常用结构体

STARTUPINFO

`CreateProcessA`的参数

```
typedef struct _STARTUPINFOA {
  0x00 DWORD   cb;                   // 结构体的大小
  0x04 LPSTR   lpReserved;
  0x08 LPSTR   lpDesktop;
  0x0c LPSTR   lpTitle;
  0x10 DWORD   dwX;
  0x14 DWORD   dwY;
  0x18 DWORD   dwXSize;
  0x1c DWORD   dwYSize;
  0x20 DWORD   dwXCountChars;
  0x24 DWORD   dwYCountChars;
  0x28 DWORD   dwFillAttribute;
  0x2c DWORD   dwFlags;              // STARTF_USESTDHANDLES:0x100 使最后三个句柄有效
                                     // STARTF_USESHOWWINDOW:0x1 使wShowWindow有效
  0x30 WORD    wShowWindow;
  0x32 WORD    cbReserved2;
  0x34 LPBYTE  lpReserved2;
  0x38 HANDLE  hStdInput;            // 进程的输入句柄
  0x3c HANDLE  hStdOutput;           // 进程的输出句柄
  0x40 HANDLE  hStdError;            // 进程的错误信息句柄
} STARTUPINFOA, *LPSTARTUPINFOA;
```

OSVERSIONINFOA

`GetVersionExA`的参数

```
typedef struct _OSVERSIONINFOA {
  0x00 DWORD dwOSVersionInfoSize;
  0x04 DWORD dwMajorVersion;
  0x08 DWORD dwMinorVersion;
  0x0c DWORD dwBuildNumber;
  0x10 DWORD dwPlatformId;       // VER_PLATFORM_WIN32_NT: 2
  0x14 CHAR  szCSDVersion[128];
} OSVERSIONINFOA, *POSVERSIONINFOA, *LPOSVERSIONINFOA;
```

CONTEXT

`SetThreadContext`的参数

```
typedef struct _CONTEXT
{
     ULONG ContextFlags;
     ULONG Dr0;
```

```
    ULONG Dr1;
    ULONG Dr2;
    ULONG Dr3;
    ULONG Dr6;
    ULONG Dr7;
    FLOATING_SAVE_AREA FloatSave;
    ULONG SegGs;
    ULONG SegFs;
    ULONG SegEs;
    ULONG SegDs;
    ULONG Edi;
    ULONG Esi;
    ULONG Ebx;          // 睡眠状态的进程，EBX寄存器存储的是指向PEB的指针
    ULONG Edx;
    ULONG Ecx;
    ULONG Eax;          // 睡眠状态的进程，EAX寄存器存储的是入口点地址
    ULONG Ebp;
    ULONG Eip;
    ULONG SegCs;
    ULONG EFlags;
    ULONG Esp;
    ULONG SegSs;
    UCHAR ExtendedRegisters[512];
} CONTEXT, *PCONTEXT;
```

TEB

fs[0]指向的就是TEB结构体

```
typedef struct _TEB
{
    0x00  NT_TIB NtTib;                // 指向TIB结构，而第一个元素是异常处理函数链表
    0x1c  PVOID EnvironmentPointer;
    0x20  CLIENT_ID ClientId;
    0x28  PVOID ActiveRpcHandle;
    0x2c  PVOID ThreadLocalStoragePointer;
    0x30  PPEB ProcessEnvironmentBlock;    // 指向PEB结构
    0x34  ULONG LastErrorValue;
    0x38  ULONG CountOfOwnedCriticalSections;
    0x3c  PVOID CsrClientThread;
    0x40  PVOID Win32ThreadInfo;
    0x44  ULONG User32Reserved[26];
    0xac  ULONG UserReserved[5];
    0xc0  PVOID WOW32Reserved;
    0xc4  ULONG CurrentLocale;
    0xc8  ULONG FpSoftwareStatusRegister;
    0xcc  VOID * SystemReserved1[54];
    0x1a4 LONG ExceptionCode;
    0x1a8 PACTIVATION_CONTEXT_STACK ActivationContextStackPointer;
    0x1ac UCHAR SpareBytes1[36];
    ULONG TxFsContext;
    GDI_TEB_BATCH GdiTebBatch;
    CLIENT_ID RealClientId;
    PVOID GdiCachedProcessHandle;
    ULONG GdiClientPID;
    ULONG GdiClientTID;
    PVOID GdiThreadLocalInfo;
    ULONG Win32ClientInfo[62];
    VOID * glDispatchTable[233];
    ULONG glReserved1[29];
    PVOID glReserved2;
    PVOID glSectionInfo;
    PVOID glSection;
    PVOID glTable;
    PVOID glCurrentRC;
    PVOID glContext;
    ULONG LastStatusValue;
    UNICODE_STRING StaticUnicodeString;
    WCHAR StaticUnicodeBuffer[261];
    PVOID DeallocationStack;
    VOID * TlsSlots[64];
    LIST_ENTRY TlsLinks;
    PVOID Vdm;
    PVOID ReservedForNtRpc;
    VOID * DbgSsReserved[2];
    ULONG HardErrorMode;
    VOID * Instrumentation[9];
    GUID ActivityId;
    PVOID SubProcessTag;
    PVOID EtwLocalData;
    PVOID EtwTraceData;
    PVOID WinSockData;
    ULONG GdiBatchCount;
    UCHAR SpareBool0;
    UCHAR SpareBool1;
    UCHAR SpareBool2;
    UCHAR IdealProcessor;
    ULONG GuaranteedStackBytes;
    PVOID ReservedForPerf;
    PVOID ReservedForOle;
    ULONG WaitingOnLoaderLock;
    PVOID SavedPriorityState;
    ULONG SoftPatchPtr1;
    PVOID ThreadPoolData;
    VOID * * TlsExpansionSlots;
    ULONG ImpersonationLocale;
    ULONG IsImpersonating;
    PVOID NlsCache;
    PVOID pShimData;
    ULONG HeapVirtualAffinity;
    PVOID CurrentTransactionHandle;
    PTEB_ACTIVE_FRAME ActiveFrame;
    PVOID FlsData;
    PVOID PreferredLanguages;
    PVOID UserPrefLanguages;
    PVOID MergedPrefLanguages;
    ULONG MuiImpersonation;
    WORD CrossTebFlags;
    ULONG SpareCrossTebBits: 16;
    WORD SameTebFlags;
    ULONG DbgSafeThunkCall: 1;
    ULONG DbgInDebugPrint: 1;
    ULONG DbgHasFiberData: 1;
    ULONG DbgSkipThreadAttach: 1;
    ULONG DbgWerInShipAssertCode: 1;
    ULONG DbgRanProcessInit: 1;
    ULONG DbgClonedThread: 1;
    ULONG DbgSuppressDebugMsg: 1;
    ULONG SpareSameTebBits: 8;
    PVOID TxnScopeEnterCallback;
    PVOID TxnScopeExitCallback;
    PVOID TxnScopeContext;
    ULONG LockCount;
    ULONG ProcessRundown;
    UINT64 LastSwitchTime;
    UINT64 TotalSwitchOutTime;
    LARGE_INTEGER WaitReasonBitMap;
} TEB, *PTEB;
```

PEB

```
typedef struct __PEB // 65 elements, 0x210 bytes
{
   0x00 BYTE bInheritedAddressSpace;
   0x01 BYTE bReadImageFileExecOptions;
   0x02 BYTE bBeingDebugged;                   // 进程是否被调试
   0x03 BYTE bSpareBool;
   0x04 LPVOID lpMutant;
   0x08 LPVOID lpImageBaseAddress;
   0x0c PPEB_LDR_DATA pLdr;                    // 指向PEB_LDR_Data结构
   0x10 LPVOID lpProcessParameters;
   0x14 LPVOID lpSubSystemData;
   0x18 LPVOID lpProcessHeap;                  // 指向进程的堆结构，其头部可用于检测调试
   0x1c PRTL_CRITICAL_SECTION pFastPebLock;
   0x20 LPVOID lpFastPebLockRoutine;
   0x24 LPVOID lpFastPebUnlockRoutine;
   0x28 DWORD dwEnvironmentUpdateCount;
   0x2c LPVOID lpKernelCallbackTable;
   0x30 DWORD dwSystemReserved;
   0x34 DWORD dwAtlThunkSListPtr32;
   0x38 PPEB_FREE_BLOCK pFreeList;
   0x3c DWORD dwTlsExpansionCounter;
   0x40 LPVOID lpTlsBitmap;
   0x44 DWORD dwTlsBitmapBits[2];
   0x4c LPVOID lpReadOnlySharedMemoryBase;
   0x50 LPVOID lpReadOnlySharedMemoryHeap;
   0x54 LPVOID lpReadOnlyStaticServerData;
   0x58 LPVOID lpAnsiCodePageData;
   0x5c LPVOID lpOemCodePageData;
   0x60 LPVOID lpUnicodeCaseTableData;
   0x64 DWORD dwNumberOfProcessors;
   0x68 DWORD dwNtGlobalFlag;                  // 若为0x70，表示在使用调试器
   LARGE_INTEGER liCriticalSectionTimeout;
   DWORD dwHeapSegmentReserve;
   DWORD dwHeapSegmentCommit;
   DWORD dwHeapDeCommitTotalFreeThreshold;
   DWORD dwHeapDeCommitFreeBlockThreshold;
   DWORD dwNumberOfHeaps;
   DWORD dwMaximumNumberOfHeaps;
   LPVOID lpProcessHeaps;
```

```
    LPVOID lpGdiSharedHandleTable;
    LPVOID lpProcessStarterHelper;
    DWORD dwGdiDCAttributeList;
    LPVOID lpLoaderLock;
    DWORD dwOSMajorVersion;
    DWORD dwOSMinorVersion;
    WORD wOSBuildNumber;
    WORD wOSCSDVersion;
    DWORD dwOSPlatformId;
    DWORD dwImageSubsystem;
    DWORD dwImageSubsystemMajorVersion;
    DWORD dwImageSubsystemMinorVersion;
    DWORD dwImageProcessAffinityMask;
    DWORD dwGdiHandleBuffer[34];
    LPVOID lpPostProcessInitRoutine;
    LPVOID lpTlsExpansionBitmap;
    DWORD dwTlsExpansionBitmapBits[32];
    DWORD dwSessionId;
    ULARGE_INTEGER liAppCompatFlags;
    ULARGE_INTEGER liAppCompatFlagsUser;
    LPVOID lppShimData;
    LPVOID lpAppCompatInfo;
    UNICODE_STR usCSDVersion;
    LPVOID lpActivationContextData;
    LPVOID lpProcessAssemblyStorageMap;
    LPVOID lpSystemDefaultActivationContextData;
    LPVOID lpSystemAssemblyStorageMap;
    DWORD dwMinimumStackCommit;
} _PEB, * _PPEB;
```

PEB_LDR_DATA

```
typedef struct _PEB_LDR_DATA {
  0x00 ULONG                    Length;
  0x04 BOOLEAN                  Initialized;
  0x08 PVOID                    SsHandle;
  0x0c LIST_ENTRY               InLoadOrderModuleList;
  0x14 LIST_ENTRY               InMemoryOrderModuleList;
  0x1c LIST_ENTRY               InInitializationOrderModuleList;
} PEB_LDR_DATA, *PPEB_LDR_DATA;
```

IP_ADAPTER_INFO

GetAdaptersInfo函数的参数

```
typedef struct _IP_ADAPTER_INFO {
  0x00  struct _IP_ADAPTER_INFO *Next;
  0x04  DWORD             ComboIndex;
  0x08  char              AdapterName[MAX_ADAPTER_NAME_LENGTH + 4];
  0x10c char              Description[MAX_ADAPTER_DESCRIPTION_LENGTH + 4];
  0x190 UINT              AddressLength;
  0x194 BYTE              Address[MAX_ADAPTER_ADDRESS_LENGTH];
  0x19c DWORD             Index;
  0x1a0 UINT              Type;       // MIB_IF_TYPE_ETHERNET: 6
                                      // IF_TYPE_IEEE80211: 71
  0x1a4 UINT              DhcpEnabled;
  0x1a8 PIP_ADDR_STRING  CurrentIpAddress;
  IP_ADDR_STRING          IpAddressList;
  IP_ADDR_STRING          GatewayList;
  IP_ADDR_STRING          DhcpServer;
  BOOL                    HaveWins;
  IP_ADDR_STRING          PrimaryWinsServer;
  IP_ADDR_STRING          SecondaryWinsServer;
  time_t                  LeaseObtained;
  time_t                  LeaseExpires;
} IP_ADAPTER_INFO, *PIP_ADAPTER_INFO;
```

FpuSavaState

```
struct FpuSaveState {
    0x00 uint32_t    control_word;
    0x04 uint32_t    status_word;
    0x08 uint32_t    tag_word;
    0x0c uint32_t    fpu_instruction_pointer;   // 上一次执行浮点运算的地址
    0x10 uint16_t    fpu_instruction_selector;
    0x12 uint16_t    fpu_opcode;
    0x14 uint32_t    fpu_operand_pointer;
    0x18 uint16_t    fpu_operand_selector;
    0x1a uint16_t    reserved;
};
```

sockaddr_in

```
struct sockaddr_in {
    short   sin_family;         // AF_INET: 2
    u_short sin_port;
    struct  in_addr sin_addr;
    char    sin_zero[8];
};
```