CPU 사용률을 체크하는 클래스

```
#ifndef __CPU_USAGE_H__
#define __CPU_USAGE_H__

//////////////////////////////////////////////////////////////////////////
// CCpuUsage CPUTime();  // CPUTime(hProcess)
//
// while ( 1 )
// {
//      CPUTIme.UpdateCpuTime();
//      wprintf(L"Processor:%f / Process:%f ₩n", CPUTime.ProcessorTotal(), CPUTime.ProcessTotal());
//      wprintf(L"ProcessorKernel:%f / ProcessKernel:%f ₩n", CPUTime.ProcessorKernel(), CPUTime.ProcessKernel());
//      wprintf(L"ProcessorUser:%f / ProcessUser:%f ₩n",  CPUTime.ProcessorUser(), CPUTime.ProcessUser());
//      Sleep(1000);
// }
//////////////////////////////////////////////////////////////////////////
class CCpuUsage
{
public:
        //------------------------------------------------------------------------
        // 생성자, 확인대상 프로세스 핸들. 미입력시 자기 자신.
        //------------------------------------------------------------------------
        CCpuUsage(HANDLE hProcess = INVALID_HANDLE_VALUE);

        void    UpdateCpuTime(void);

        float   ProcessorTotal(void)     { return _fProcessorTotal; }
        float   ProcessorUser(void)      { return _fProcessorUser; }
        float   ProcessorKernel(void)    { return _fProcessorKernel; }

        float   ProcessTotal(void)       { return _fProcessTotal; }
        float   ProcessUser(void)        { return _fProcessUser; }
        float   ProcessKernel(void)      { return _fProcessKernel; }

private:

        HANDLE  _hProcess;
        int     _iNumberOfProcessors;

        float    _fProcessorTotal;
        float    _fProcessorUser;
        float    _fProcessorKernel;

        float   _fProcessTotal;
        float   _fProcessUser;
        float   _fProcessKernel;

        ULARGE_INTEGER   _ftProcessor_LastKernel;
        ULARGE_INTEGER   _ftProcessor_LastUser;
        ULARGE_INTEGER   _ftProcessor_LastIdle;

        ULARGE_INTEGER   _ftProcess_LastKernel;
        ULARGE_INTEGER   _ftProcess_LastUser;
        ULARGE_INTEGER   _ftProcess_LastTime;
};
#endif
```

```
#include <windows.h>
#include "CpuUsage.h"

//------------------------------------------------------------------------
// 생성자, 확인대상 프로세스 핸들. 미입력시 자기 자신.
//------------------------------------------------------------------------
CCpuUsage::CCpuUsage(HANDLE hProcess)
{
	//--------------------------------------------------------------------
	// 프로세스 핸들 입력이 없다면 자기 자신을 대상으로...
	//--------------------------------------------------------------------
	if ( hProcess == INVALID_HANDLE_VALUE )
	{
		_hProcess = GetCurrentProcess();
	}

	//--------------------------------------------------------------------
	// 프로세서 개수를 확인한다.
	//
	// 프로세스 (exe) 실행률 계산시 cpu 개수로 나누기를 하여 실제 사용률을 구함.
	//--------------------------------------------------------------------
	SYSTEM_INFO SystemInfo;

	GetSystemInfo(&SystemInfo);
	_iNumberOfProcessors = SystemInfo.dwNumberOfProcessors;



	_fProcessorTotal = 0;
	_fProcessorUser = 0;
	_fProcessorKernel = 0;

	_fProcessTotal = 0;
	_fProcessUser = 0;
	_fProcessKernel = 0;

	_ftProcessor_LastKernel.QuadPart = 0;
	_ftProcessor_LastUser.QuadPart = 0;
	_ftProcessor_LastIdle.QuadPart = 0;

	_ftProcess_LastUser.QuadPart = 0;
	_ftProcess_LastKernel.QuadPart = 0;
	_ftProcess_LastTime.QuadPart = 0;

	UpdateCpuTime();
}
```

```
////////////////////////////////////////////////////////////////////
// CPU 사용률을 갱신한다. 500ms ~ 1000ms 단위의 호출이 적절한듯.
//
//
////////////////////////////////////////////////////////////////////
void CCpuUsage::UpdateCpuTime()
{
	//------------------------------------------------------
	// 프로세서 사용률을 갱신한다.
	//
	// 본래의 사용 구조체는 FILETIME 이지만, ULARGE_INTEGER 와 구조가 같으므로 이를 사용함.
	// FILETIME 구조체는 100 나노세컨드 단위의 시간 단위를 표현하는 구조체임.
	//------------------------------------------------------
	ULARGE_INTEGER Idle;
	ULARGE_INTEGER Kernel;
	ULARGE_INTEGER User;

	//------------------------------------------------------
	// 시스템 사용 시간을 구한다.
	//
	// 아이들 타임 / 커널 사용 타임 (아이들포함) / 유저 사용 타임
	//------------------------------------------------------
	if ( GetSystemTimes((PFILETIME)&Idle, (PFILETIME)&Kernel, (PFILETIME)&User ) == false )
	{
		return;
	}

	// 커널 타임에는 아이들 타임이 포함됨.
	ULONGLONG KernelDiff     = Kernel.QuadPart - _ftProcessor_LastKernel.QuadPart;
	ULONGLONG UserDiff       = User.QuadPart   - _ftProcessor_LastUser.QuadPart;
	ULONGLONG IdleDiff       = Idle.QuadPart   - _ftProcessor_LastIdle.QuadPart;

	ULONGLONG Total          = KernelDiff + UserDiff;
	ULONGLONG TimeDiff;


	if ( Total == 0 )
	{
		_fProcessorUser = 0.0f;
		_fProcessorKernel = 0.0f;
		_fProcessorTotal = 0.0f;
	}
	else
	{
		// 커널 타임에 아이들 타임이 있으므로 빼서 계산.
		_fProcessorTotal  = (float)((double)(Total - IdleDiff) / Total * 100.0f);
		_fProcessorUser   = (float)((double)UserDiff / Total * 100.0f);
		_fProcessorKernel = (float)((double)(KernelDiff - IdleDiff) / Total * 100.0f);
	}

	_ftProcessor_LastKernel = Kernel;
	_ftProcessor_LastUser = User;
	_ftProcessor_LastIdle = Idle;
```

```
        //--------------------------------------------------------
        // 지정된 프로세스 사용률을 갱신한다.
        //--------------------------------------------------------
        ULARGE_INTEGER None;
        ULARGE_INTEGER NowTime;

        //--------------------------------------------------------
        // 현재의 100 나노세컨드 단위 시간을 구한다. UTC 시간.
        //
        // 프로세스 사용률 판단의 공식
        //
        // a = 샘플간격의 시스템 시간을 구함. (그냥 실제로 지나간 시간)
        // b = 프로세스의 CPU 사용 시간을 구함.
        //
        // a : 100 = b : 사용률   공식으로 사용률을 구함.
        //--------------------------------------------------------

        //--------------------------------------------------------
        // 얼마의 시간이 지났는지 100 나노세컨드 시간을 구함,
        //--------------------------------------------------------
        GetSystemTimeAsFileTime((LPFILETIME)&NowTime);

        //--------------------------------------------------------
        // 해당 프로세스가 사용한 시간을 구함.
        //
        // 두번째, 세번째는 실행,종료 시간으로 미사용.
        //--------------------------------------------------------
        GetProcessTimes(_hProcess, (LPFILETIME)&None, (LPFILETIME)&None, (LPFILETIME)&Kernel, (LPFILETIME)&User);

        //--------------------------------------------------------
        // 이전에 저장된 프로세스 시간과의 차를 구해서 실제로 얼마의 시간이 지났는지 확인.
        //
        // 그리고 실제 지나온 시간으로 나누면 사용률이 나옴.
        //--------------------------------------------------------
        TimeDiff   = NowTime.QuadPart    - _ftProcess_LastTime.QuadPart;
        UserDiff   = User.QuadPart       - _ftProcess_LastUser.QuadPart;
        KernelDiff = Kernel.QuadPart     - _ftProcess_LastKernel.QuadPart;

        Total    = KernelDiff + UserDiff;


        _fProcessTotal   = (float)(Total / (double)_iNumberOfProcessors / (double)TimeDiff * 100.0f);
        _fProcessKernel  = (float)(KernelDiff / (double)_iNumberOfProcessors / (double)TimeDiff * 100.0f);
        _fProcessUser    = (float)(UserDiff / (double)_iNumberOfProcessors / (double)TimeDiff * 100.0f);

        _ftProcess_LastTime = NowTime;
        _ftProcess_LastKernel = Kernel;
        _ftProcess_LastUser = User;

}
```