**병렬 프로그래밍이란?**

**병렬 프로그래밍이란 서로 독립된 여러 개의 작업을 여러 개의 프로세스를 이용하여 동시에 처리하게 만드는 프로그래밍 기법**을 말합니다. 하드웨어가 발전하면서 CPU 코어 개수는 점점 늘어나고 여러 작업을 동시에 처리할 수 있는 능력이 대폭 향상되었죠. 하지만 프로그래밍 레벨에서 이런 스펙을 최대한 이용하기란 쉽지 않은 일이었습니다. 여러 개의 스레드를 직접 만들어서 관리하고 동기화와 리소스관리까지 한다는 것은 개발자에게는 큰 부담이고 여간 귀찮은 일이 아닐 수 없습니다. 이 강좌에서는 이런 병렬 프로그래밍의 개념에 대하여 정확히 이해하고 최소한의 코드를 이용하여 효율인 결과물을 얻을 수 있는 여러 가지 방법에 대하여 다뤄보려고 합니다. 강좌가 다소 길어지더라도 차근차근 따라와 주시기 바랍니다. ^^

**Parallel Patterns Library**

자, 이제 병렬 프로그래밍을 위한 첫 번째 대주제인 Parallel Patterns Library 에 대해 알아보겠습니다. 보통 줄여서 PPL이라고 부르니 저도 이후부터는 저도 PPL이란 단어로 대신 하겠습니다. **PPL은 이름 그대로 병렬 처리를 쉽게 구현할 수 있도록 도와주는 라이브러리** 입니다. 이는 C++ 표준에 포함되어 언어 수준에서 제공하는 것은 아니며, Microsoft에서 제공하는 라이브러리로서 Visual C++ 환경에서 사용할 수 있습니다. 단, **Visual Studio 2010 이상 버전에서만 가능하니** 이전 버전 사용자들은 업그레이드를 하셔야 합니다.

PPL은 여러 개의 템플릿 클래스와 함수들로 구성되어 있으며 람다 함수 등 최신 표준인 C++11을 일부 활용하여 만들어져 있습니다. PPL은 STL과 비슷한 인터페이스 제공하며 기존 컨테이너, 알고리즘과도 호환이 가능 하기 때문에 병렬프로그래밍에 익숙하지 않은 사용자들도 어렵지 않게 접근할 수 있고 기존에 STL을 이용하여 작성된 코드를 PPL을 이용한 병렬 처리 코드로 쉽게 변환할 수 있다는 장점을 가지고 있습니다. PPL의 기능을 굳이 나누자면 다음과 같이 4가지 정도로 분류될 수 있으며, 이후 강좌에서는 차례대로 모든 기능에 대하여 설명할 예정입니다.

**1. Task Parallelism(태스크 기반의 병렬 처리)**

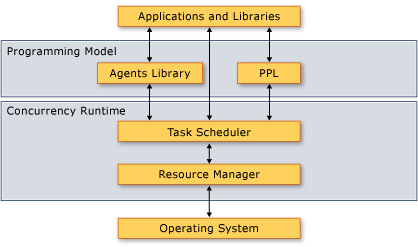
**2. Parallel Algorithm(병렬 알고리즘)**

**3. Parallel Containers and Objects(병렬 컨테이너와 오브젝트)**

**4. Cancellation in PPL(병렬 작업의 취소)**

**Concurrency Runtime**

**Concurrency Runtime은 병행 프로그래밍을 위한 프레임워크로 PPL을 포함하는 개념입니다**. PPL은 Concurrency Runtime이 제공하는 4가지 구성 요소 중 하나로서 상위 레이어에 위치하여 사용자가 응용프로그램 또는 라이브러리를 개발하는데 직접적으로 사용하게 됩니다. 하위 레이어에는 Task Scheduler와 Resource Manager가 위치해 있으며 이곳에서는 Work-stealing 알고리즘, Cooperative Scheduling, Cooperative Blocking 등을 통해 효율적으로 작업을 분배하고 시스템 자원을 관리하는 역할을 해주고 있습니다. 하지만 사용자와 하위 레이어 사이에 PPL이 위치해 있기 때문에 사용자는 굳이 하위 레이어의 존재를 알지 못하더라도 PPL을 통해 최적화된 병렬처리가 가능하게 됩니다. 여기서는 그냥 Cuncurrency Runtime에서 PPL이 이런 위치에 있다 정도로만 이해하시고 넘어가면 됩니다.



< <http://msdn.microsoft.com/en-us/library/ee207192.aspx>>

**들어가기 전에...**

 지금까지 병렬 프로그래밍과 PPL, Concurrency Runtime의 개념에 대하여 간단하게나마 알아보았습니다. 이번 장은 워밍업 정도로 생각하시면 되구요, 다음 장부터는 예제 코드를 보면서 PPL에 포함된 기능에 대하여 알아보고 이것을 어떻게 활용할 수 있는지에 대하여 설명하도록 하겠습니다. 그리고 이 강좌는 STL 컨테이너와 알고리즘, 그리고 람다 함수(Lambda Expression)의 전반적인 이해가 있다는 가정 하에 작성될 것이므로 혹시 이러한 항목들에 대한 이해가 부족하신 분들은 이 항목들의 개념을 먼저 숙지하시고 이 강좌를 보시기를 권장합니다. 그럼 다음 장에서 뵙죠.