线程篇（上）

七日成蝶——Windows API 编程入门

七日做茧，一朝成蝶！

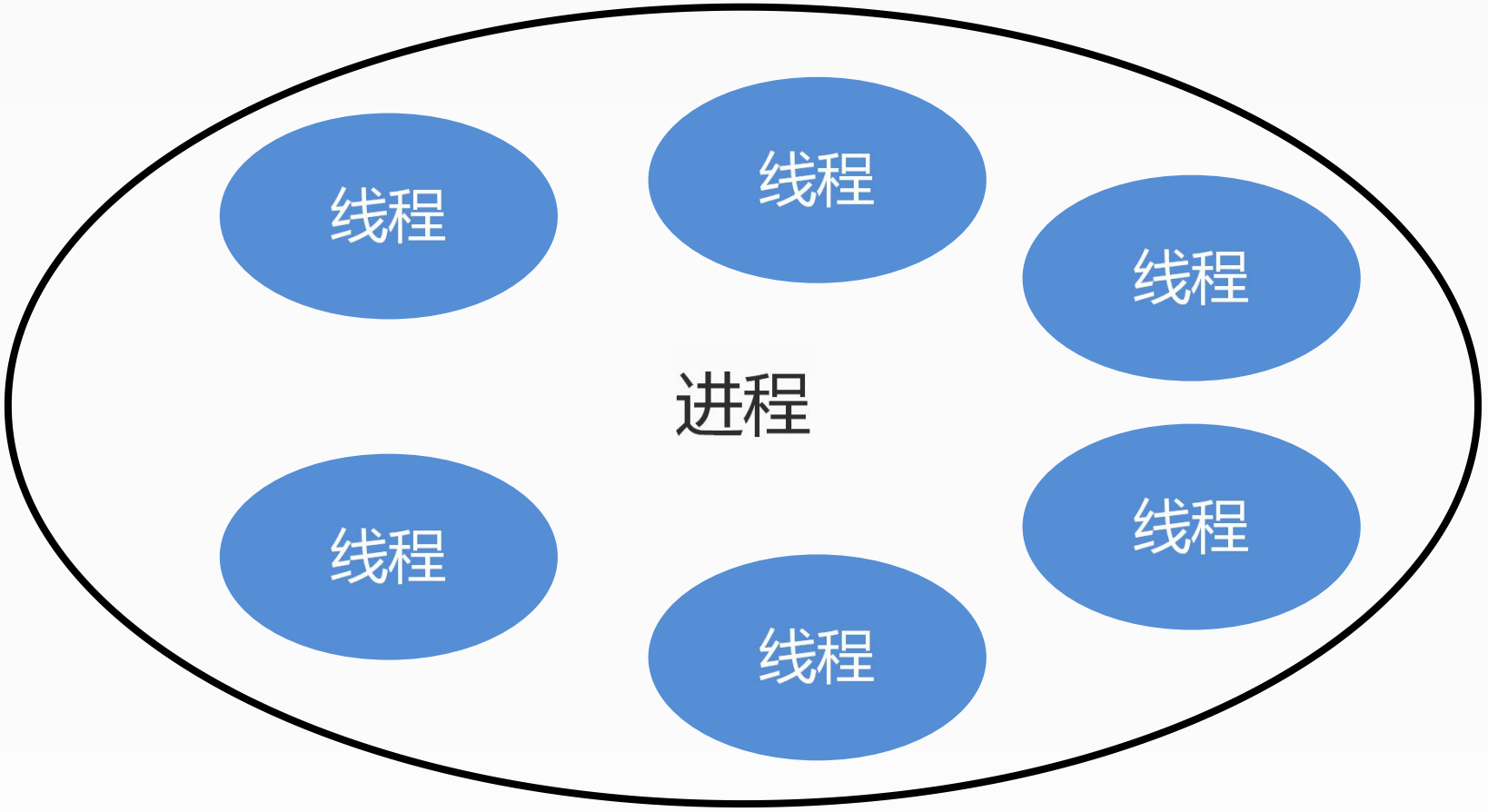
主讲：袁春旭

个人博客：http://8413723.blog.51cto.com/

课程主页：http://edu.51cto.com/lecturer/8403723.html

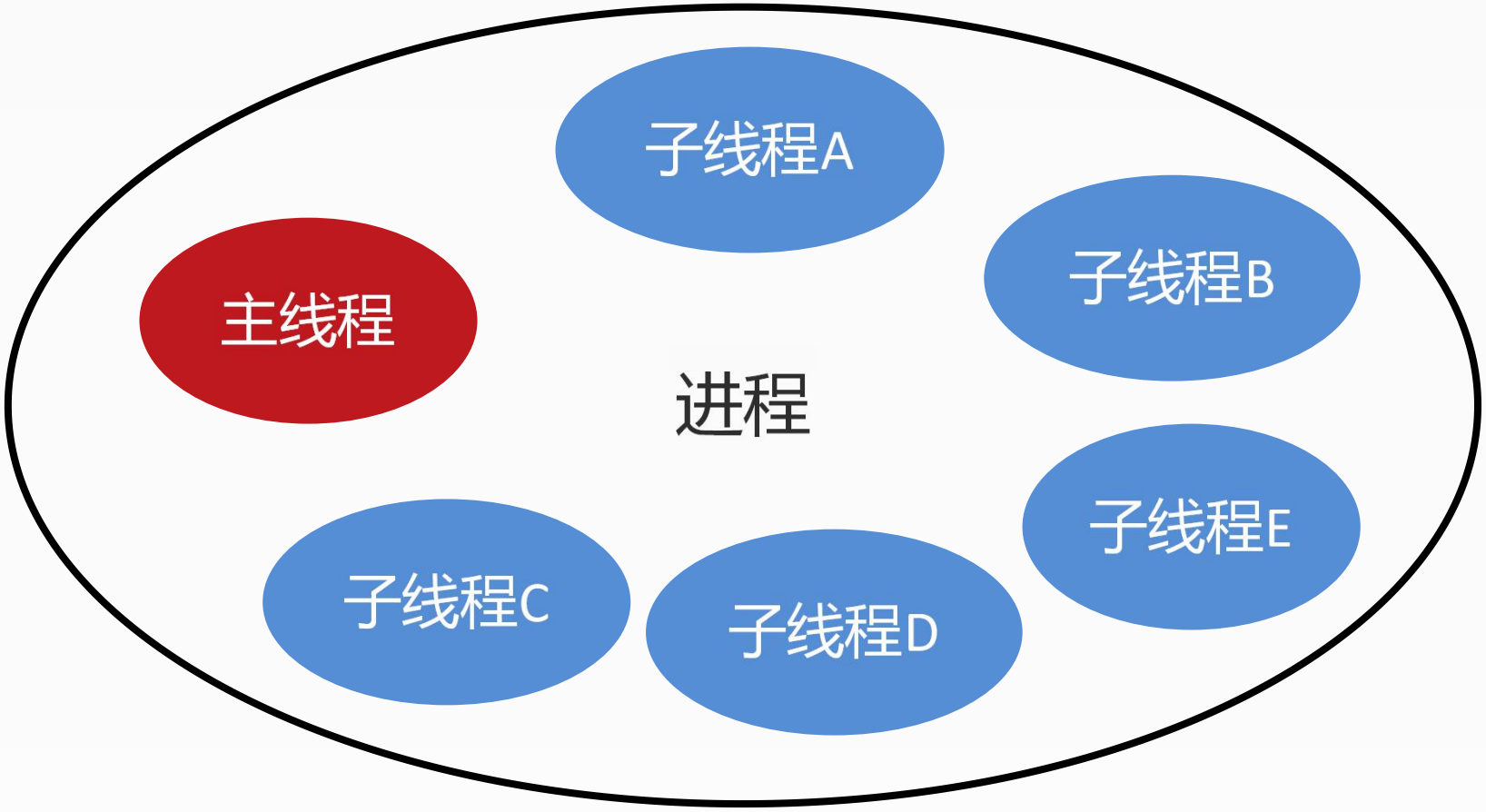
线程

线程



主线程

主线程



线程组成

线程组成

进程
- 进程内核对象
- 进程地址空间

线程
- 线程内核对象
- 线程栈

关于线程的几个问题

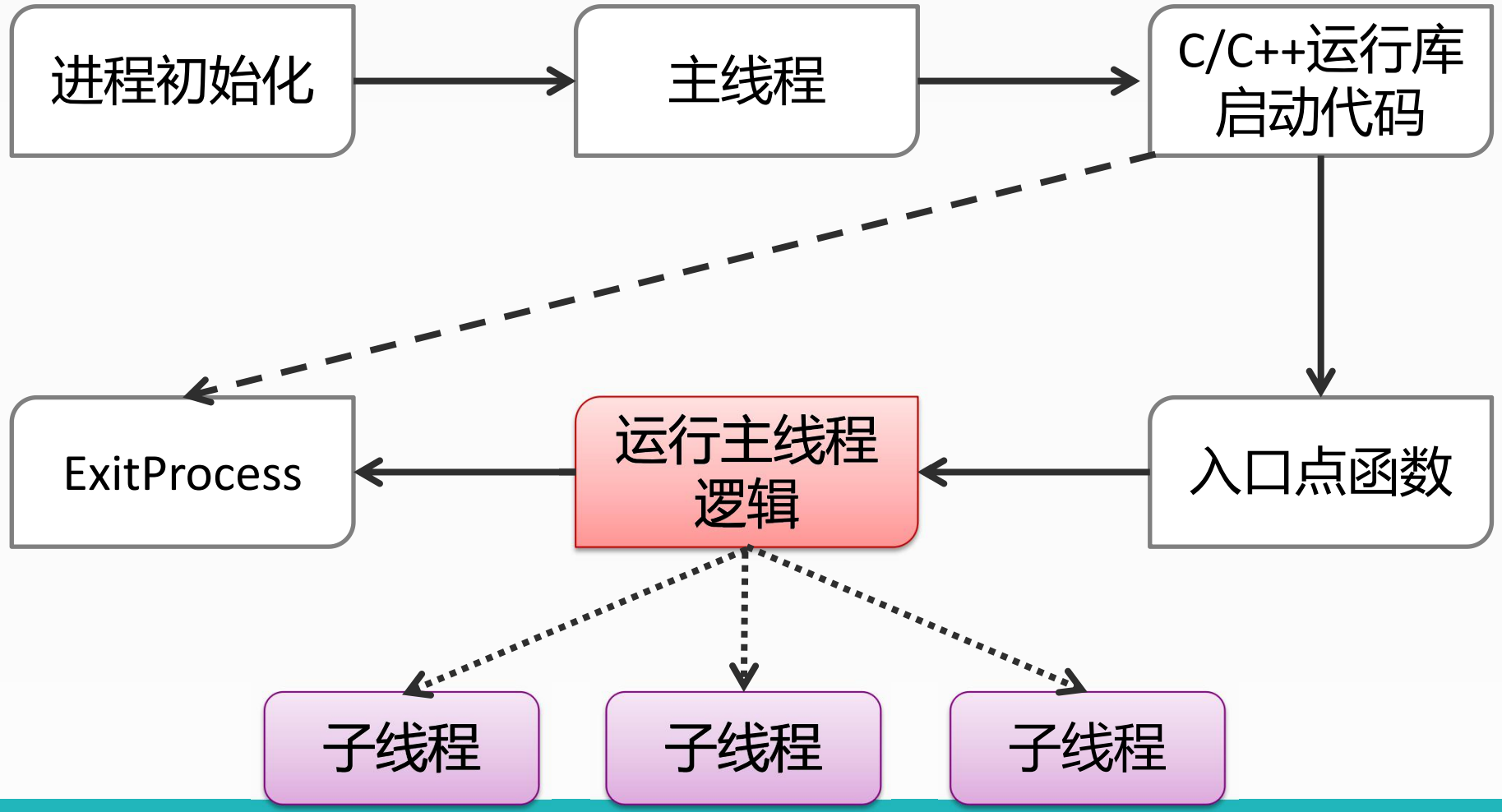
关于线程的几个问题

主线程由谁创建	操作系统
主线程何时创建	进程初始化
线程资源来源	进程资源
一个进程中只有一个线程吗	多个线程

关于线程的几个问题

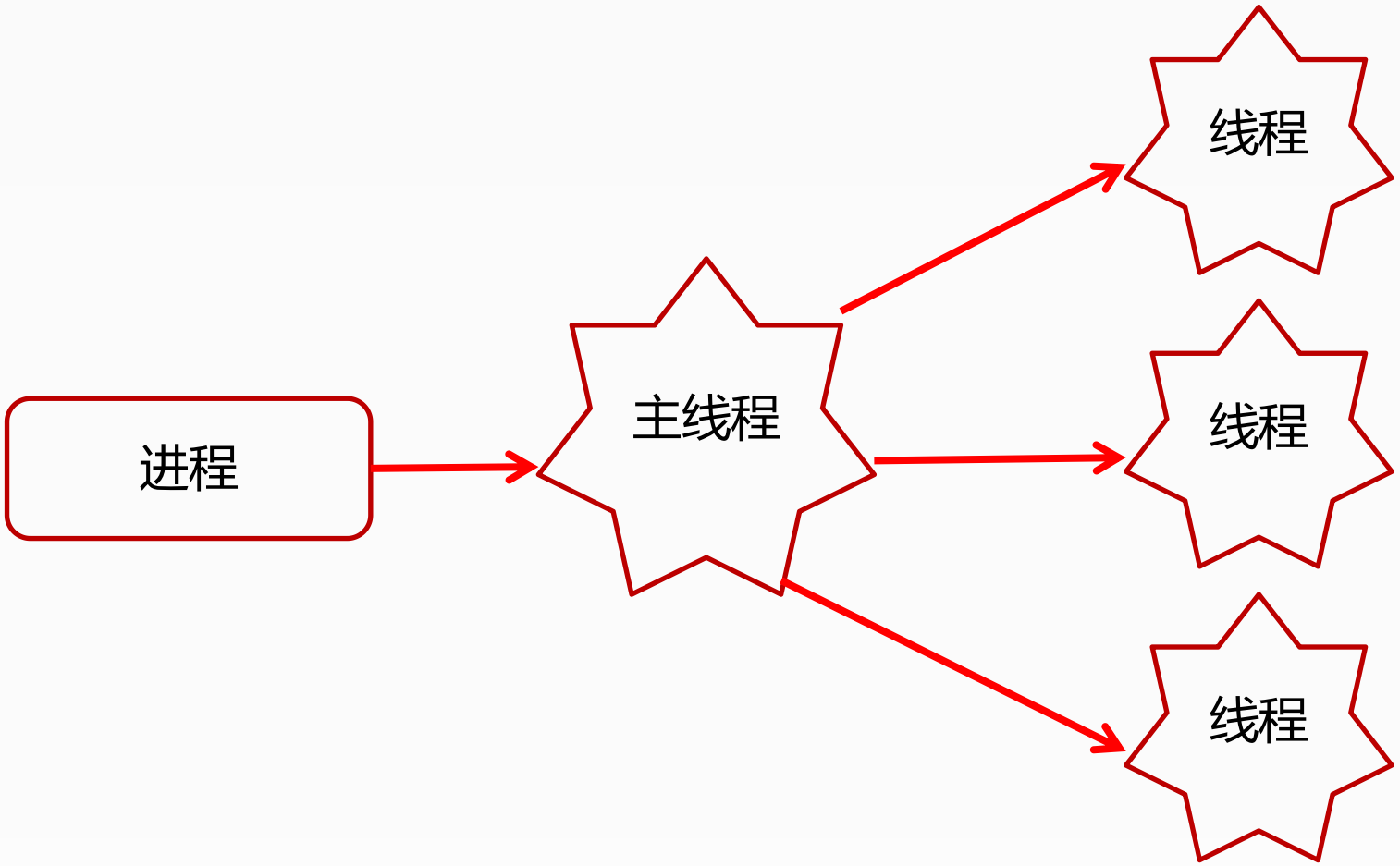
主线程生命周期	C/C++启动代码，入口点函数，ExitProcess
主线程入口点函数	main、wmain、WinMain、wWinMain
线程、进程谁更省	线程

关于线程的几个问题



创建线程

创建线程



创建线程

```
HANDLE CreateThread(
  LPSECURITY_ATTRIBUTES lpThreadAttributes,  //线程安全属性

  DWORD dwStackSize,  //线程堆栈大小

  LPTHREAD_START_ROUTINE lpStartAddress,  //线程函数地址

  LPVOID lpParameter,  //线程函数参数

  DWORD dwCreationFlags,  //指定线程是否立即启动

  DWORD* lpThreadId  //存储线程ID号
)
```

创建线程

```
SECURITY_ATTRIBUTES sa;
sa.nLength = sizeof(sa);
sa.lpSecurityDescriptor = NULL;
sa.bInheritHandle = TRUE;  //设置句柄可以继承

//句柄h可以被子进程继承
HANDLE h = CreateThread(&sa, ……);
```

创建线程

```
DWORD WINAPI ThreadPro(LPVOID lpParam);
int main(int argc, char *argv[])
{
        HANDLE hThread;
        DWORD dwThreadId;
        hThread = CreateThread(
                        NULL,           //默认安全属性
                        NULL,           //默认堆栈大小
                        ThreadPro,      //线程入口地址
                        NULL,           //传给函数的参数
                        0,              //指定线程立即运行
                        &dwThreadId     //线程ID号
                        );
```

创建线程

```
        for(int i = 0; i < 4; i++)    //循环打印4次
        {
                printf("hello\n");
        }
        CloseHandle(hThread);   //关闭线程句柄
        system("pause");
        return 0;
}
DWORD WINAPI ThreadPro(LPVOID lpParam)
{
        for(int i = 0; i < 4; i++)    //循环打印4次
        {
                printf("world\n");
        }
        return 0;
}
```

创建线程

程序的运行效果：

创建线程

```
#include <process.h>

unsigned int _beginthreadex(
  void *_Security,  //线程安全属性
  unsigned _StackSize,  //线程堆栈大小
  unsigned (*_beginthreadex_proc_type)(void*) _StartAddress,  //线程函数地址
  void *_ArgList,  //线程函数参数
  unsigned _InitFlag,  //指定线程是否立即启动
  unsigned *_ThrdAddr  //存储线程ID号
)
```

线程优先级

线程优先级

BOOL SetThreadPriority(HANDLE hThread, **int nPriority**);

线程优先级

优先级名称	优先级说明
THREAD_PRIORITY_TIME_CRITICAL	实时
THREAD_PRIORITY_HIGHEST	最高
THREAD_PRIORITY_ABOVE_NORMAL	高于正常
THREAD_PRIORITY_NORMAL	正常
THREAD_PRIORITY_BELOW_NORMAL	低于正常
THREAD_PRIORITY_LOWEST	最低
THREAD_PRIORITY_IDLE	空闲

线程优先级

- 优先级号0（最低）-31（最高）。
- 调度方式—高优先级优先调度，同优先级机会均等。

线程A	THREAD_PRIORITY_ABOVE_NORMAL
线程B	THREAD_PRIORITY_IDLE
线程C	THREAD_PRIORITY_NORMAL
线程D	THREAD_PRIORITY_NORMAL
线程E	THREAD_PRIORITY_NORMAL
线程F	THREAD_PRIORITY_ABOVE_NORMAL
线程G	THREAD_PRIORITY_LOWEST
…	

线程优先级

```
DWORD WINAPI Thread1(LPVOID lpParam){
        for(int i = 0; i < 10; i++){
                printf("hello\n");
        }
        return 0;
}
```

```
DWORD WINAPI Thread2(LPVOID lpParam){
        for(int i = 0; i < 10; i++){
                printf("world\n");
        }
        return 0;
}
```

线程优先级

思考：低优先级线程是否完全没有机会？

线程终止

线程终止

- 线程终止的四种情况：

– 线程函数自然退出。（推荐）
– 使用ExitThread函数终止线程。（自身终止）
– 使用TerminateThread函数终止线程。（A终止B）
– 使用ExitProcess结束线程所在进程。（进程退出）

线程终止

- 线程正常终止时会发生下列事件：

– 线程中所有对象用各自的析构函数销毁。
– 该线程使用的堆栈被释放。
– 系统将Exit Code设置为线程函数返回值。
– 递减内核对象中的Usage Code的值。

思考：暴力终止线程存在的问题

终止线程

```
HANDLE TerminateThread(
    HANDLE hThread,      //目标线程句柄
    DWORD dwExitCode    //退出代码
  )
```

终止线程

```
HANDLE ExitThread(

    DWORD dwExitCode    //退出代码

  )
```

终止线程

```
void _endthreadex(

    unsigned retcode    //退出码

  )
```

废弃函数

废弃函数

```
void _beginthread(
   _beginthread_proc_type _StartAddress,
   unsigned               _StackSize,
   void*                  _ArgList
)
```

```
void _endthread(void)
```

编码实战

Thank You !