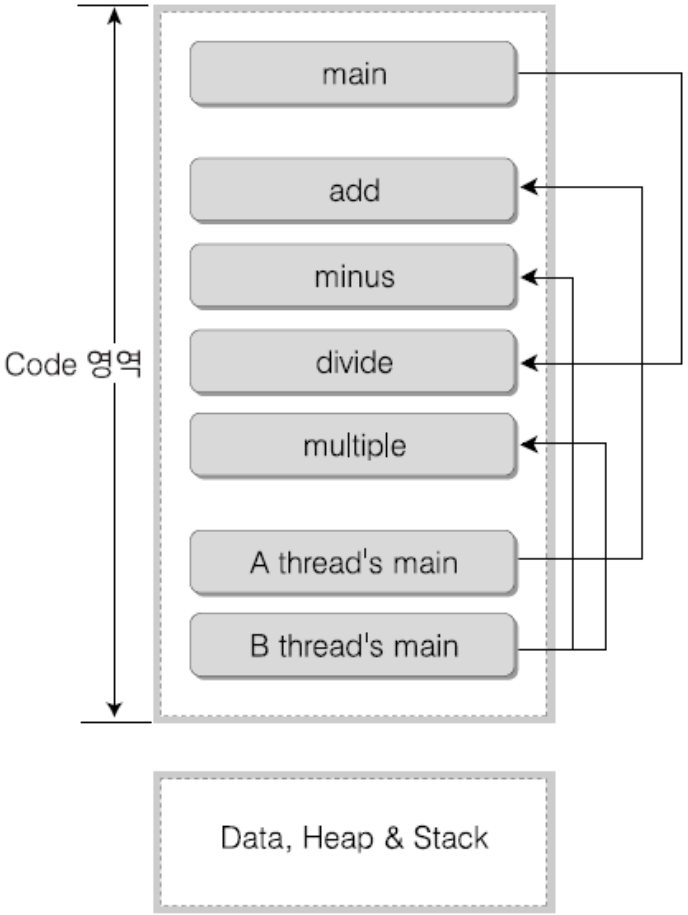
홍익대학교
김혜영편집

프로세스 vs. 쓰레드


부모 프로세스
Code 영역
Data 영역
Heap 영역
Stack 영역
생성
생성
자식 프로세스 A
Code 영역
Data 영역
Heap 영역
Stack 영역
자식 프로세스 B
Code 영역
Data 영역
Heap 영역
Stack 영역

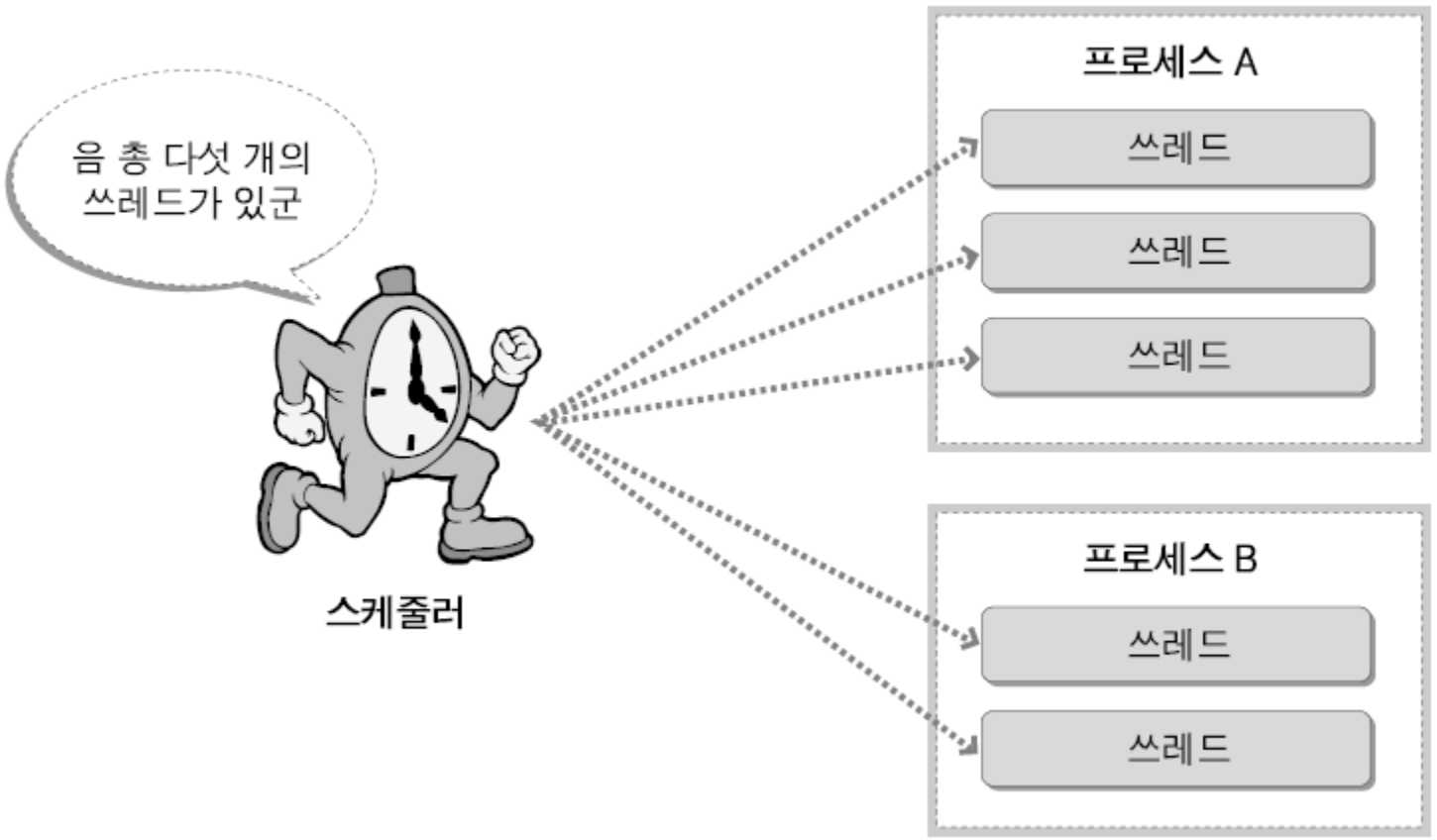

쓰레드 A
Code 영역
Data 영역
Heap 영역
A thread Stack 영역
공유
공유
공유
부모 프로세스
Code 영역
Data 영역
Heap 영역
Stack 영역
공유
공유
공유
쓰레드 B
Code 영역
Data 영역
Heap 영역
B thread Stack 영역

프로세스와 쓰레드의 관계

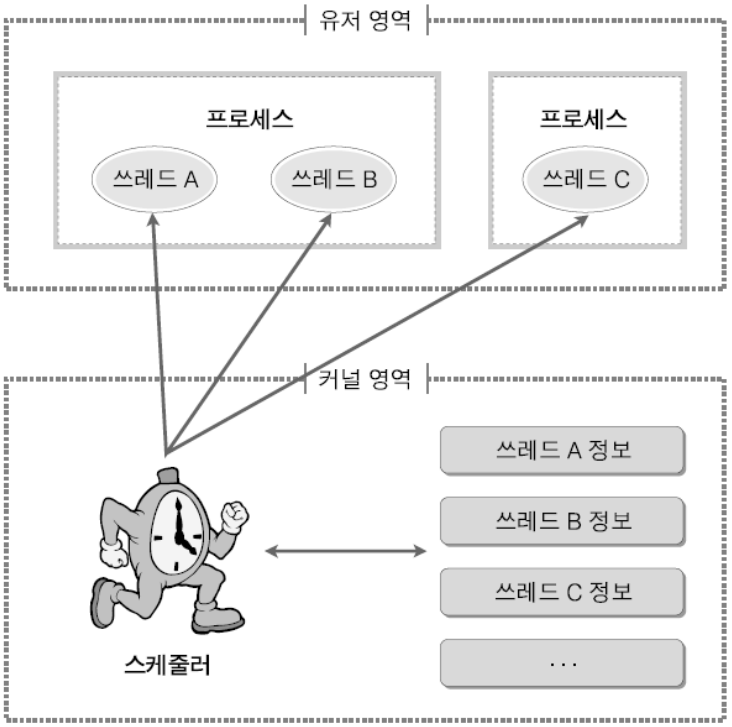


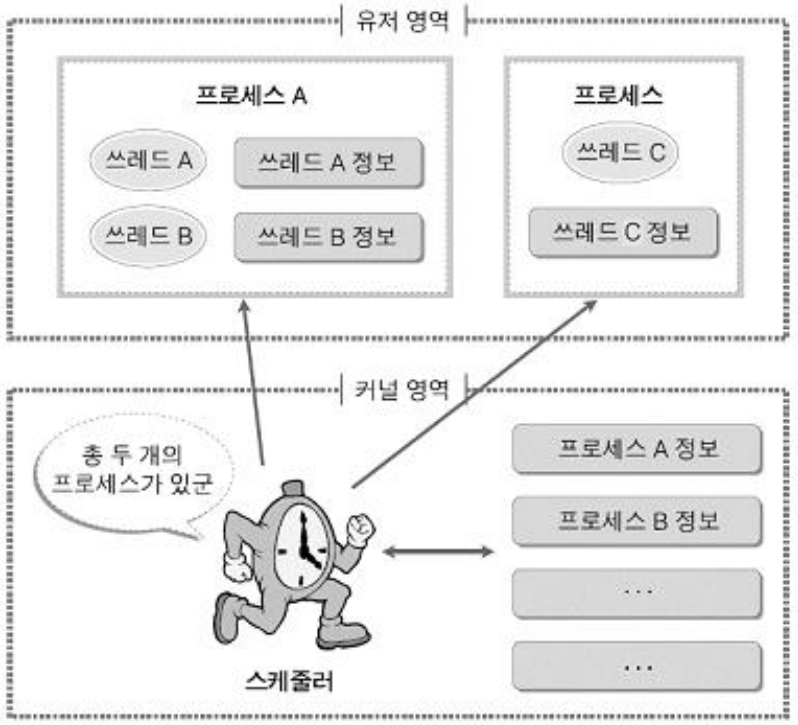
프로세스 & 쓰레드

Windows에서의 프로세스와 쓰레드

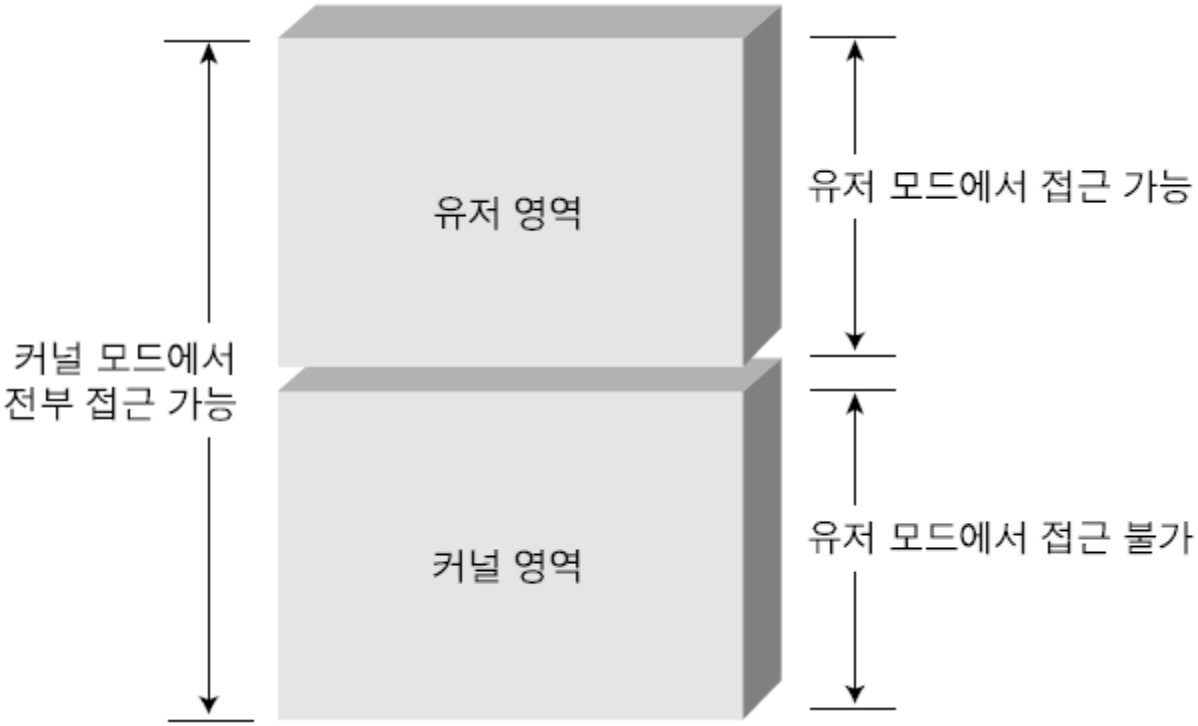


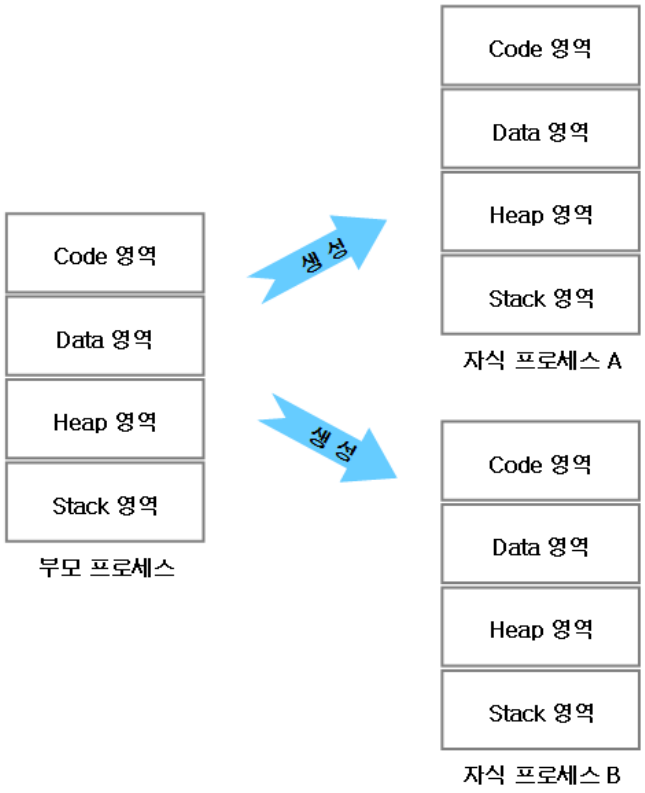
커널 레벨 쓰레드 vs. 유저 레벨 쓰레드

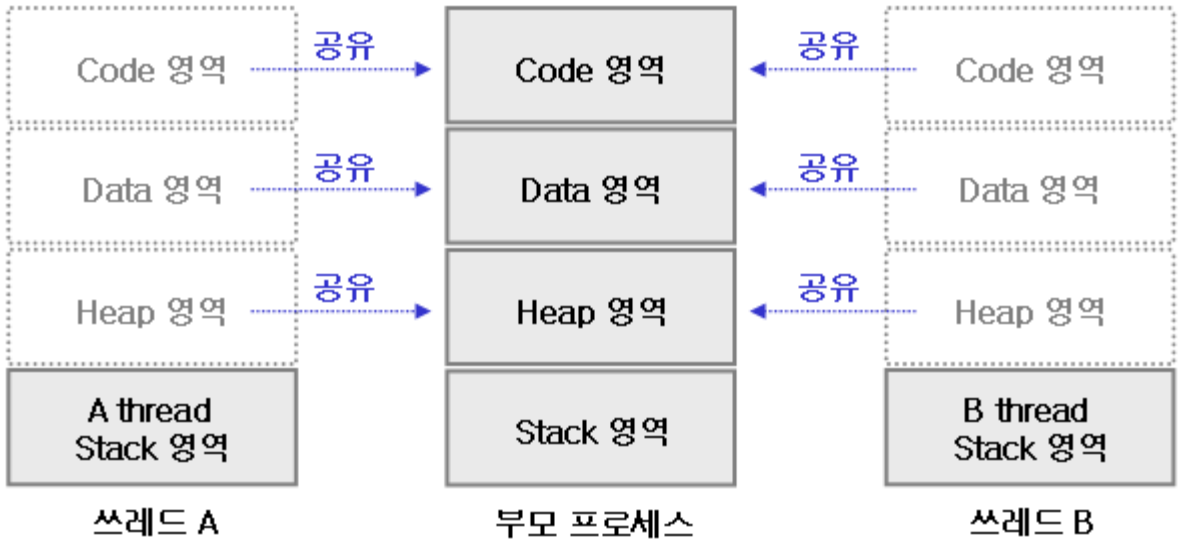


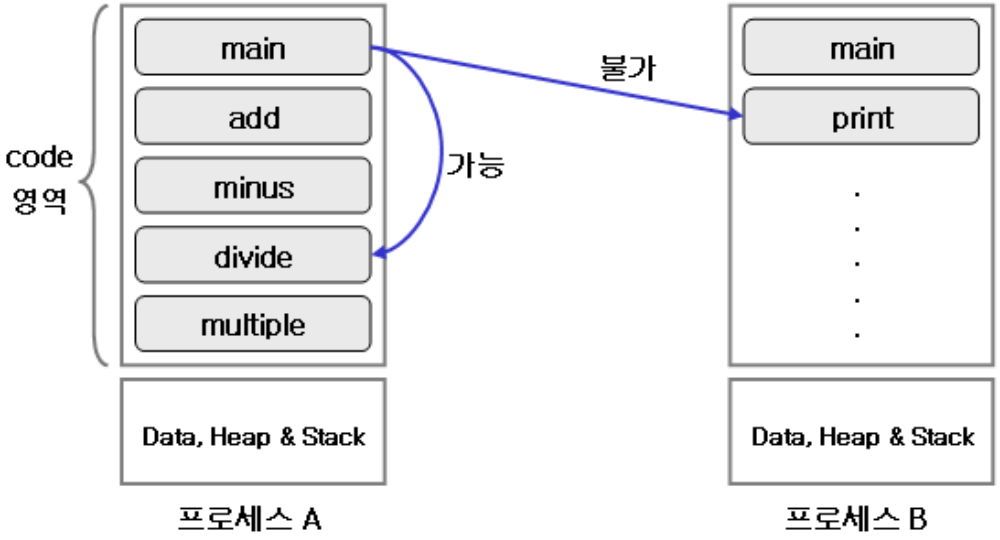


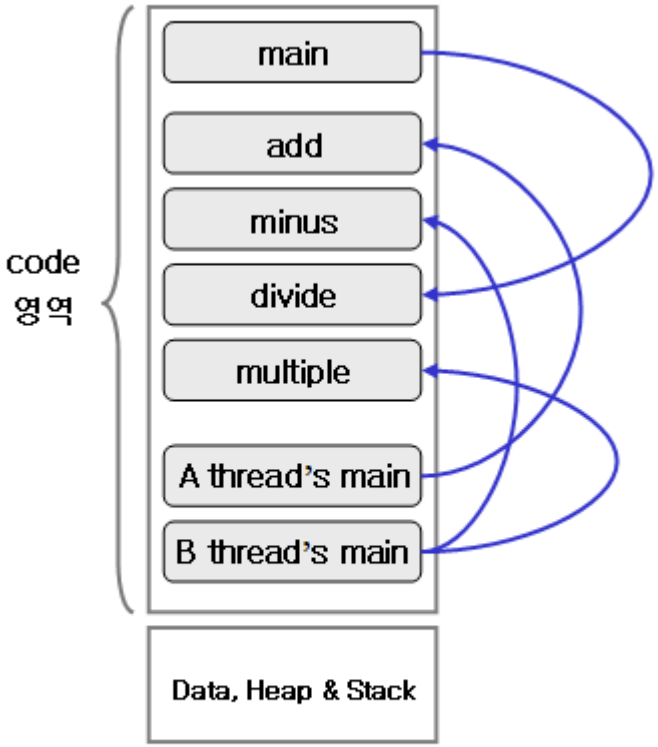
커널 모드와 유저모드

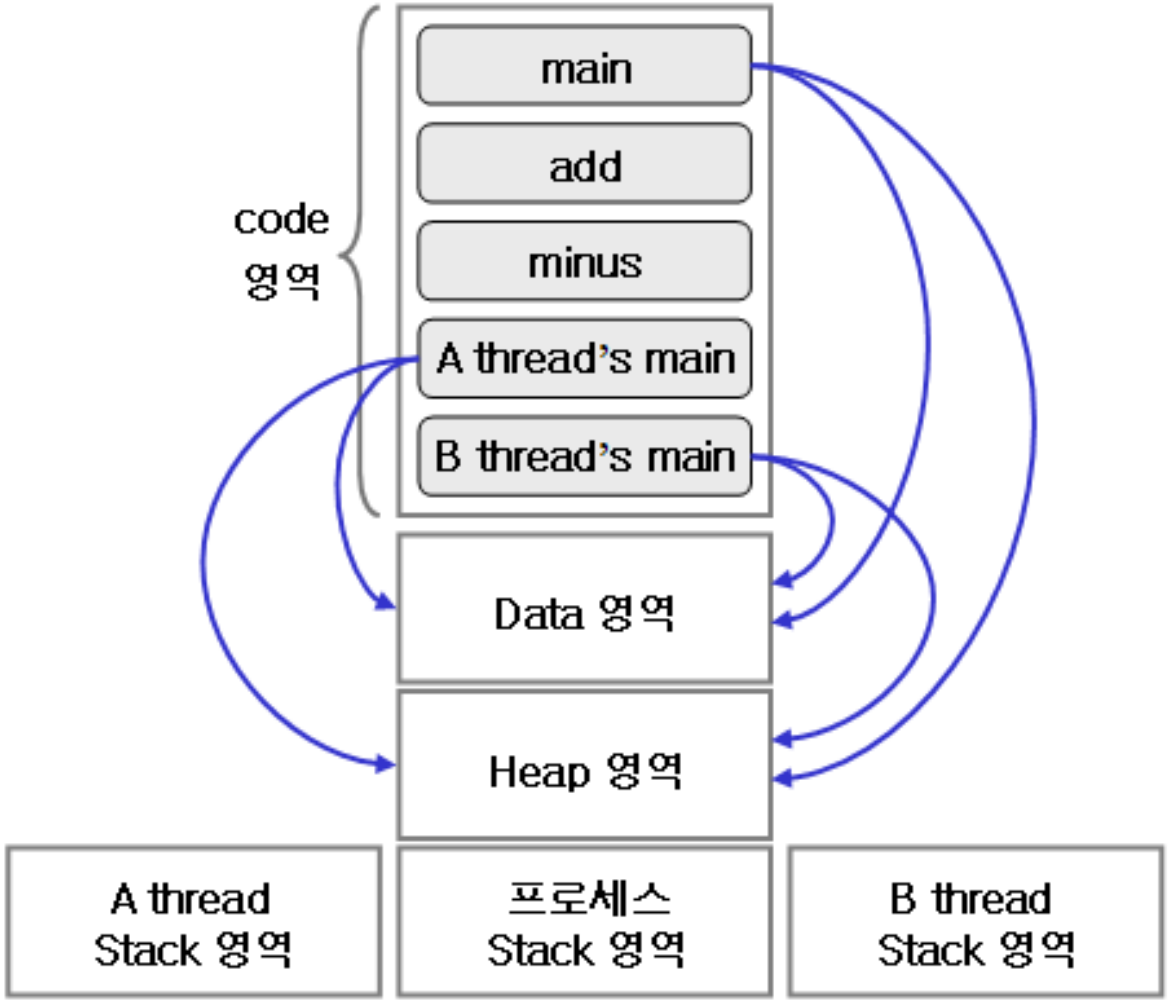


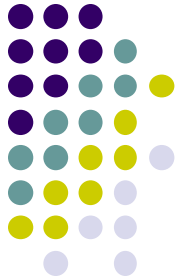

Code 영역
Data 영역
Heap 영역
Stack 영역
부모 프로세스
생성
생성
Code 영역
Data 영역
Heap 영역
Stack 영역
자식 프로세스 A
Code 영역
Data 영역
Heap 영역
Stack 영역
자식 프로세스 B


Code 영역
Data 영역
Heap 영역
A thread
Stack 영역
쓰레드 A
공유
공유
공유
Code 영역
Data 영역
Heap 영역
Stack 영역
부모 프로세스
공유
공유
공유
Code 영역
Data 영역
Heap 영역
B thread
Stack 영역
쓰레드 B


main
add
minus
divide
multiple
code
영역
Data, Heap & Stack
프로세스 A
불가
가능
main
print
Data, Heap & Stack
프로세스 B


main
add
minus
divide
multiple
A thread's main
B thread's main
code
영역
Data, Heap & Stack
프로세스 & 쓰레드


main
add
minus
A thread's main
B thread's main
code
영역
Data 영역
Heap 영역
A thread
Stack 영역
프로세스
Stack 영역
B thread
Stack 영역



장점과 단점

☞ **커널 레벨 쓰레드의 장점 및 단점**

장점: 커널에서 직접 제공해 주기 때문에 안전성과 다양한 기능성 제공

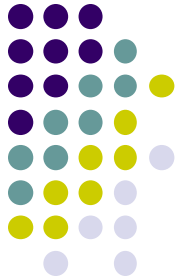
단점: 유저 모드에서 커널 모드로의 전환이 빈번

☞ **유저 레벨 쓰레드의 장점 및 단점**

장점: 유저 모드에서 커널 모드로의 전환이 필요 없다.

단점: 프로세스 단위 블로킹

쓰레드의 생성



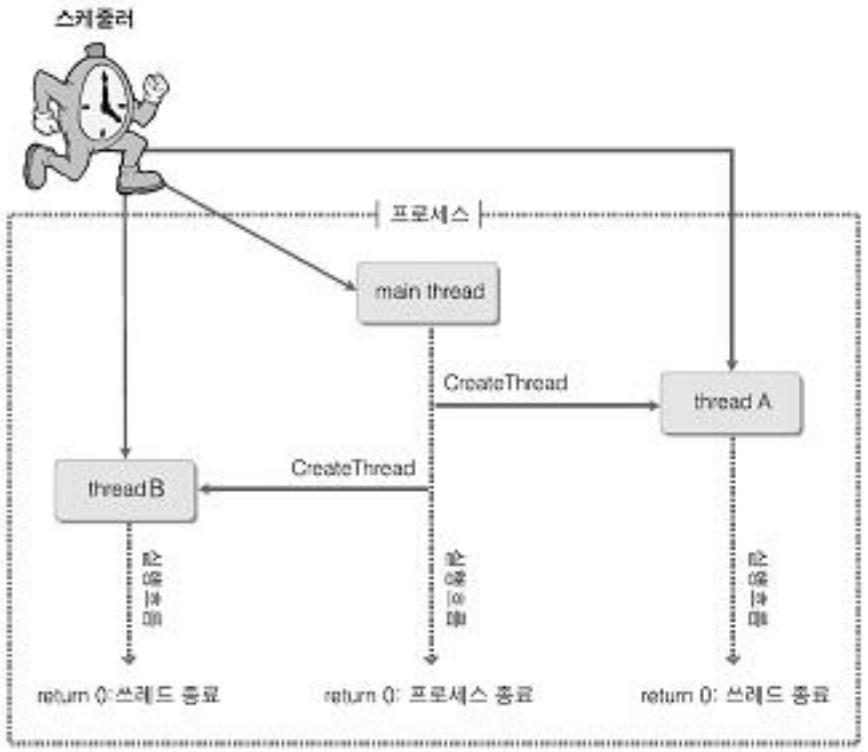
```
HANDLE CreateThread(
    LPSECURITY_ATTRIBUTES lpThreadAttributes,      // 핸들의 상속여부 결정
    SIZE_T dwStackSize,                            // initial stack size
    LPTHREAD_START_ROUTINE lpStartAddress,         // thread function
    LPVOID lpParameter,                            // thread argument
    DWORD dwCreationFlags,                         // creation option
    LPDWORD lpThreadId                             // thread identifier
);
```

If the function fails, the return value is NULL.

```
typedef  DWORD (WINAPI *PTHREAD_START_ROUTINE)(LPVOID lpThreadParameter);
```

생성 가능한 쓰레드 개수는?

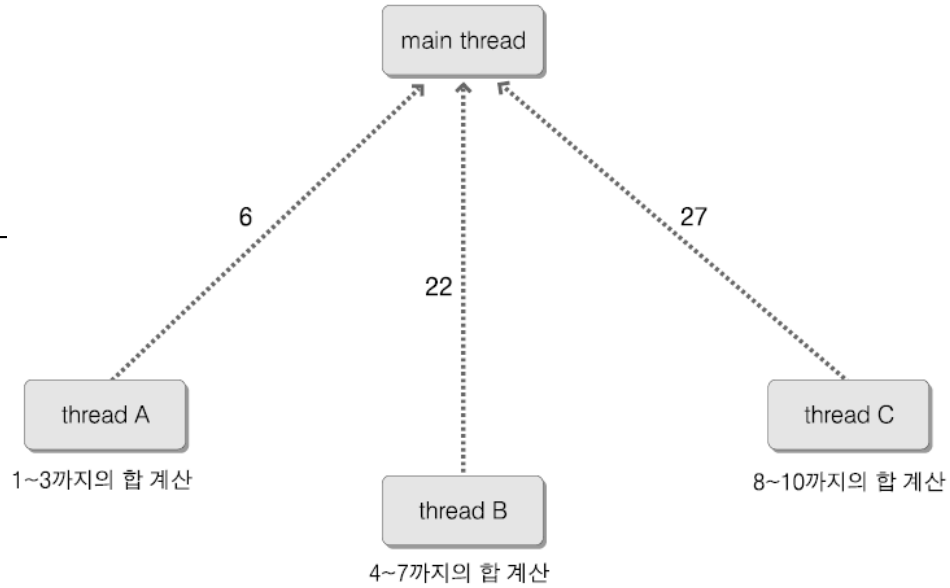
[예제 12-1] CountThread.cpp



쓰레드의 소멸 case1:

쓰레드 종료 시 return을 이용하면 좋은 경우(거의 대부분의 경우)

[예제 12-3] ThreadAdderOne.cpp



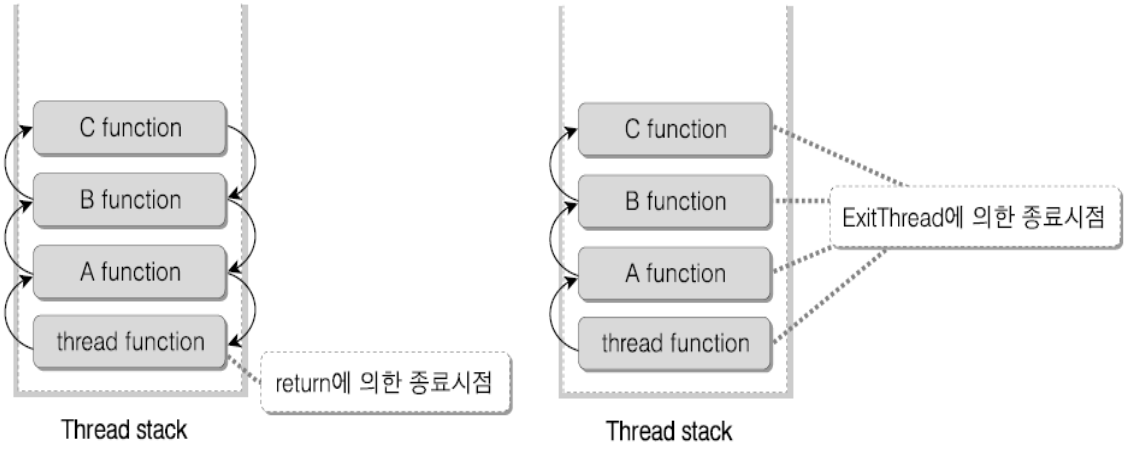

```
BOOL GetExitCodeThread (
    HANDLE hThread,
    LPDWORD lpExitCode
);
```

If the function fails, the return value is zero.

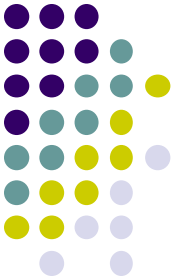
쓰레드의 소멸 case2:

쓰레드 종료 시 ExitThread 함수 호출이 유용한 경우

(특정 위치에서 쓰레드의 실행을 종료시키고자 하는 경우)

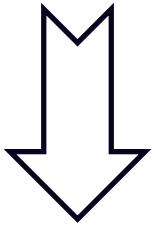


쓰레드의 소멸 case3:



쓰레드 종료 시 TerminateThread 함수 호출이 유용한 경우

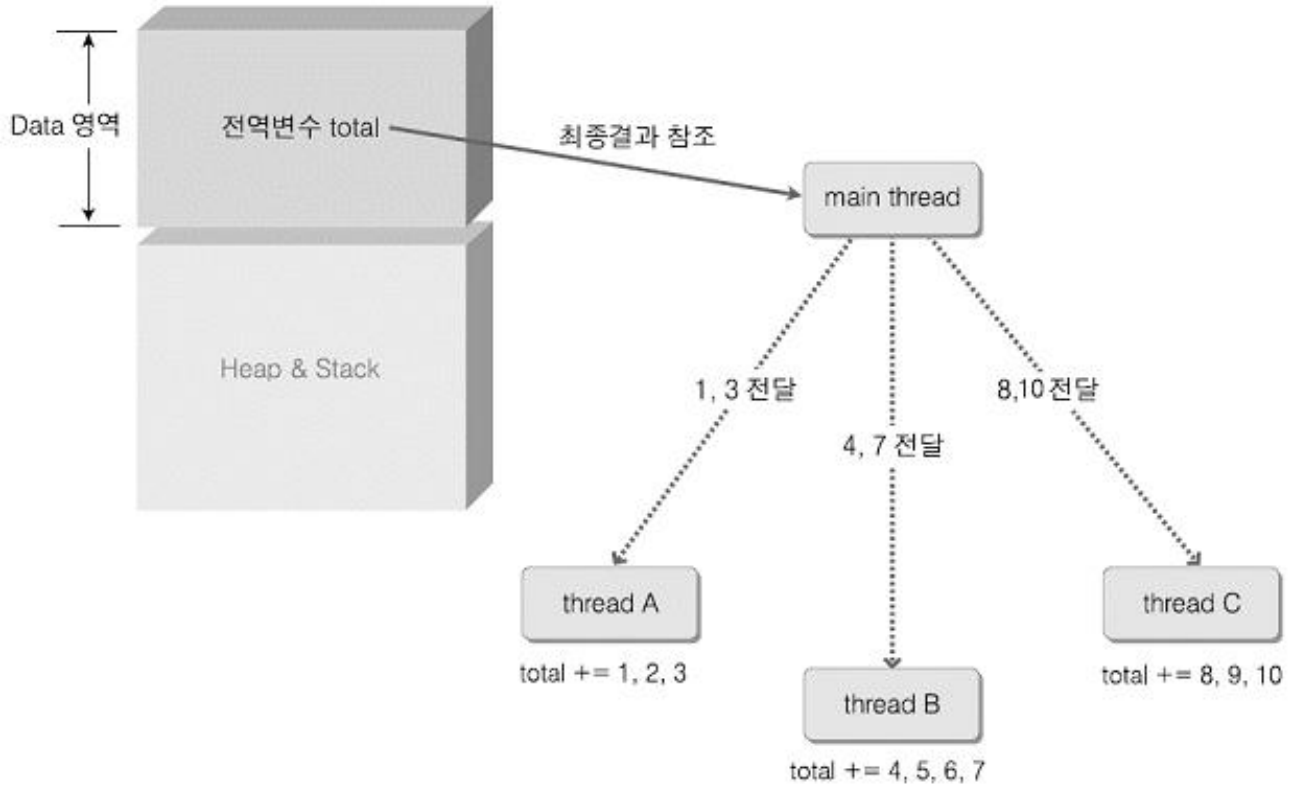
(외부에서 쓰레드 종료시키고자 하는 경우)



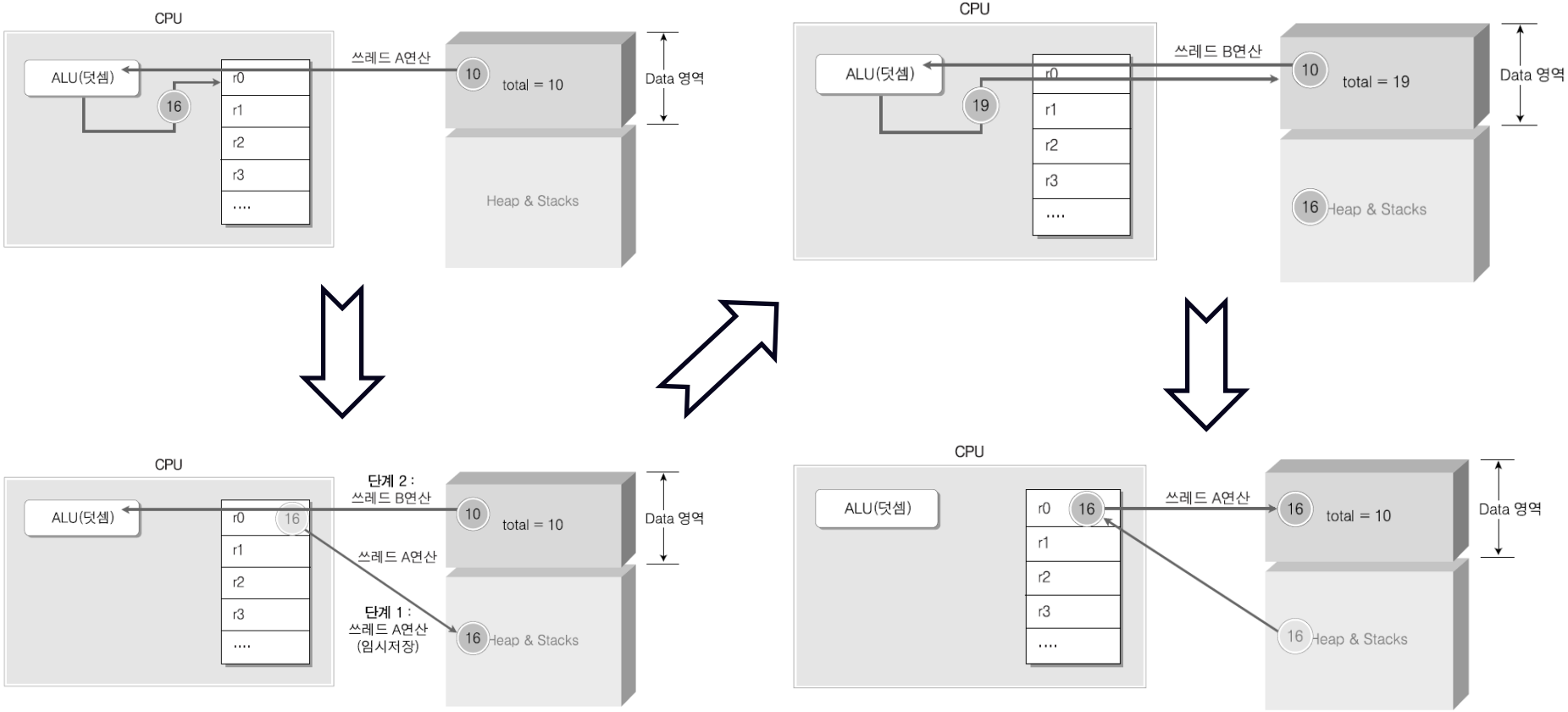
쓰지 말자

힙, 데이터, 코드 영역의 공유 검증

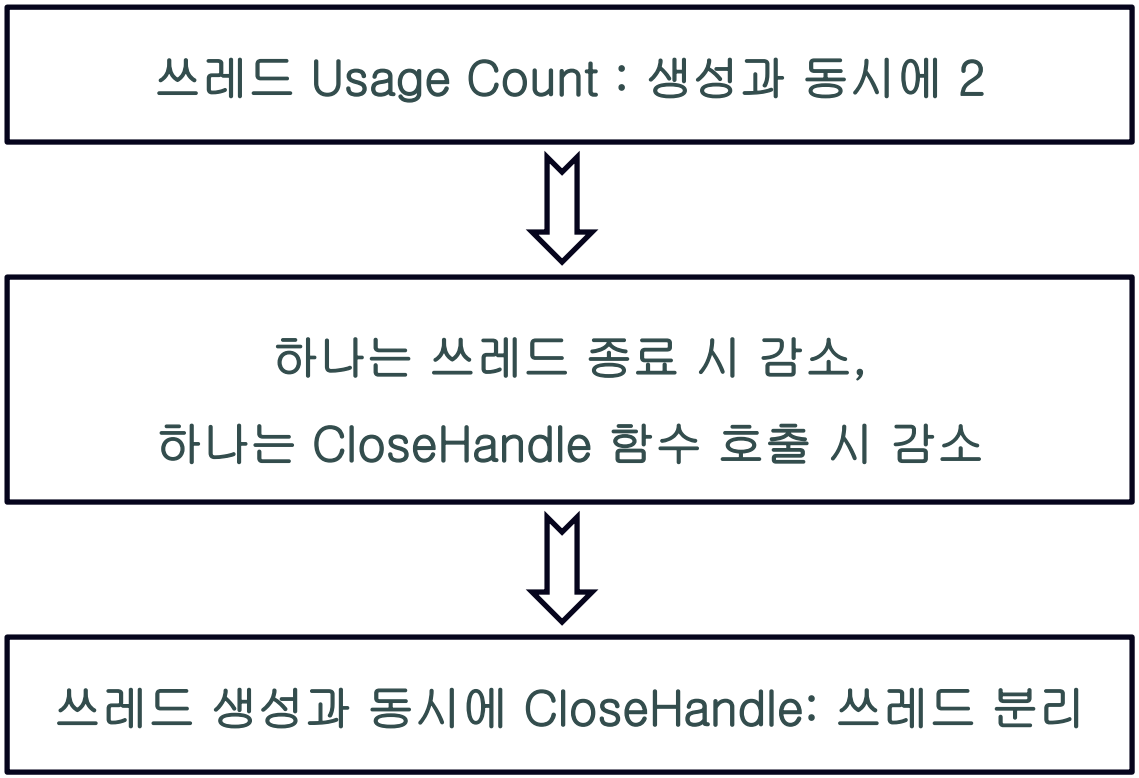
[예제 12-4] ThreadAdderTwo.cpp



동시접근의 문제점


CPU
ALU(덧셈)
16
r0
r1
r2
r3
....
쓰레드 A연산
10
total = 10
Heap & Stacks
Data 영역
CPU
ALU(덧셈)
19
r0
r1
r2
r3
....
쓰레드 B연산
10
total = 19
16
Heap & Stacks
Data 영역
CPU
ALU(덧셈)
r0
16
r1
r2
r3
....
단계 2 :
쓰레드 B연산
10
total = 10
쓰레드 A연산
단계 1 :
쓰레드 A연산
(임시저장)
16
Heap & Stacks
Data 영역
CPU
ALU(덧셈)
r0
16
r1
r2
r3
....
쓰레드 A연산
16
total = 10
16
Heap & Stacks
Data 영역

프로세스로부터의 쓰레드 분리.



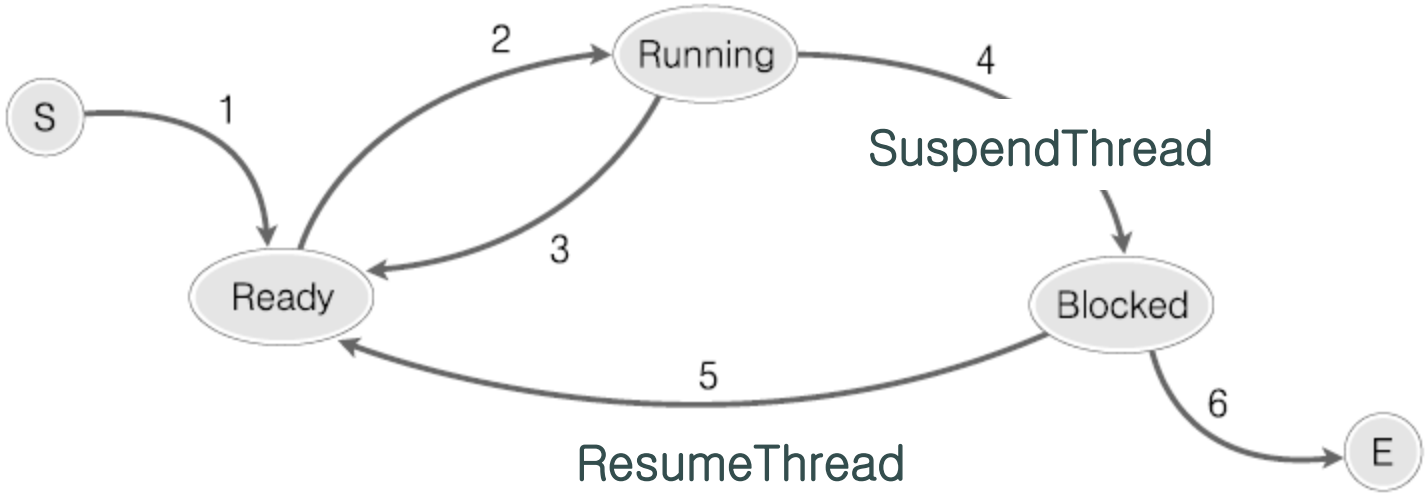
ANSI 표준 C 라이브러리와 쓰레드

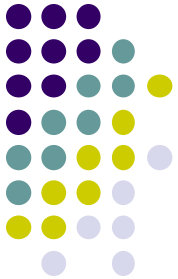
```
int _tmain(int argc, TCHAR* argv[])
{
    TCHAR string[] =
      _T("Hey, get a life!")
      _T("You don't even ... ...together.");

    TCHAR seps[] = _T(" ,.!");

    // 토큰 분리 조건, 문자열 설정 및 첫 번째 토큰 반환.
    TCHAR * token = _tcstok( string, seps );

    // 계속해서 토큰을 반환 및 출력.
    while( token != NULL )
    {
        _tprintf( _T(" %s₩n"), token );
        token = _tcstok( NULL, seps );
    }
    return 0;
}
```

CreateThread, ExitThread

_beginthreadex, _endthreadex

쓰레드의 상태 컨트롤


S
1
Ready
2
Running
3
4
SuspendThread
Blocked
5
ResumeThread
6
E

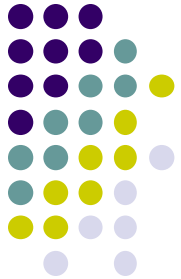


쓰레드의 우선순위 결정 요소

Priority	Meaning
IDLE_PRIORITY_CLASS	기준 우선순위 4
NORMAL_PRIORITY_CLASS	기준 우선순위 9 (Default)
HIGH_PRIORITY_CLASS	기준 우선순위 13
REALTIME_PRIORITY_CLASS	기준 우선순위 24

Priority	Meaning
THREAD_PRIORITY_LOWEST	−2
THREAD_PRIORITY_BELOW_NORMAL	−1
THREAD_PRIORITY_NORMAL	0 (Default)
THREAD_PRIORITY_ABOVE_NORMAL	+1
THREAD_PRIORITY_HIGHEST	+2

쓰레드의 우선순위 컨트롤 함수



```
BOOL SetThreadPriority (
    HANDLE hThread,
    int nPriority
);
```

If the function fails, the return value is zero.

```
int GetThreadPriority (
    HANDLE hThread
);
```

If the function fails, the return value is THREAD_PRIORITY_ERROR_RETURN

실습 1

- 10개의 숫자를 입력받아 값들을 출력한 후, 합계를 구할 때 3개의 스레드를 사용하여 프로그램을 작성하시오.