# Thread

## 1. 함수 포인터

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <thread>

#define THRD_SIZE 10

void CallBack_ThrdFunc(void *pParam)
{
	int nNum = *((int*)pParam);

	int nCNT = 0;
	while (nCNT < nNum)
	{
		printf("[%d] %d Thrd\n", nCNT++, nNum);
	}
}

int main(void)
{
	int nThrdSize = THRD_SIZE;
	std::thread thrd[THRD_SIZE];
	for (int i = 0; i < THRD_SIZE; ++i)
	{
		thrd[i] = std::thread(CallBack_ThrdFunc, &i);
	}
```

```
        for (int i = 0; i < THRD_SIZE; ++i)
        {
                thrd[i].join();
        }

        system("pause");
        return 0;
}
```

---

## 2. 함수 객체

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <thread>

class Counter
{
public:
        Counter(char* strName, int nFir, int nSec)
        {
                m_strName = (char*)malloc(10);
                //strcpy(m_strName, strName);

                m_nFir = nFir;
                m_nSec = nSec;
        }

        void operator()() const
```

```
        {
                for (int i = m_nFir; i <= m_nSec; ++i)
                {
                        printf("[%s] %d\n", m_strName, i);
                }
        }

private:
        char *m_strName;
        int m_nFir;
        int m_nSec;
};

int main(void)
{
        char strName[10] = "Thrd1";
        std::thread t1{ Counter(strName, 1, 5) };

        t1.join();

        system("pause");
        return 0;
}
```

---

# 3. 클래스 메소드

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
```

```
#include <thread>

class Counter
{
public:
        Counter(char* strName, int nFir, int nSec)
        {
                m_strName = (char*)malloc(10);
                strcpy(m_strName, strName);

                m_nFir = nFir;
                m_nSec = nSec;
        }

        void loop() const
        {
                for (int i = m_nFir; i <= m_nSec; ++i)
                {
                        printf("[%s] %d\n", m_strName, i);
                }
        }

private:
        char *m_strName;
        int m_nFir;
        int m_nSec;
};

int main(void)
{
```

```
        char strName[10] = "Thrd1";

        Counter c1(strName, 1, 5);
        std::thread t1{ &Counter::loop, &c1};

        t1.join();

        system("pause");
        return 0;
}
```

---

# 4. 람다표현식

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <thread>

int main(void)
{
        char strName[10] = "Thrd1";

        std::thread t1([](char* strName, int nFir, int nSec)
        {
                for (int i = nFir; i <= nSec; ++i)
                {
                        printf("[%s] %d\n", strName, i);
                }
        }, strName, 1, 5);
```

```
        t1.join();

        system("pause");
        return 0;
}
```

---

# 5. 클래스 메소드 (고급)

## 1. Thread.h

```
#pragma once
#include <process.h>

struct _ST_CREATE_THREAD_DATA
{
        int(*pfEntry)(void* pContext);
        void*                           pContext;
};

static unsigned WINAPI _InternalThreadCaller(void* pContext)
{
        _ST_CREATE_THREAD_DATA* pData = (_ST_CREATE_THREAD_DAT
A*)pContext;
        int nRet = pData->pfEntry(pData->pContext);
        delete pData;
        return nRet;
}

HANDLE _CreateThread(int(*pfEntry)(void* pContext), void* pCon
text)
{
```

```
        _ST_CREATE_THREAD_DATA* pData = new _ST_CREATE_THREAD_
DATA;
        pData->pfEntry = pfEntry;
        pData->pContext = pContext;

        HANDLE hThread = (HANDLE)::_beginthreadex(NULL, 0, _In
ternalThreadCaller, pData, 0, NULL);
        if (INVALID_HANDLE_VALUE == hThread)
        {
                printf("_beginthreadex operation failure");
                delete pData;
                return NULL;
        }

        return hThread;
}

struct INTERNAL_COMMON_THREAD_DATA
{
        virtual ~INTERNAL_COMMON_THREAD_DATA(void) {}
        virtual int ThreadFunc(void) = 0;
};

template<typename T>
struct INTERNAL_TEMPLATE_THREAD_DATA : public INTERNAL_COMMON_
THREAD_DATA
{
        T* pInstance;
        int (T::*pfMemberFunc)(void* pContext);
        void* pContext;
```

```
        int ThreadFunc(void) { return (pInstance->*pfMemberFun
c)(pContext); }
};

int InternalThreadCaller(void* pContext)
{
        INTERNAL_COMMON_THREAD_DATA* pThreadData = (INTERNAL_C
OMMON_THREAD_DATA*)pContext;
        int nErrCode = pThreadData->ThreadFunc();
        delete pThreadData;
        return nErrCode;
}

template<typename T>
HANDLE CreateThread(T* pInstance, int (T::*pfEntry)(void*), vo
id* pContext)
{
        INTERNAL_TEMPLATE_THREAD_DATA<T>* pThreadData = new IN
TERNAL_TEMPLATE_THREAD_DATA<T>;
        if (NULL == pThreadData)
                return NULL;
        pThreadData->pInstance = pInstance;
        pThreadData->pfMemberFunc = pfEntry;
        pThreadData->pContext = pContext;
        return _CreateThread(InternalThreadCaller, pThreadDat
a);
}
```

## 2. Main.cpp

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
```

```
#include <windows.h>

#include "Thread.h"

class TestClass
{
public:
        TestClass() {}
        ~TestClass() {}

private:
        HANDLE m_thrd;
        bool m_bCheck;

public:
        void StartUp()
        {
                m_bCheck = true;
                m_thrd = CreateThread(this, &TestClass::Thread
_Func, NULL);
        }
        void ShutDown(DWORD dwTimeOut = 1000)
        {
                m_bCheck = false;

                DWORD dwRet = ::WaitForSingleObject(m_thrd, dw
TimeOut);
                if (WAIT_OBJECT_0 != dwRet)
                {
                        printf("ShutDown Error : %d.", dwRet);
```

```
                        ::TerminateThread(m_thrd, 0);
                }
                else
                        printf("ShutDown Success.\n");
                ::CloseHandle(m_thrd);
        }

private:
        int        Thread_Func(void* pContext)
        {
                printf("Start\n");
                while (m_bCheck)
                {
                        printf("TEST");
                        Sleep(1000);
                }
                printf("\nEnd\n");
                return 0;
        }
};

int main(void)
{
        TestClass Test;
        Test.StartUp();

        int nCNT = 0;
        while (nCNT++ < 10)
        {
                Sleep(1000);
```

```
        }
        Test.ShutDown();

        system("pause");
        return 0;
}
```