





프로세스와 스레드 [2016.06.30]

- 1 프로세스
- 2 멀티태스킹 시스템
- 3 스레드
 - 3.1 예시 - AutoMerge Tool
 - 3.2 스레드의 종류
 - 3.3 PolarisOffice Engine 내부 스레드
- 4 프로세스와 스레드의 차이
- 5 멀티스레드 프로그램
- 6 멀티태스킹과 멀티스레드
- 7 멀티코어 프로그래밍
- 8 출처

프로세스

하드 디스크에 저장되어 있는 EXE 파일을 보통 프로그램이라고 부른다.

 CollaborationBuilder.exe	2016-06-15 오후...	응용 프로그램	6,100KB
 IssueAutoMerge.exe	2015-12-29 오전...	응용 프로그램	5,354KB
 PCOfficeAutoMerge.exe	2016-05-24 오전...	응용 프로그램	5,931KB
 PolarisOfficeBuilder.exe	2016-06-03 오전...	응용 프로그램	7,225KB

프로그램이 실행되어 동작되고 있으면 이를 프로세스 혹은 태스크 라고 부른다.

프로세스를 인스턴스라고 일컫기도 하나, 인스턴스는 프로그램이 메모리에 로딩되어 있는 상태를 말하는 것이고, 프로세스는 메모리상에서 실행되고 있는 상태를 말하는 것이다.

- 프로세스는 **실행 중**이라는 것이 초점을 두고 있고, 인스턴스는 실행하기 위해서 **메모리에 생성**되는 것에 초점. 결론은 같지만 관점의 차이
- ex) 태스크 - 일감 / 스레드 - 일꾼 / 프로세스 - 응용프로그램의 실행 단위

Windows 작업 관리자

파일(F) 옵션(O) 보기(V) 도움말(H)

응용 프로그램 | **프로세스** | 서비스 | 성능 | 네트워크 | 사용자

이미지 이름	PID	사용자...	CPU	메모리(...)	핸들	스레드	설명
OUTLOOK.EXE	440	songjh	00	108,432 ...	4,700	44	Microso...
pino.exe *32	7140	songjh	00	6,396 KB	310	20	Pino
pinomate.exe *32	3736	songjh	00	1,044 KB	60	2	pinomate
PolarisOffice.exe *32	7636	songjh	00	67,420 KB	649	24	Polaris ...
POSyncCenter.exe *32	3308	songjh	00	15,152 KB	358	7	Polaris ...
PSheet.exe *32	3116	songjh	00	65,080 KB	661	29	Polaris ...
PSheet.exe *32	8656	songjh	00	23,184 KB	474	17	Polaris ...
PSNoticeChecker.exe *32	4336	songjh	00	3,588 KB	313	7	Polaris ...
PWord.exe *32	9432	songjh	00	78,712 KB	673	31	Polaris ...
PWord.exe *32	10176	songjh	00	47,972 KB	421	12	Polaris ...
SearchProtocolHost.exe	2176	songjh	00	3,012 KB	594	12	Microso...
SnippingTool.exe	6820	songjh	00	2,784 KB	151	11	캡처 도구
StikyNot.exe	3252	songjh	00	5,992 KB	170	8	스티커 ...
taskhost.exe	3564	songjh	00	10,604 KB	348	12	Window...
taskmgr.exe	9760	songjh	00	3,952 KB	150	8	Window...
TortoiseProc.exe	4888	songjh	00	20,808 KB	364	6	Tortoise...
TSVNCosha.exe	3124	songjh	00	31,024 KB	156	14	Tortoise...

모든 사용자의 프로세스 표시(S) | 프로세...

프로세스: 97 | CPU 사용: 23% | 실제 메모리: 70%

멀티태스킹 시스템

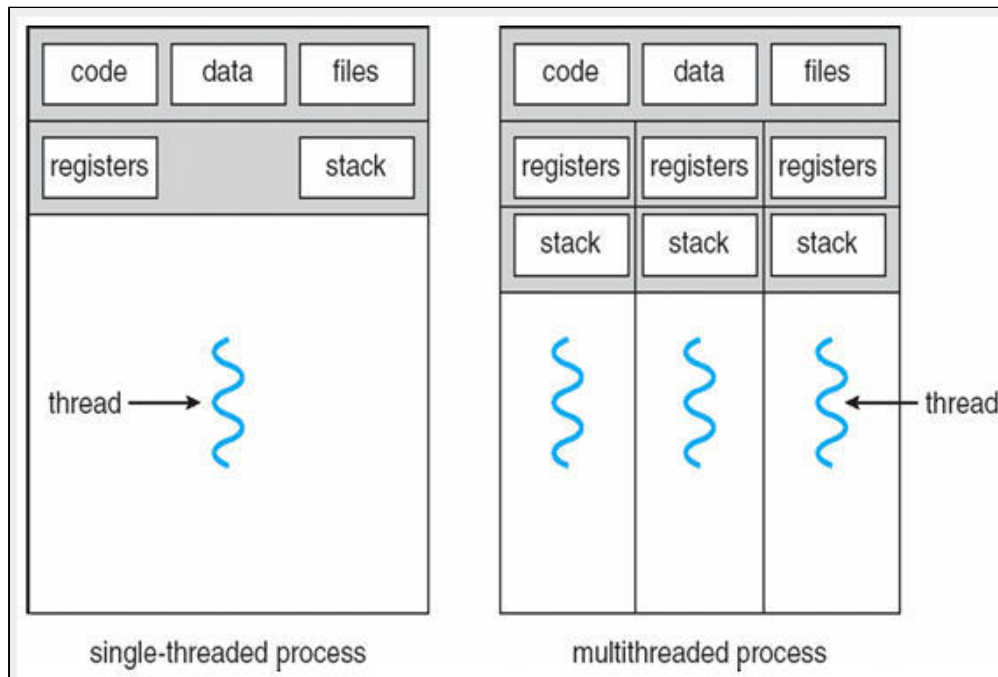
윈도우는 멀티태스킹 시스템이다. 여러 개의 프로세스를 동시에 실행시킬 수 있는 시스템이라는 뜻이다.

멀티태스킹은 사실 여러 개의 프로세스가 동시에 실행되는 것이 아니라, 아주 짧은 시간을 여러 조각으로 나누어 여러 개의 프로세스를 돌아가면서 실행하는 것이다.

- 장점
 - 하나의 프로세스 동작이 다 끝날 때까지 기다리지 않고, 여러 개의 프로세스 실행시킬 수 있다. -> OS에서 스케줄링이 필요함
 - CPU의 활용도 증가
- 단점
 - 복잡한 메모리 관리 시스템 -> 동시에 여러 개의 프로그램이 메모리에 상주되므로 메모리 관리 및 보호 시스템 필요함
 - 적절한 응답 시간을 제공해야 함

스레드

1. 스레드란



- a. 프로세스 내의 실행 흐름이며, 프로세스보다 작은 단위
- b. **스레드**(thread)는 어떠한 프로그램 내에서, 특히 **프로세스** 내에서 실행되는 흐름의 단위를 말한다. 일반적으로 한 프로그램은 하나의 스레드를 가지고 있지만, 프로그램 환경에 따라 둘 이상의 스레드를 동시에 실행할 수 있다. 이러한 실행 방식을 **멀티스레드**(multithread)라고 한다.

2. 예시 - AutoMerge Tool

```
void CAutoMergeDlg::OnSemiAutoMerge()
{
    ThreadStop(TRUE);
    m_bisThreadRunning = true;
    m_pThread = AfxBeginThread( MergeThreadFunction, this, THREAD_PRIORITY_NORMAL, 0,
    CREATE_SUSPENDED);
    m_pThread->m_bAutoDelete = FALSE;
    m_pThread->ResumeThread();
}
```

```

UINT MergeThreadFunction(LPVOID lpParam)
{
    CAutoMergeDlg* pClass = (CAutoMergeDlg*)lpParam;

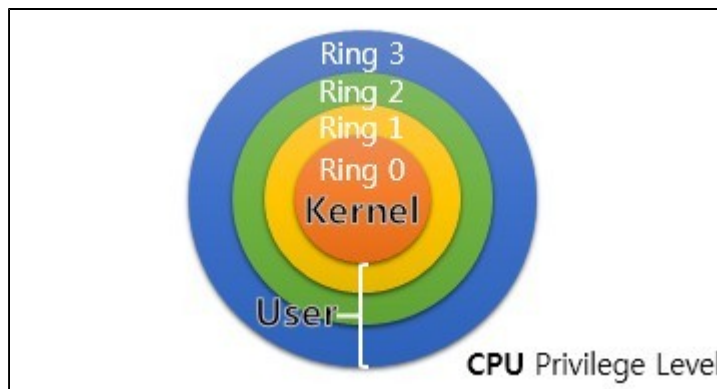
    pClass->_OnAutoMerge();
    return 0;
}

```

MSDN AfxBeginThread() : <https://msdn.microsoft.com/ko-kr/library/s3w9x78e.aspx> - 이 함수를 호출하여 새 스레드를 만듭니다.

매개 변수	설명
AFX_THREADPROC pfnThreadProc	작업자 스레드에 대한 제어 함수를 가리킵니다. NULL 이 될 수 없습니다. 이 함수는 다음과 같이 선언해야 합니다. UINT __cdecl MyControllingFunction(LPVOID pParam);
LPVOID pParam	pfnThreadProc의 함수 선언에 매개 변수가 전달되는 것처럼 기능 제어에 매개 변수가 전달됩니다.
int nPriority	스레드의 원하는 우선 순위. 사용 가능한 우선 순위의 전체 목록과 설명을 보려면 Windows SDK에서 SetThreadPriority 를 참조하십시오.
UINT nStackSize	새 스레드의 스택 크기를 지정합니다(바이트 단위). 0인 경우 스택 크기 기본값은 만드는 스레드와 스택의 크기가 동일합니다.
DWORD dwCreateFlags	스레드 생성을 제어하는 추가 플래그를 지정합니다. 이 플래그는 두 값 중 하나를 포함할 수 있습니다. a. CREATE_SUSPENDED 일시 중단 횟수 1로 스레드를 시작합니다. 스레드가 실행되기 전에 CWinThread 개체의 모든 멤버 데이터를 초기화하려면 (예: m_bAutoDelete 또는 파생 클래스의 모든 멤버) CREATE_SUSPENDED 를 사용합니다. 초기화가 완료되면 CWinThread::ResumeThread 를 사용하여 스레드 실행을 시작합니다. 스레드는 CWinThread::ResumeThread 가 호출될 때까지 실행되지 않습니다. b. 0 스레드를 만든 후 즉시 시작합니다.
LPSECURITY_ATTRIBUTES lpSecurityAttrs	스레드의 보안 특성을 지정하는 SECURITY_ATTRIBUTES 를 가리킵니다. NULL 인 경우 스레드 생성과 동일한 보안 특성이 사용됩니다. 이 구조에 대한 자세한 내용은 Windows SDK를 참조하십시오.

3. 스레드의 종류



- a.
- b. **유저 스레드 (User Thread)**
 - i. 사용자 스레드는 커널 영역의 상위에서 지원되며 일반적으로 사용자 레벨의 라이브러리를 통해 구현되며, 라이브러리는 스레드의 생성 및 스케줄링 등에 관한 관리 기능을 제공한다. 동일한 메모리 영역에서 스레드가 생성 및 관리되므로 속도가 빠른 장점이 있다.
 - ii. context switching의 오버헤드가 커널 스레드일 때보다 적다.
 - iii. 커널에서 볼 때는 프로세스 내부가 보이지 않아 통째로 하나로 인식한다. 따라서 스레드 하나가 커널에 진입할 경우 해당 프로세스를 Blocked 큐에 보낸다.
- c. **커널 스레드 (Kernel Thread)**
 - i. 커널 스레드는 운영체제가 지원하는 스레드 기능으로 구현되며, 커널이 스레드의 생성 및 스케줄링 등을 관리한다. 스레드가 시스템 호출 등으로 중단되더라도, 커널은 프로세스 내의 다른 스레드를 중단시키지 않고 계속 실행시켜준다. 다중처리 환경에서 커널은 여러 개의 스레드를 각각 다른 처리기에 할당할 수 있다. 다만, 사용자 스레드에 비해 생성 및 관리하는 것이 느리다.
 - ii. context switching 시 유저모드->커널모드로 전환해야 하는 작업이 추가로 더 필요하다.
 - iii. 각각의 스레드들을 커널이 직접 관리하기 때문에, 스레드 중 하나가 Blocked 큐로 들어간다고 해도 그 프로세스 내의 다른 스레드들은 Ready 큐나, Running 큐에 머무를 수 있다.

4. PolarisOffice Engine 내부 스레드

- a. 엔진 내부 스레드의 종류는 크게 4가지 종류가 있다.
- i. LongProcess Thread : 일반적인 UI 이벤트 처리를 위해 생성한다.. (ex. Open, Save, Thumbnail, Find, ...)

```
void RunMainQ( BoraThreadTraits::context_type& context, LPBrBaseEventType
pBaseEvent, CInterfaceProc* Q)
{
    /* 중략 */
    switch(pBaseEvent->nType)
    {
    case BROPEN_EVENT:
    {
        B_ResetErrorCode();
        LPBrOpenEventType pEvent = (LPBrOpenEventType)pBaseEvent;
        B_ReadyLoadStatus();
        #if defined(RENDERING_WITH_BORATHREAD)
        #if defined(USE_THUMBNAI_THREAD)
        if ( pEvent->bDirectOpen )
        {
            CLongProcessThread m_Thread(context, (LPBrBaseEventType)pEvent);
            m_Thread.Run(context);
        }
        else
        #endif //defined(USE_THUMBNAI_THREAD)
        start_longprocess(pBaseEvent);
        }
        break;
    /* 중략 */
    }
```

- ii. CacheThread : Engine 자체 판단으로 생성하며, Slide/PDF 포맷에서 페이지 캐싱을 위해 사용한다.
 - iii. BGLoad Thread : Engine 자체 판단으로 생성하며, Word/Sheet 포맷에서 백그라운드 로딩을 위해 사용한다.
 - iv. GeneralThread : 거의 사용하지 않음. 파일브라우저 문서 미리보기 이미지 생성 시에 사용함.
- b. 복수의 엔진 내부 스레드가 존재할 수 있지만, 각 스레드가 동시에 수행되지 않는다.
- c. 특정 주기마다 UI Timer에 의해 구동된다.
- i. Engine에서 Sub-Thread를 생성하며 UI Timer를 On
 - ii. 엔진에 Timer 이벤트가 들어오면 Sub-Thread로 switching
 - iii. Sub-Thread는 수행 도중 Yield 코드를 만나거나 작업이 완료되면 Main-Thread로 Switching
 - iv. Sub-Thread의 작업이 완료되거나 취소되는 경우, 엔진에서 UI Timer를 Off

프로세스와 스레드의 차이

차이점 분류	프로세스	스레드
-----------	------	-----

수행 범위/범주	하나의 실행 흐름을 가짐	하나의 프로세스 안에 여러 개의 스레드가 존재
메모리	프로세스 간의 메모리는 독립적이므로, 서로의 영역에 접근하기 어려움.	프로세스의 code영역과 data영역은 스레드간에 공유
switching	프로세스간의 switch 비용이 큼 (상대적으로 Heavy weight)	스레드들은 같은 메모리영역을 사용하므로 스레드간 switch light weight)



Context Switching

실행하고 있는 프로그램 혹은 프로세스를 교환하는 것. 실행에 이용되는 프로그램 카운터, 스택 포인터, 레지스터 등의 내용을 넣어두고, 거기까지 실행해 간 프로세스 등의 실행에 필요한 정보를 보존하여 다음 실행을 시작하는 프로세스 등의 정보를 이용할 수 있게 하는 조작.

ex) PolarisOffice Engine<->UI 간의 thread switching

멀티스레드 프로그램

1. 멀티스레드 프로그램이란

주 프로세스 외에 하나 이상의 스레드를 가지고 동작하는 프로그램을 멀티스레드 프로그램이라고 한다.

프로그램에서 한 가지 이상의 일을 동시에 수행해야 할 때, 단일 프로세스로 프로그램을 구현하는 것보다 멀티스레드로 구현하면 프로그램을 훨씬 부드럽게 동작하도록 할 수 있다.

2. 멀티스레드의 장점

- Responsiveness
 - 프로그램의 어느 부분을 담당하는 스레드가 block되거나 시간이 걸리는 작업을 하더라도, 다른 스레드들은 실행되기 때문에 user 입장에서 반응성이 보장된다.
- Resource sharing
 - 스레드 간에는 프로세서의 메모리와 다른 자원들을 공유한다.
- Economy
 - 스레드들은 하나의 프로세스 메모리 영역에서 실행하기 때문에 새로운 프로세스를 생성하는 것보다 스레드를 생성하는 것이 비용이 적게 들어간다.
- Scalability(확장성), parallelism
 - 여러 개의 스레드가 각각 다른 프로세서에서 동시에 실행이 가능하다.

3. 멀티스레드의 단점

- consistency 유지 -> 동기화 문제가 발생한다.
- debugging이 어렵다.
- thread가 많은 경우 thrashing 발생한다. -> OS는 열심히 일하고 있는데 context switching에 시간을 많이 쏟고 있어 정작 유효한 일을 할 수 없는 상태를 말한다.

4. 예시

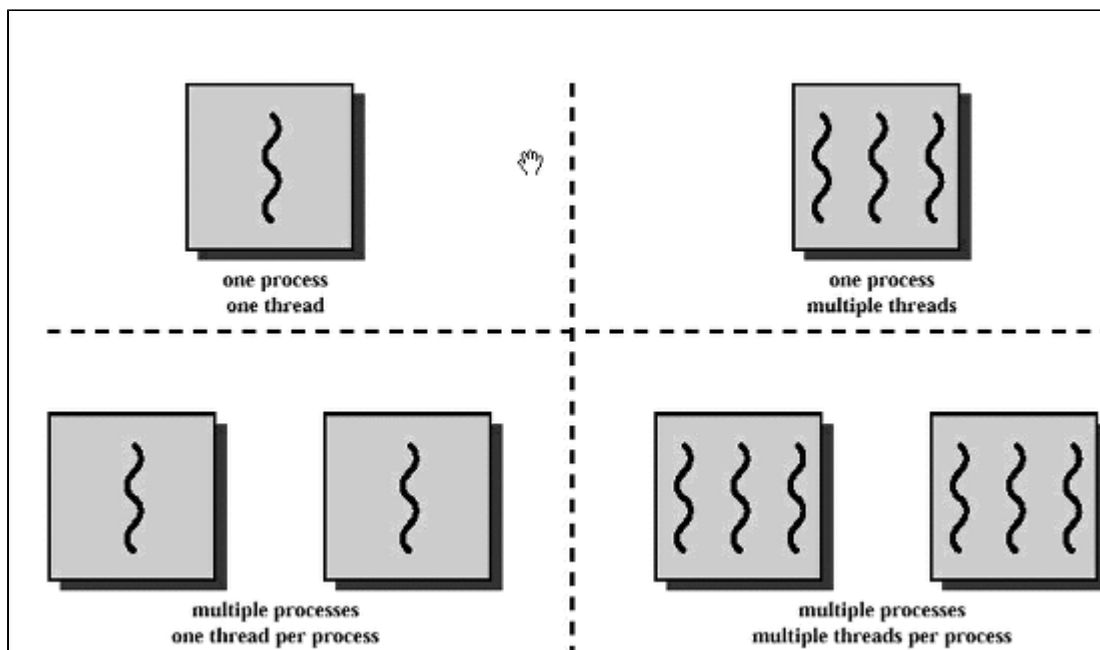
스레드					
검색:	X 호출 스택 검색		그룹화 방법(Y): 프로세스 ID		
	ID	관리 ID	범주	이름	위치
▲ 프로세스 ID: 0x00001CC4 (22 스레드)					
▼	0x000020C8	0x00	주 스레드	주 스레드	POfficeBase.dll!CEngine::Ge
▼	0x00002BF4	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!7702016d
▼	0x00002690	0x00	작업자 스레드	msvcr110d.dll!_threadstartex	mfc110ud.dll!AfxInternalPur
▼	0x000016DC	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56
▼	0x00000920	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56
▼	0x00001E70	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56
▼	0x000022C8	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56
▼	0x00002118	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56
▼	0x0000298C	0x00	작업자 스레드	wininet.dll 스레드	wininet.dll!74fd8857
▼	0x000021AC	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56
▼	0x0000119C	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56
▼	0x000020E4	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56
▼	0x00001F6C	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56
▼	0x00001B48	0x00	RPC 스레드	RPC 콜백 스레드	ole32.dll!768dd95d
▼	0x00000EDC	0x00	작업자 스레드	crypt32.dll!ILS_WaitForThreadProc()	crypt32.dll!FreeDllWaitForCa
▼	0x000026D4	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56
▼	0x00002B00	0x00	작업자 스레드	GdiPlus.dll 스레드	GdiPlus.dll!71f07901
▼	0x000014DC	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56
▼	0x000001D8	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56
▼	0x000025D4	0x00	작업자 스레드	ntdll.dll 스레드	ntdll.dll!77021f56

멀티태스킹과 멀티스레드

멀티태스킹과 멀티스레드는 모두, 아주 짧은 시간 간격을 두고 여러 개의 프로세스를 번갈아 실행됨으로써 동시에 여러 개의 프로세스가 실행되는 것처럼 보이게 한다.

차이점은 멀티태스킹은 동시에 여러 개의 프로그램을 실행시키는 것이고, 멀티스레딩은 하나의 프로그램을 여러 개의 기능으로 나누고 이를 동시에 실행시킨다는 것이다.

또한 멀티태스킹은 win32환경에서 작성된 모든 애플리케이션의 경우 윈도우 운영체제가 알아서 처리하지만, 멀티스레드는 프로그래머가 직접 구현을 해야 이용할 수 있다.



멀티코어 프로그래밍

1. 멀티코어 프로그래밍

- 멀티코어 프로세서는 운영체제에서 각각의 core를 하나의 프로세서로 인식하고 스케줄링함.
- 각각의 thread에 core를 할당하여 실행 가능.
- 멀티 프로세서가 달린 멀티코어는 cache를 공유하기 때문에 data, code등 프로세스의 자원을 공유하는 멀티스레드 프로그래밍에 보다 효과적임.

2. PolarisOffice Engine에서의 멀티코어 스레드

- 현재 엔진에서는 multi core를 활용하여 thread를 추가로 생성할 경우 pthread를 하나만 생성해서 사용한다.
- 하나의 큰 실행흐름의 구간에서 thread를 생성한다.
- 주로 렌더링, 문서 저장할 때 생성하게 되며, debug 창으로 봤을 때 작업자 스레드들 중 Base 함수가 workProc()으로 시작하는 경우 multicore thread이다.

스레드

검색:

✖ 호출 스택 검색

▼ ▾ ▴

 그룹화 방법(Y):

	ID	관리 ID	범주	이름	위치
^ 프로세스 ID: 0x00000E70 (25 스레드)					
▼	0x000015C8	0x00	주 스레드	주 스레드	▼ PolarisOffice_1
▼	0x00001E08	0x00	작업자 스레드	ntdll.dll 스레드	▼ ntdll.dll!76f001
▼	0x00001278	0x00	작업자 스레드	ntdll.dll 스레드	▼ ntdll.dll!76f01f
▼	0x00001180	0x00	작업자 스레드	ntdll.dll 스레드	▼ ntdll.dll!76f01f
▼	0x00001B50	0x00	작업자 스레드	ntdll.dll 스레드	▼ ntdll.dll!76f01f
▼	0x00000A84	0x00	작업자 스레드	ntdll.dll 스레드	▼ ntdll.dll!76f01f
▼	0x00000E0C	0x00	작업자 스레드	ntdll.dll 스레드	▼ ntdll.dll!76f01f
▼	0x00002210	0x00	작업자 스레드	ntdll.dll 스레드	▼ ntdll.dll!76f01f
▼	0x00001B18	0x00	작업자 스레드	msvcrt.dll!_endthreadex()	^ pthreadVC2.dll ntdll.dll!76f001 [아래 프레임은 ntdll.dll!76f001 KernelBase.dll! pthreadVC2.dll pthreadVC2.dll pthreadVC2.dll pthreadVC2.dll pthreadVC2.dll PolarisOffice_1 PolarisOffice_1 pthreadVC2.dll msvcrt.dll!_er msvcrt.dll!_er kernel32.dll!76 ntdll.dll!76f199 ntdll.dll!76f198

d.

출처

VISUAL C++ 6 완벽 가이드 / 김용성 저

2016 신입사원 교육 발표자료 - Thread, Multi-core, Memory.pptx

<http://boanin.tistory.com/84><http://ingorae.tistory.com/473><https://msdn.microsoft.com/ko-kr/library/s3w9x78e.aspx><http://divineprocess.tistory.com/entry/Multi-Thread-%EA%B5%AC%EC%B6%95><http://terms.naver.com/entry.nhn?docId=818893&cid=50376&categoryId=50376><https://ko.wikipedia.org/wiki/%EC%8A%A4%EB%A0%88%EB%93%9C>

<http://algorithmsandme.in/2014/12/operating-systems-thread-models-using-user-and-kernel-space-threads/>