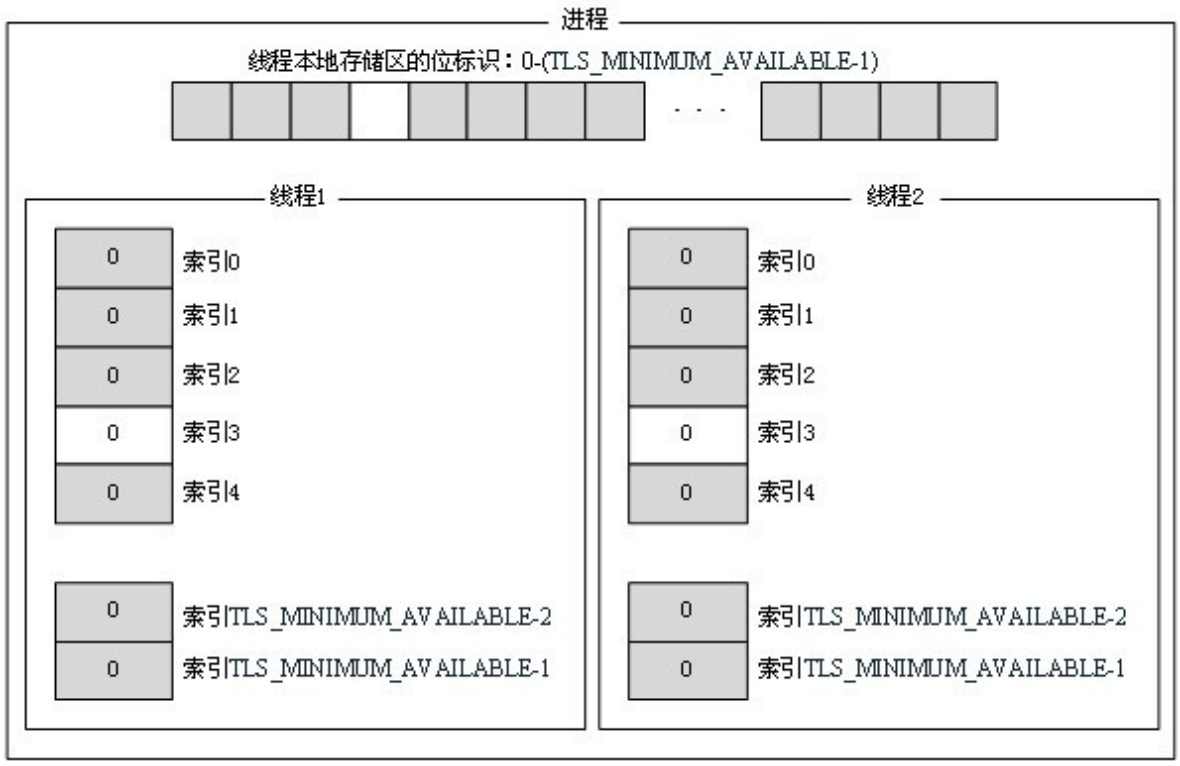
TLS

线程局部存储（英文：Thread Local Storage），各线程独立，属于线程自己的全局变量（可在线程内部跨函数）

在`TEB`的`E10`处存放了64项的数组用来实现TLS

线程是通过使用一个数组来保存与线程相关联的数据的，这个数组由`TLS_MINIMUM_AVAILABLE`个元素组成，在`winnt.h`文件中该值被定义为**64**个。也就是说当线程创建时，系统给每一个线程分配了一个数组，这个数组共有`TLS_MINIMUM_AVAILABLE`个元素，并且将这个数组的各个元素初始化为**0**，之后系统把这个数组与新创建的线程关联起来。每一个线程中都有它自己的数组，数组中的每一个元素都能保存一个**32位**的值。

此数组在`TEB`中



函数`TlsAlloc`判断数组中一个元素可用后，就把这个元素分配给调用的线程，并保留给调用线程。要为数组中的某个元素赋值可以使用函数`TlsSetValue`，要得到某个元素的值可以使用`TlsGetValue`。

相关函数

TlsAlloc

```
DWORD TlsAlloc();
```

检索可用的数组单元返回其索引

TlsFree

```
BOOL TlsFree(
  DWORD dwTlsIndex
);
```

释放此单元

TlsSetValue

```
BOOL TlsSetValue(
  DWORD  dwTlsIndex,
  LPVOID lpTlsValue
);
```

设置值

TlsGetValue

```
LPVOID TlsGetValue(
  DWORD dwTlsIndex
);
```

获取值

示例

```
#include <stdlib.h>
#include <time.h>

#define THREAD_NUM 5

struct MyArg {
    DWORD arg;
};

void ThreadWork(DWORD index)
{
    // 获取值
    MyArg *work = (MyArg *)::TlsGetValue(index);
    work->arg = index;
    for (int i = 0; i < 5; i++) {
        printf("CurrentThreadID: %d, Work: %d\n", ::GetCurrentThreadId(), work-
>arg);
        Sleep(index);
    }
}

unsigned int __stdcall ThreadProc(void *arg)
{
    MyArg *work = new MyArg;
    // 申请tls空间
    DWORD index = ::TlsAlloc();
    // 设置值
    ::TlsSetValue(index, work);
```

```
    // 执行相关业务代码
    ThreadWork(index);
    // 释放tls空间
    ::TlsFree(index);

    delete work;
    return 0;
}

int main(void)
{
    srand((unsigned int)time(NULL));
    HANDLE hThreads[THREAD_NUM] = { 0 };
    for(int i = 0; i < THREAD_NUM; i++) {
        hThreads[i] = (HANDLE)_beginthreadex(NULL, 0, ThreadProc, (void *)i,
NULL, NULL);
    }

    ::WaitForMultipleObjects(THREAD_NUM, hThreads, TRUE, INFINITE);
    return 0;
}
```

在PE中

主要负责为主线程提供TLS功能，以下描述的地址都是VA

```
#define IMAGE_DIRECTORY_ENTRY_TLS 9

typedef struct _IMAGE_TLS_DIRECTORY32 {
    DWORD   StartAddressOfRawData;  // TLS初始化数据的起始地址
    DWORD   EndAddressOfRawData;    // TLS初始化数据的结束地址，两个正好定位一个范围,范
围放初始化的值
    DWORD   AddressOfIndex;         // TLS索引的位置
    DWORD   AddressOfCallBacks;     // TLS初始化回调函数的数组指针，以0结尾，回调跟
dllmain一样，TLS函数执行在主线程前
    DWORD   SizeOfZeroFill;         // 填充0的个数
    union {
        DWORD Characteristics;      // 保留
        struct {
            DWORD Reserved0 : 20;
            DWORD Alignment : 4;
            DWORD Reserved1 : 8;
        } DUMMYSTRUCTNAME;
    } DUMMYUNIONNAME;
} IMAGE_TLS_DIRECTORY32;
```

`__declspec(thread) 类型 变量名`将变量声明为主线程的TLS，其他线程也有用其独立的备份

在为主线程提供TLS功能时，`TEB + x2c`处是指向主线程TLS下标（此下标意指为地址）的数组，初始化时，操作系统会将`TLS DATA`复制到堆上

```
eip=76fde9e2 esp=00eff4fc ebp=00eff528 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
ntdll!LdrpDoDebuggerBreak+0x2b:
76fde9e2 cc              int     3
0:000> dt _TEB
ntdll!_TEB
   +0x000 NtTib            : _NT_TIB
   +0x01c EnvironmentPointer : Ptr32 Void
   +0x020 ClientId         : _CLIENT_ID
   +0x028 ActiveRpcHandle  : Ptr32 Void
   +0x02c ThreadLocalStoragePointer : Ptr32 Void
   +0x030 ProcessEnvironmentBlock : Ptr32 _PEB
   +0x034 LastErrorValue   : Uint4B
   +0x038 CountOfOwnedCriticalSections : Uint4B
   +0x03c CsrClientThread  : Ptr32 Void
   +0x040 Win32ThreadInfo  : Ptr32 Void
   +0x044 User32Reserved   : [26] Uint4B
   +0x0ac UserReserved     : [5] Uint4B
   +0x0c0 WOW32Reserved    : Ptr32 Void
   +0x0c4 CurrentLocale    : Uint4B
   +0x0c8 FpSoftwareStatusRegister : Uint4B
   +0x0cc ReservedForDebuggerInstrumentation : [16] Ptr32 Void
   +0x10c SystemReserved1  : [26] Ptr32 Void
   +0x174 PlaceholderCompatibilityMode : Char
   +0x175 PlaceholderHydrationAlwaysExplicit : UChar
   +0x176 PlaceholderReserved : [10] Char
   +0x180 ProxiedProcessId : Uint4B
   +0x184 _ActivationStack : _ACTIVATION_CONTEXT_STACK
   +0x19c WorkingOnBehalfTicket : [8] UChar
   +0x1a4 ExceptionCode    : Int4B
   +0x1a8 ActivationContextStackPointer : Ptr32 _ACTIVATION_CONTEXT_STACK
   +0x1ac InstrumentationCallbackSp : Uint4B
   +0x1b0 InstrumentationCallbackPreviousPc : Uint4B
   +0x1b4 InstrumentationCallbackPreviousSp : Uint4B
   +0x1b8 InstrumentationCallbackDisabled : UChar
   +0x1b9 SpareBytes       : [23] UChar
   +0x1d0 TxFsContext      : Uint4B
   +0x1d4 GdiTebBatch      : _GDI_TEB_BATCH
   +0x6b4 RealClientId     : _CLIENT_ID
   +0x6bc GdiCachedProcessHandle : Ptr32 Void
   +0x6c0 GdiClientPID     : Uint4B
   +0x6c4 GdiClientTID     : Uint4B
   +0x6c8 GdiThreadLocalInfo : Ptr32 Void
   +0x6cc Win32ClientInfo  : [62] Uint4B
   +0x7c4 glDispatchTable  : [233] Ptr32 Void
   +0xb68 glReserved1      : [29] Uint4B
```