Win32 Thread Information Block

From Wikipedia, the free encyclopedia

In computing, the **Win32 Thread Information Block** (TIB) is a data structure in Win32 on x86 that stores information about the currently running thread. This structure is also known as the Thread Environment Block (TEB).[1]

The TIB is officially undocumented for Windows 9x. The Windows NT series DDK includes a struct NT_TIB in winnt.h that documents the subsystem independent part. Wine includes declarations for the extended (subsystem-specific part of) TIB. Yet so many Win32 programs use these undocumented fields that they are effectively a part of the API. The first field, in particular, is directly referenced by the code produced by Microsoft's own compiler.[1]

The TIB can be used to get a lot of information on the process without calling Win32 API. Examples include emulating GetLastError(), GetVersion(). Through the pointer to the PEB one can obtain access to the import tables (IAT), process startup arguments, image name, etc.

Contents

Contents of the TIB (32-bit Windows)

Position	Length	Windows Versions	Description
FS:[0x00]	4	Win9x and NT	Current Structured Exception Handling (SEH) frame
FS:[0x04]	4	Win9x and NT	Stack Base / Bottom of stack (high address)
FS:[0x08]	4	Win9x and NT	Stack Limit / Ceiling of stack (low address)
FS:[0x0C]	4	NT	SubSystemTib
FS:[0x10]	4	NT	Fiber data
FS:[0x14]	4	Win9x and NT	Arbitrary data slot
FS:[0x18]	4	Win9x and NT	Linear address of TIB
---- End of NT subsystem independent part ----			
FS:[0x1C]	4	NT	Environment Pointer
FS:[0x20]	4	NT	Process ID (in some windows distributions this field is used as 'DebugContext')
FS:[0x24]	4	NT	Current thread ID
FS:[0x28]	4	NT	Active RPC Handle
FS:[0x2C]	4	Win9x and NT	Linear address of the thread-local storage array
FS:[0x30]	4	NT	Linear address of Process Environment Block (PEB)
FS:[0x34]	4	NT	Last error number
FS:[0x38]	4	NT	Count of owned critical sections
FS:[0x3C]	4	NT	Address of CSR Client Thread
FS:[0x40]	4	NT	Win32 Thread Information
FS:[0x44]	124	NT, Wine	Win32 client information (NT), user32 private data (Wine), 0x60 = LastError (Win95), 0x74 = LastError (WinME)
FS:[0xC0]	4	NT	Reserved for Wow64. Contains a pointer to FastSysCall in Wow64.
FS:[0xC4]	4	NT	Current Locale
FS:[0xC8]	4	NT	FP Software Status Register
FS:[0xCC]	216	NT, Wine	Reserved for OS (NT), kernel32 private data (Wine) herein: FS:[0x124] 4 NT Pointer to KTHREAD (ETHREAD) structure
FS:[0x1A4]	4	NT	Exception code
FS:[0x1A8]	18	NT	Activation context stack
FS:	24	NT, Wine	Spare bytes (NT), ntdll private data (Wine)

[0x1BC]			
FS:[0x1D4]	40	NT, Wine	Reserved for OS (NT), ntdll private data (Wine)
FS:[0x1FC]	1248	NT, Wine	GDI TEB Batch (OS), vm86 private data (Wine)
FS:[0x6DC]	4	NT	GDI Region
FS:[0x6E0]	4	NT	GDI Pen
FS:[0x6E4]	4	NT	GDI Brush
FS:[0x6E8]	4	NT	Real Process ID
FS:[0x6EC]	4	NT	Real Thread ID
FS:[0x6F0]	4	NT	GDI cached process handle
FS:[0x6F4]	4	NT	GDI client process ID (PID)
FS:[0x6F8]	4	NT	GDI client thread ID (TID)
FS:[0x6FC]	4	NT	GDI thread locale information
FS:[0x700]	20	NT	Reserved for user application
FS:[0x714]	1248	NT	Reserved for GL
FS:[0xBF4]	4	NT	Last Status Value
FS:[0xBF8]	532	NT	Static UNICODE_STRING buffer
FS:[0xE0C]	4	NT	Pointer to deallocation stack
FS:[0xE10]	256	NT	TLS slots, 4 byte per slot
FS:[0xF10]	8	NT	TLS links (LIST_ENTRY structure)
FS:[0xF18]	4	NT	VDM
FS:[0xF1C]	4	NT	Reserved for RPC
FS:[0xF28]	4	NT	Thread error mode (RtlSetThreadErrorMode)

FS maps to a TIB which is embedded in a data block known as the TDB (thread data base). The TIB contains the thread-specific exception handling chain and pointer to the TLS (thread local storage.) The thread local storage is not the same as C local storage.

Note: The above description ONLY refers to 32-bit Windows on x86. On x86-64 (64-bit) Windows, GS (and not FS) is used as the segment register that points to the TIB. Additionally some of the variable slots in the structure above have a different size (typically 8 instead of 4 bytes for pointer-sized data slots).

Accessing the TIB

The TIB of the current thread can be accessed as an offset of segment register FS (x86) or GS (x64).

It is not common to access the TIB fields by an offset from FS:[0], but rather first getting a linear self-referencing pointer to it stored at FS:[0x18]. That pointer can be used with pointer arithmetics or be cast to a struct pointer.

Example in C inlined-assembly for 32-bit x86:

```
// gcc (AT&T-style inline assembly).
void *getTIB() {
    void *pTIB;
    __asm__("movl %%fs:0x18, %0" : "=r" (pTIB) : : );
    return pTIB;
}
```

```
// Microsoft C
void *getTIB() {
    void *pTIB;
    __asm {
        mov EAX, FS:[0x18]
        mov pTIB, EAX
    }
    return pTIB;
}
```

```
// Using Microsoft's intrinsics instead of inline assembly
void *getTIB() {
    return = (void *)__readfsdword(0x18);
}
```

See also

- Structured Exception Handling

References

1. Pietrek, Matt (May 1996). "Under The Hood". *Microsoft Systems Journal*. Retrieved 2010-07-07.

Further reading

- Pietrek, Matt (March 1996). *Windows 95 Programming Secrets* (pdf). IDG. pp. 136–138. ISBN 1-56884-318-6. Retrieved 2010-07-17.

External links

- TEB layout on NTinternals.net (http://undocumented.ntinternals.net/UserMode/Undocumented%20Functions/NT%20Objects/Thread/TEB.html)
- Structured Exception Handling and the TIB (http://www.microsoft.com/msj/0197/exception/exception.aspx)

Categories: Microsoft application programming interfaces | Threads (computing)