작성자(학번 이름) : 2019180052 한창우

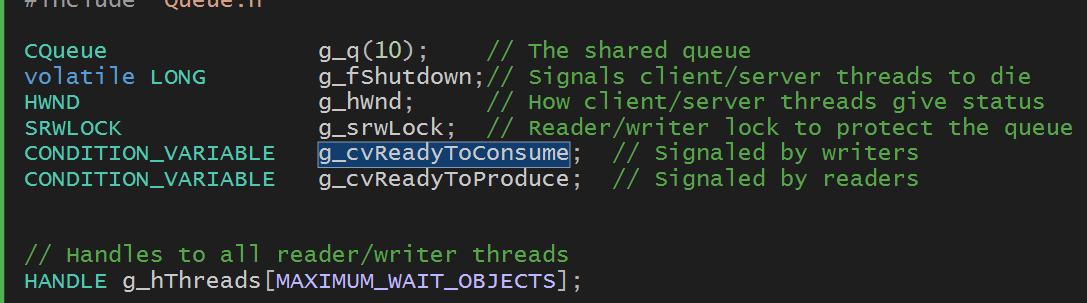
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **주차** | 8주차 | **기간** | 2024.8. 12~ 8. 18 | **지도교수** |  |
| 이번주 한일 요약 | Windows Via C/C++  챕터 9 “커널 모드 스레드 동기화” | | | | |

<상세 수행내용>

Queue예제 애플리케이션 실습 **(챕터8 실습코드)**

한 개의 SRWLock과 두개의 조건변수를 이용하여 요청 큐를 제어하는 실습 코드이다.

이번에 실습하는 코드의 상황은 다중 쓰레드 환경에서 **“생산자-소비자 패턴”**을 이용하여, 자원이 준비될 때까지 소비자가 대기하는 상황이다. 소비자는 큐가 비어있으면 생산자가 큐에 데이터를 넣을 때까지 대기를하고, 생산자가 데이터를 넣어서 소비자를 깨우면, 소비자는 할 일을 한다. 그리고 그때까지 생산자 스레드는 대기하다 소비자가 볼 일을 다 보면 다시 데이터를 넣고 이 행위를 반복하는 것이다.

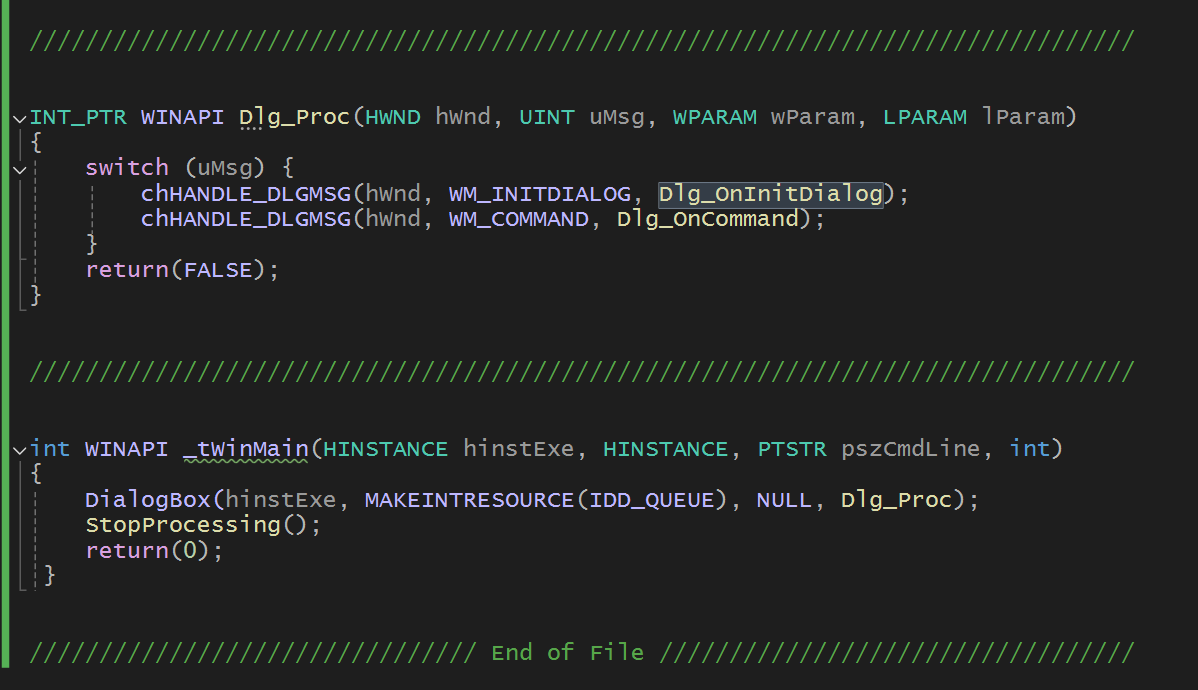


위는 전역변수들이다.

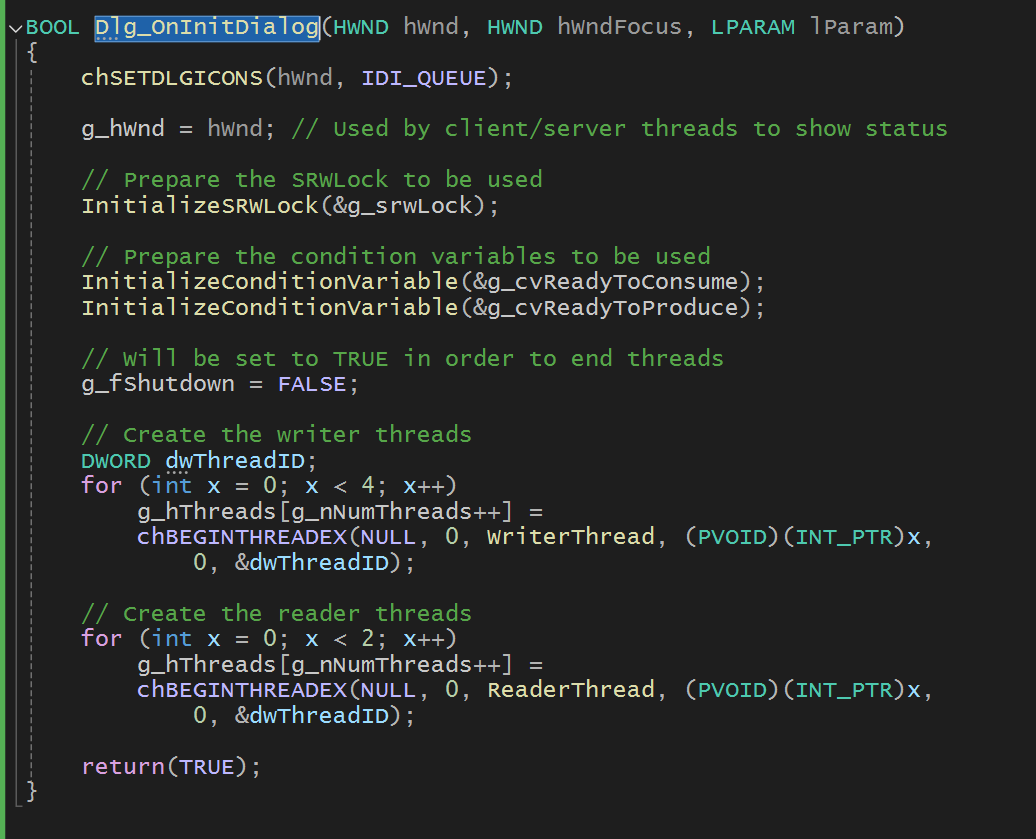
g\_cvReadyToConsume 조건변수는 라이터 스레드가 큐에 데이터를 넣으면, 해당 조건변수를

시그널 상태로 만들어서 서버를 깨우는데 사용하는 변수이다.

g\_cvReadyToProduce 조건변수도 서버스레드가 큐에 있느 데이터를 꺼내어서 처리하면, 해당 조건변수를 시그널 상태로 만들어서 라이터 스레드를 깨우는 것이다.



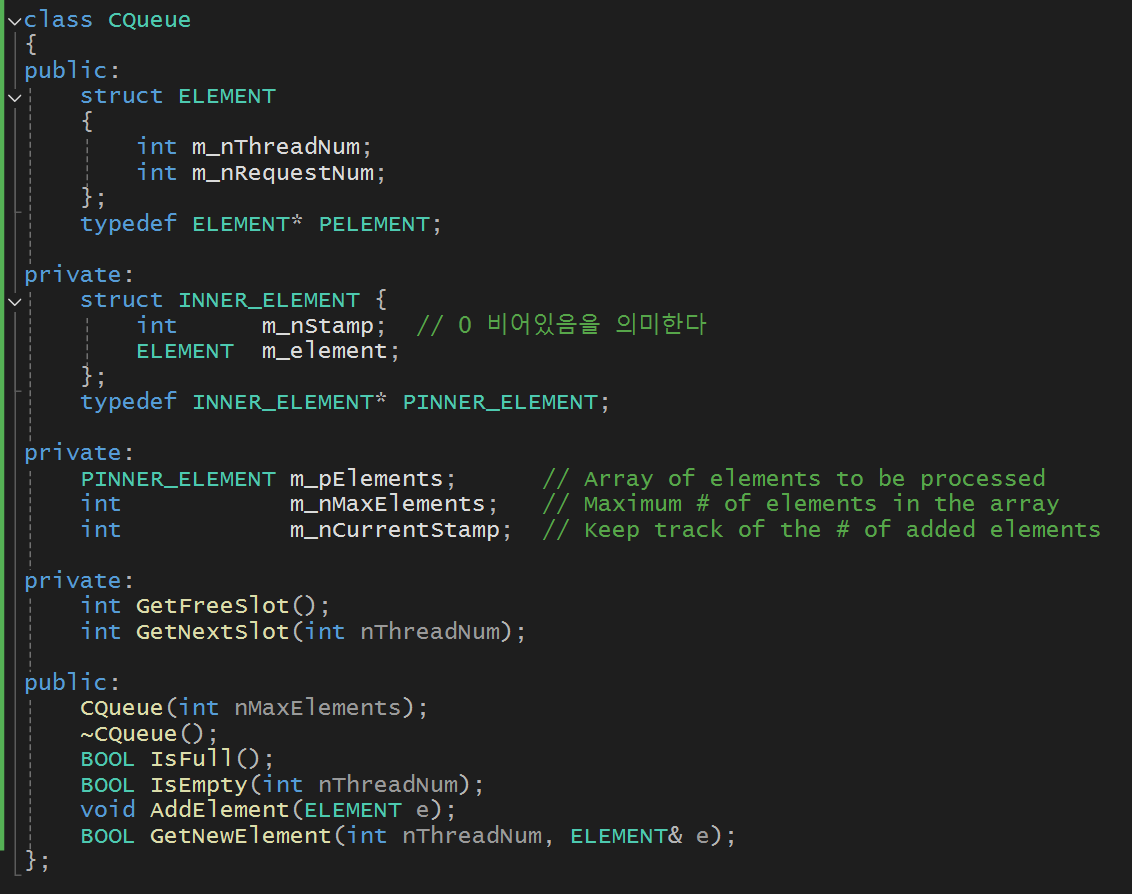
일단 주 스레드가 main함수를 시작하면 대화상자를 만들게 된다.

그리고 Dlg\_OnInitDialog 함수를 호출하여 초기화 작업을 한다. 코드는 아래와 같다

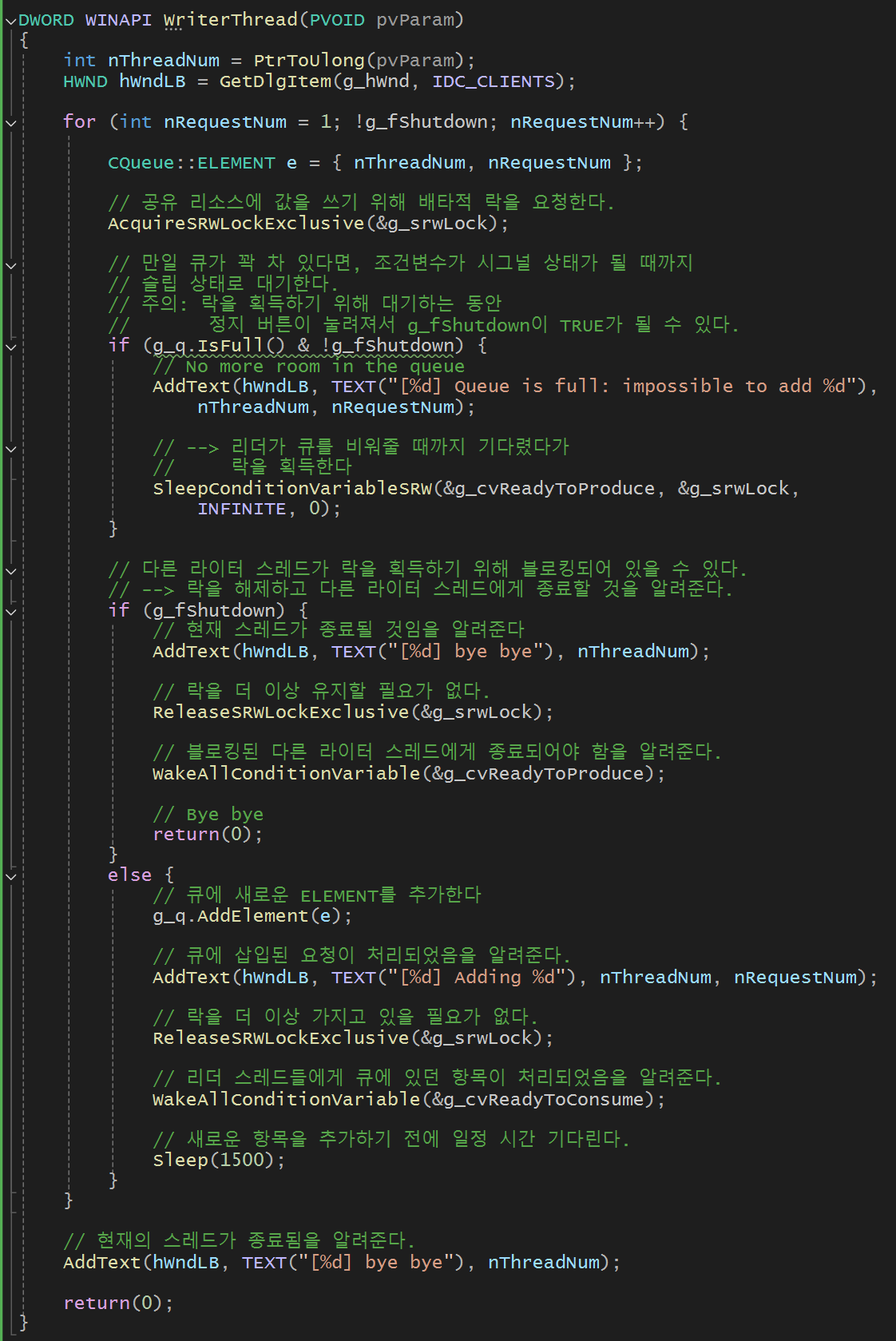
SRWLock과 조건변수들을 초기화한다. 그리고 WrierThread함수를 메인으로 돌리는 스레드 4개,

ReaderThread함수를 메인으로 돌리는 스레드 2개를 생성한다.

**(참고로 큐는 라이브러리에서 제공하는 큐가 아닌 사용자 정의 class 이다)**



**WriterThread 함수**



지금 스레드 4개가 WriterThread에 배치되어 있는 상태이다.

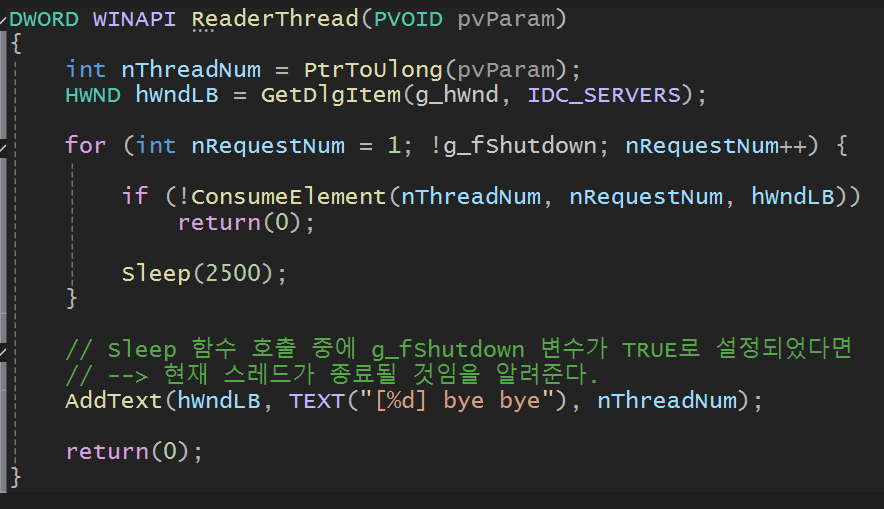
**(WriterThread 함수와 ReaderThread 함수가 동시에 호출되지만, 프로그램을 실행하면 락은 항상 Writer쪽에서 먼저 잡았다.)**

다시 본론으로 돌아와서 위 코드를 살펴보면, 4개의 스레드가 동시에 AcquireSRWLockExclusive 함수를 호출하여 락 획득에 시도한다. 하지만 큐에 데이터를 넣을 때는 한 스레드만 접근해야하므로 배타적으로 접근해야한다. 그래서 4개의 스레드 중 하나의 스레드만 먼저 락을 획득하고, 큐에 데이터를 넣을 것이다.

데이터를 넣기 전에, 큐가 꽉차있는지 확인하고, 차 있다면 SleepConditionVariableSRW를 호출하여 다시 대기모드로 들어간다.

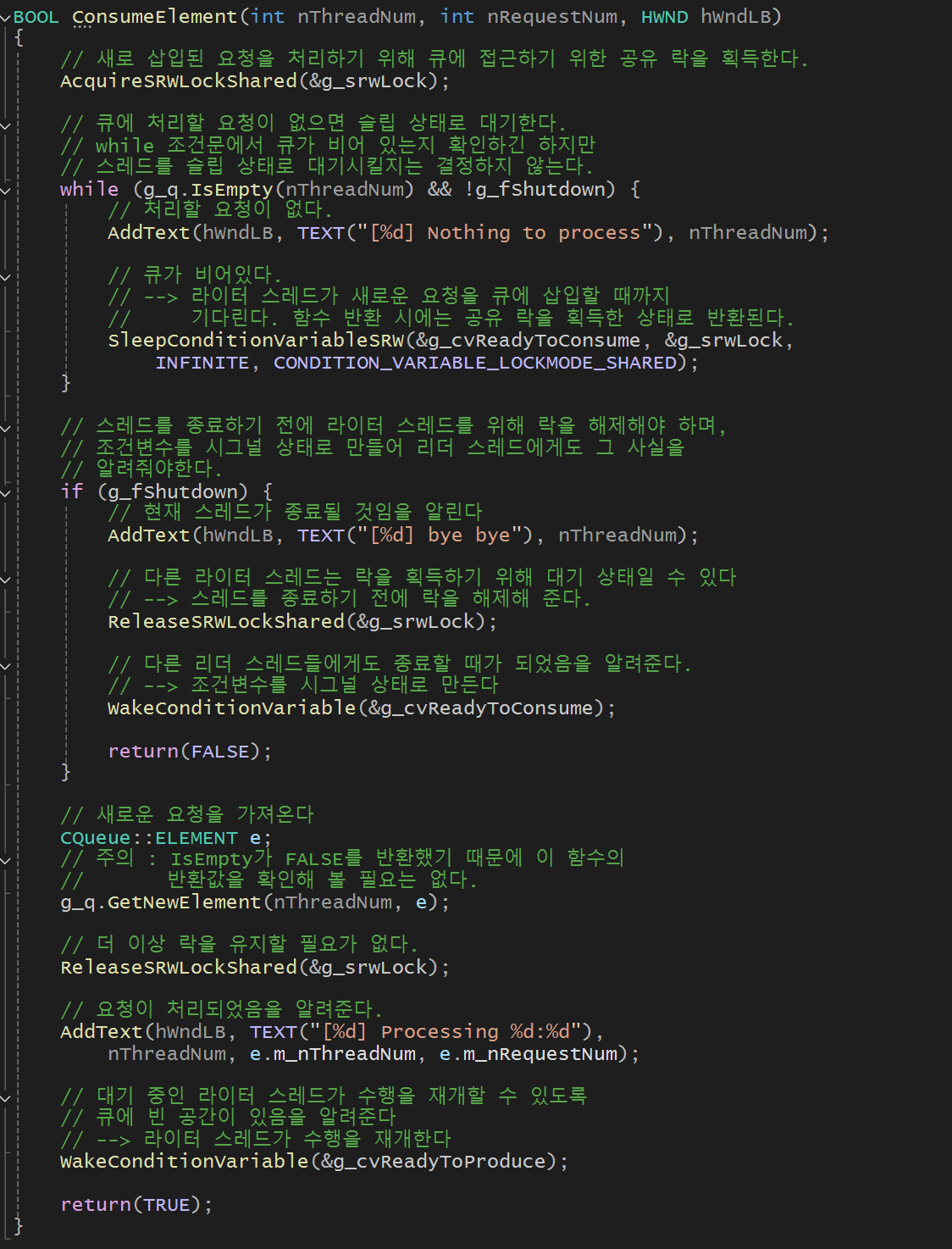
그렇다면 Writer 쪽에서 성공적으로 큐에 데이터를 넣었다면, 락을 반환하고 WakeAllConditionVariable를 호출하여 서버 스레드를 깨울 것이다.

**ReaderThread 함수**



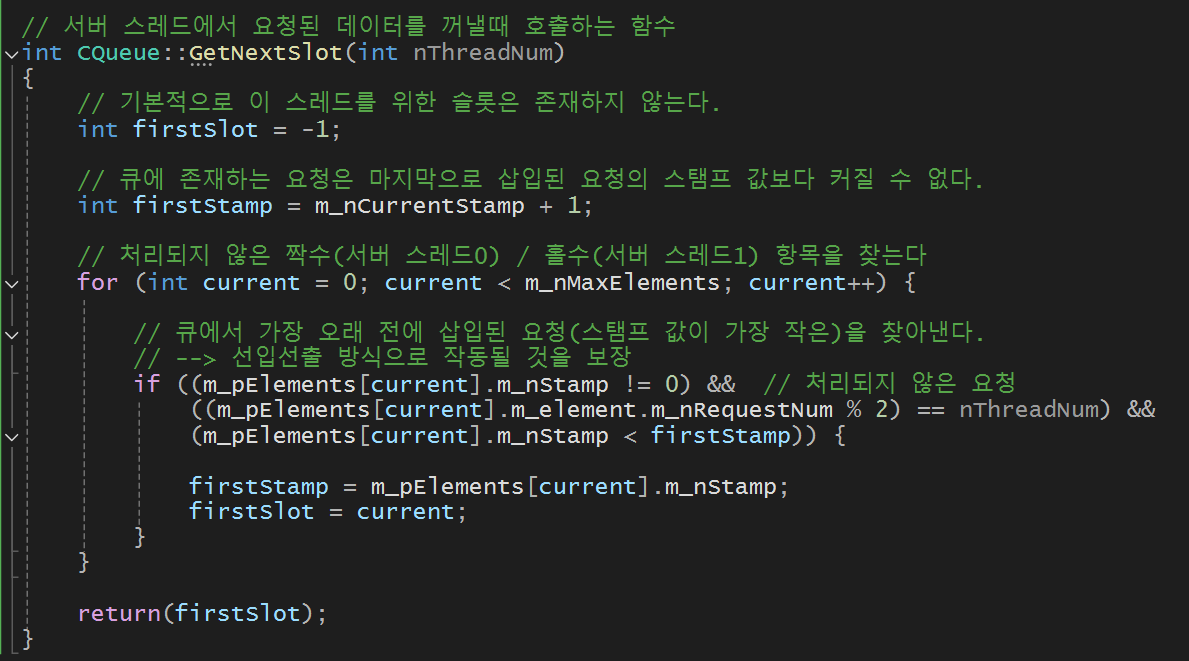
ReaderThread 함수는 2개의 스레드가 배치되어 있다.

ReaderThread에서 ConsumeElement 함수를 호출하여 큐에 있는 내용을 처리하고 있고, 그 아래사진이 ConsumeElement 함수에 대한 코드이다.



ReaderThread 부분도 위 WriterThread와 하는 일은 비슷하다. 단지 Reader는 큐에 있는 내용을 꺼내서 처리하는 것이다. 그리고 성공적으로 데이터를 꺼내어서 처리하면 다시 Writer를 깨워서 큐에 새로운 데이터를 넣을 것을 요청하는 것이다.

여기서 한가지 다른 점이라면, ConsumeElement 함수에서 AcquireSRWLockShared 함수를 호출하여 락을 얻고 있는데, Reader는 Writer 와 다르게 2개의 스레드가 동시에 락을 얻어서 큐에 있는 내용을 꺼내어서 처리할 수 있다. 그렇다면 큐에 있는 동일한 데이터를 2개의 스레드가 동시에 접근하는 위험이 있지만, 이는 큐에서 그러지 않도록 처리하고 있다.



이것이 그 해당 코드이다. 앞서 말했듯이, 이 CQeue는 사용자 정의 클래스이기 때문에,

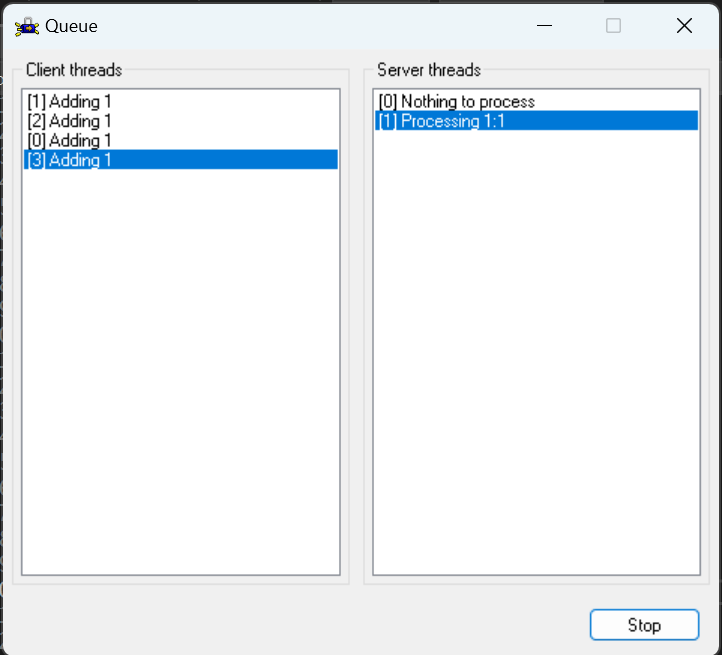
GetNextSlot 코드에서 2개의 스레드가 동시에 똑 같은 데이터를 치리하는 것을 방지하고 있다.

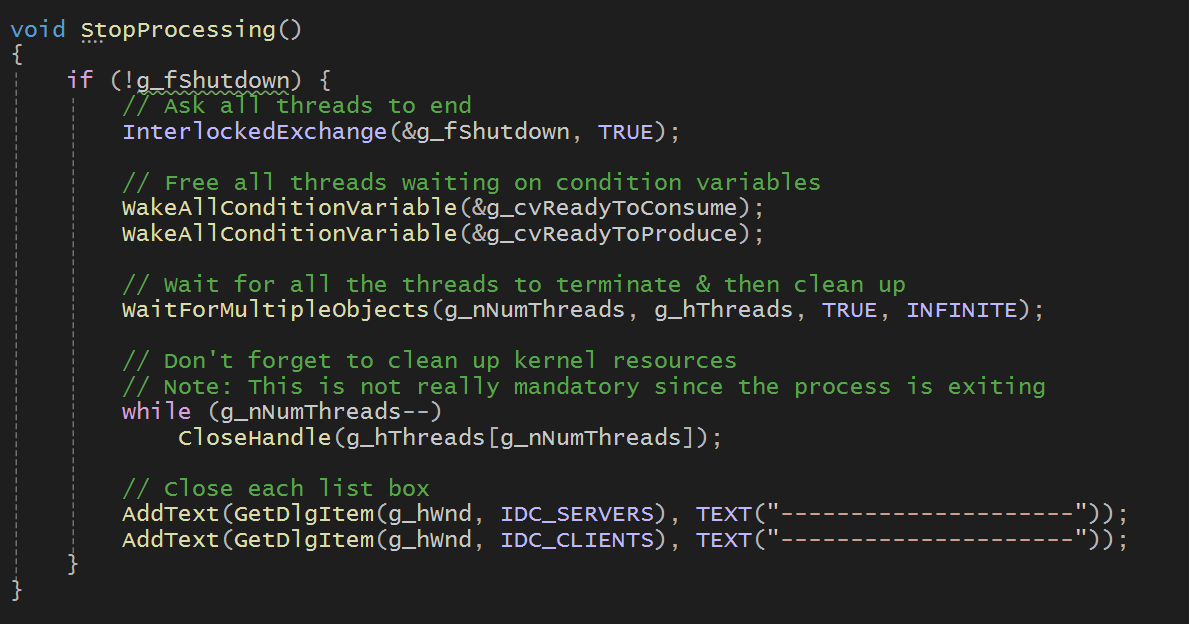
간단하게 정리하자면, 2개의 스레드 중 하나는 짝수 데이터를 처리하고 또 다른 하나는 홀수 데이터를 처리하도록 설계되어 있다. 이렇게 락을 또 걸지않고 유연하게 데이터를 꺼내어서 처리하는 방법에 대해서 배우는 시간이 되었다.

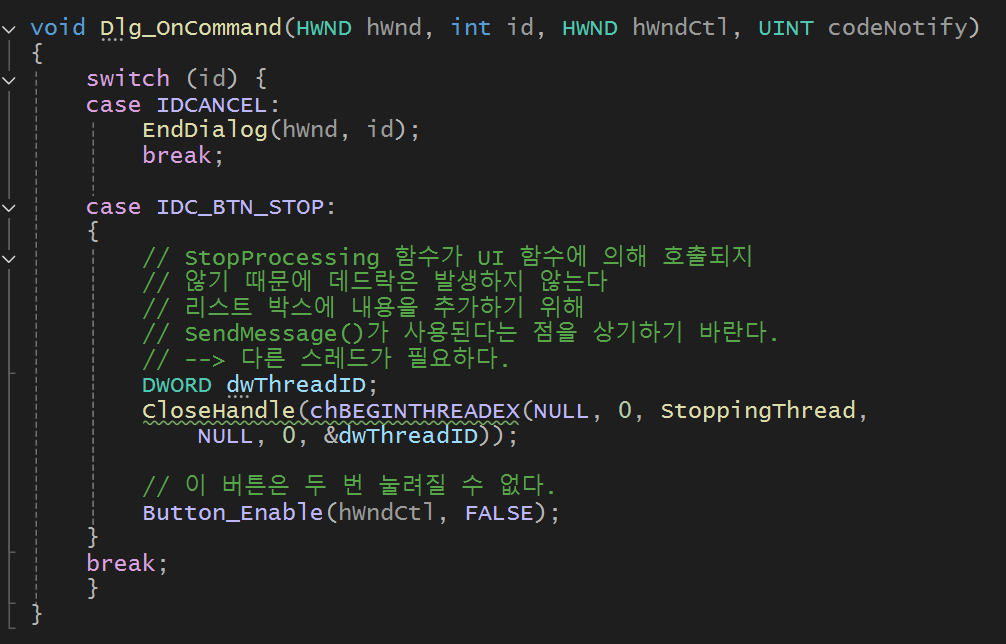
**이번 실습코드에서 말해주는 주의할 점!**

윈도우프로그램에서 대화상자를 생성하여 사용자의 요청을 처리하는 스레드가 있고 큐에 데이터를 넣어서 요청하는 **“라이터 스레드”** 4개와 큐에서 데이터를 꺼내어 요청을 처리하는 2개의 **“리더 스레드”**들, 이렇게 총 7개의 스레드들이 있다.

그리고 사용자가 “Stop” 버튼을 눌러서 대화박스를 종료하면 StopProcessing 함수를 호출한다.





근데 사용자의 인터페이스를 처리하는 스레드가 StopProcessing 함수에 진입하여 위 그림의 코드처럼 리더스레드와 라이터스레드의 종료를 기다리기 위해 WaitFroMultipleObjects를 호출하면 블로킹 상태가 될 것이다. 하지만 리더스레드와 라이터스레가 종료되면서 대화상자에 새로운 메시지를 출력하기 위해 사용자 인터페이스를 담당하는 스레드에게 요청할 것이다. 하지만 불행하게도 사용자 인터페이스 처리를 담당하는 스레드는 블로킹 상태이기 때문에 데드락 상황이 발생하는 것이다.

그래서 StopingThread 함수도 새로운 스레드를 만들어서 처리하도록 되어있다.

**요약)**

공유 리소스를 동기적으로 접근하는 것과 같이 내부적으로 블로킹을 유발할 수 있는 스레드들이 사용자 인터페이스와 관련된 메시지를 동시에 처리해야 하는 경우에는 항상 유사한 형태의 데드락에 노출될 수 있음을 잊지 말아야 한다.

**대기 목록 순회 API를 이용한 데드락 감지**

멀티 스레드로 동작하는 애플리케이션은 데드락이나 무한 대기와 같은 락과 관련된 문제점이 발생하고, 이를 찾는 것은 어려운 작업 중 하나이다.

윈도우 비스타에 추가된 “**대기 목록 순회 Wait Chain Traversal (WCT)**”와 관련된 API를 이용하면 하나의 프로세스나 다수의 프로세스들 사이에서 발생한 락을 나열하고 데드락을 발견하는 것이 가능하다.

|  |  |
| --- | --- |
| 락의 원인 | 설명 |
| 크리티컬 섹션 | 윈도우는 어떤 스레드가 크리티컬 섹션을 소유하는지 추적한다. |
| 뮤텍스 | 윈도우는 어떤 스레드가 뮤텍스를 소유하고 있는지 추적한다. 버려진 뮤텍스도 추적한다. |
| 프로세스와 스레드 | 윈도우는 어떤 스레드가 프로세스나 스레드의 종료를 대기하고 있는지 추적한다. |

* **SRWLock, 이벤트, 세마포어, 대기 타이머 드으이 다양한 커널 오브젝트는 WCT에 의해 추적되지 않는다. 그 이유는 이 오브젝트들은 어떤 스레드든지 이러한 커널 오브젝트를 시그널 상태로 언제든지 변경할 수 있기 때문이다.**

--------------------------------------------------------------------------------------------------------------

진행 상황 유튜브 링크:

|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** |  | | |
| **해결방안** |  | | |
| **다음주차** | 9주차 | **다음기간** | 8. 19 ~ 8. 25 |
| **다음주 할일** | 9주차에 계획되어있는 내용들 | | |
| **지도 교수**  **Comment** |  | | |