APPENDIX C

Constants, Enumerations, and Structures

The code samples and descriptions in this book make frequent references to data definitions from the Windows 2000 Device Driver Kit (DDK), the Win32 Platform Software Development Kit (SDK), and header files found on the companion CD of this book. To allow easy lookup of these definitions, I have compiled the most important ones in Appendix C. Most of the definitions are drawn from the `w2k_def.h` header file found in the CD directory `\src\common\include`.

CONSTANTS

This section contains definitions of symbolic constants used throughout the book. They are also referred to in subsequent sections of this appendix.

Dispatcher Object Type Codes

```
#define DISP_TYPE_NOTIFICATION_EVENT         0
#define DISP_TYPE_SYNCHRONIZATION_EVENT      1
#define DISP_TYPE_MUTANT                     2
#define DISP_TYPE_PROCESS                    3
#define DISP_TYPE_QUEUE                      4
#define DISP_TYPE_SEMAPHORE                  5
#define DISP_TYPE_THREAD                     6
#define DISP_TYPE_NOTIFICATION_TIMER         8
#define DISP_TYPE_SYNCHRONIZATION_TIMER      9
```

File Object Flags

```
#define FO_FILE_OPEN                        0x00000001
#define FO_SYNCHRONOUS_IO                   0x00000002
#define FO_ALERTABLE_IO                     0x00000004
#define FO_NO_INTERMEDIATE_BUFFERING        0x00000008
#define FO_WRITE_THROUGH                    0x00000010
#define FO_SEQUENTIAL_ONLY                  0x00000020
#define FO_CACHE_SUPPORTED                  0x00000040
#define FO_NAMED_PIPE                       0x00000080
#define FO_STREAM_FILE                      0x00000100
#define FO_MAILSLOT                         0x00000200
#define FO_GENERATE_AUDIT_ON_CLOSE          0x00000400
#define FO_DIRECT_DEVICE_OPEN               0x00000800
#define FO_FILE_MODIFIED                    0x00001000
#define FO_FILE_SIZE_CHANGED                0x00002000
#define FO_CLEANUP_COMPLETE                 0x00004000
#define FO_TEMPORARY_FILE                   0x00008000
#define FO_DELETE_ON_CLOSE                  0x00010000
#define FO_OPENED_CASE_SENSITIVE            0x00020000
#define FO_HANDLE_CREATED                   0x00040000
#define FO_FILE_FAST_IO_READ                0x00080000
#define FO_RANDOM_ACCESS                    0x00100000
#define FO_FILE_OPEN_CANCELLED              0x00200000
#define FO_VOLUME_OPEN                      0x00400000
```

Portable Executable Section Directory IDs

```
#define IMAGE_DIRECTORY_ENTRY_EXPORT          0
#define IMAGE_DIRECTORY_ENTRY_IMPORT          1
#define IMAGE_DIRECTORY_ENTRY_RESOURCE        2
#define IMAGE_DIRECTORY_ENTRY_EXCEPTION       3
#define IMAGE_DIRECTORY_ENTRY_SECURITY        4
#define IMAGE_DIRECTORY_ENTRY_BASERELOC       5
#define IMAGE_DIRECTORY_ENTRY_DEBUG           6
#define IMAGE_DIRECTORY_ENTRY_COPYRIGHT       7
#define IMAGE_DIRECTORY_ENTRY_GLOBALPTR       8
#define IMAGE_DIRECTORY_ENTRY_TLS             9
#define IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG    10
#define IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT   11
#define IMAGE_DIRECTORY_ENTRY_IAT            12
#define IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT   13
#define IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR 14

#define IMAGE_NUMBEROF_DIRECTORY_ENTRIES     16
```

I/O System Data Structure Type Codes

```
#define IO_TYPE_ADAPTER                    1
#define IO_TYPE_CONTROLLER                 2
#define IO_TYPE_DEVICE                     3
#define IO_TYPE_DRIVER                     4
#define IO_TYPE_FILE                       5
#define IO_TYPE_IRP                        6
#define IO_TYPE_MASTER_ADAPTER             7
#define IO_TYPE_OPEN_PACKET                8
#define IO_TYPE_TIMER                      9
#define IO_TYPE_VPB                       10
#define IO_TYPE_ERROR_LOG                 11
#define IO_TYPE_ERROR_MESSAGE             12
#define IO_TYPE_DEVICE_OBJECT_EXTENSION   13

#define IO_TYPE_APC                       18
#define IO_TYPE_DPC                       19
#define IO_TYPE_DEVICE_QUEUE              20
#define IO_TYPE_EVENT_PAIR                21
#define IO_TYPE_INTERRUPT                 22
#define IO_TYPE_PROFILE                   23
```

I/O Request Packet Functions

```
#define IRP_MJ_CREATE                      0
#define IRP_MJ_CREATE_NAMED_PIPE           1
#define IRP_MJ_CLOSE                       2
#define IRP_MJ_READ                        3
#define IRP_MJ_WRITE                       4
#define IRP_MJ_QUERY_INFORMATION           5
#define IRP_MJ_SET_INFORMATION             6
#define IRP_MJ_QUERY_EA                    7
#define IRP_MJ_SET_EA                      8
#define IRP_MJ_FLUSH_BUFFERS               9
#define IRP_MJ_QUERY_VOLUME_INFORMATION   10
#define IRP_MJ_SET_VOLUME_INFORMATION     11
#define IRP_MJ_DIRECTORY_CONTROL          12
#define IRP_MJ_FILE_SYSTEM_CONTROL        13
#define IRP_MJ_DEVICE_CONTROL             14
#define IRP_MJ_INTERNAL_DEVICE_CONTROL    15
#define IRP_MJ_SHUTDOWN                   16
#define IRP_MJ_LOCK_CONTROL               17
#define IRP_MJ_CLEANUP                    18
#define IRP_MJ_CREATE_MAILSLOT            19
#define IRP_MJ_QUERY_SECURITY             20
```

```
#define IRP_MJ_SET_SECURITY                21
#define IRP_MJ_POWER                       22
#define IRP_MJ_SYSTEM_CONTROL              23
#define IRP_MJ_DEVICE_CHANGE               24
#define IRP_MJ_QUERY_QUOTA                 25
#define IRP_MJ_SET_QUOTA                   26
#define IRP_MJ_PNP                         27
#define IRP_MJ_MAXIMUM_FUNCTION            27

#define IRP_MJ_FUNCTIONS (IRP_MJ_MAXIMUM_FUNCTION + 1)
```

Object Header Flags

```
#define OB_FLAG_CREATE_INFO    0x01 // has OBJECT_CREATE_INFO
#define OB_FLAG_KERNEL_MODE    0x02 // created by kernel
#define OB_FLAG_CREATOR_INFO   0x04 // has OBJECT_CREATOR_INFO
#define OB_FLAG_EXCLUSIVE      0x08 // OBJ_EXCLUSIVE
#define OB_FLAG_PERMANENT      0x10 // OBJ_PERMANENT
#define OB_FLAG_SECURITY       0x20 // has security descriptor
#define OB_FLAG_SINGLE_PROCESS 0x40 // no HandleDBList
```

Object Type Array Indexes

```
#define OB_TYPE_INDEX_TYPE              1 // [ObjT] "Type"
#define OB_TYPE_INDEX_DIRECTORY         2 // [Dire] "Directory"
#define OB_TYPE_INDEX_SYMBOLIC_LINK     3 // [Symb] "SymbolicLink"
#define OB_TYPE_INDEX_TOKEN             4 // [Toke] "Token"
#define OB_TYPE_INDEX_PROCESS           5 // [Proc] "Process"
#define OB_TYPE_INDEX_THREAD            6 // [Thre] "Thread"
#define OB_TYPE_INDEX_JOB               7 // [Job ] "Job"
#define OB_TYPE_INDEX_EVENT             8 // [Even] "Event"
#define OB_TYPE_INDEX_EVENT_PAIR        9 // [Even] "EventPair"
#define OB_TYPE_INDEX_MUTANT           10 // [Muta] "Mutant"
#define OB_TYPE_INDEX_CALLBACK         11 // [Call] "Callback"
#define OB_TYPE_INDEX_SEMAPHORE        12 // [Sema] "Semaphore"
#define OB_TYPE_INDEX_TIMER            13 // [Time] "Timer"
#define OB_TYPE_INDEX_PROFILE          14 // [Prof] "Profile"
#define OB_TYPE_INDEX_WINDOW_STATION   15 // [Wind] "WindowStation"
#define OB_TYPE_INDEX_DESKTOP          16 // [Desk] "Desktop"
#define OB_TYPE_INDEX_SECTION          17 // [Sect] "Section"
#define OB_TYPE_INDEX_KEY              18 // [Key ] "Key"
#define OB_TYPE_INDEX_PORT             19 // [Port] "Port"
#define OB_TYPE_INDEX_WAITABLE_PORT    20 // [Wait] "WaitablePort"
#define OB_TYPE_INDEX_ADAPTER          21 // [Adap] "Adapter"
#define OB_TYPE_INDEX_CONTROLLER       22 // [Cont] "Controller"
#define OB_TYPE_INDEX_DEVICE           23 // [Devi] "Device"
```

```
#define OB_TYPE_INDEX_DRIVER           24 // [Driv] "Driver"
#define OB_TYPE_INDEX_IO_COMPLETION    25 // [IoCo] "IoCompletion"
#define OB_TYPE_INDEX_FILE             26 // [File] "File"
#define OB_TYPE_INDEX_WMI_GUID         27 // [WmiG] "WmiGuid"
```

Object Type Tags

```
#define OB_TYPE_TAG_TYPE           'TjbO' // [ObjT] "Type"
#define OB_TYPE_TAG_DIRECTORY      'eriD' // [Dire] "Directory"
#define OB_TYPE_TAG_SYMBOLIC_LINK  'bmyS' // [Symb] "SymbolicLink"
#define OB_TYPE_TAG_TOKEN          'ekoT' // [Toke] "Token"
#define OB_TYPE_TAG_PROCESS        'corP' // [Proc] "Process"
#define OB_TYPE_TAG_THREAD         'erhT' // [Thre] "Thread"
#define OB_TYPE_TAG_JOB            ' boJ' // [Job ] "Job"
#define OB_TYPE_TAG_EVENT          'nevE' // [Even] "Event"
#define OB_TYPE_TAG_EVENT_PAIR     'nevE' // [Even] "EventPair"
#define OB_TYPE_TAG_MUTANT         'atuM' // [Muta] "Mutant"
#define OB_TYPE_TAG_CALLBACK       'llaC' // [Call] "Callback"
#define OB_TYPE_TAG_SEMAPHORE      'ameS' // [Sema] "Semaphore"
#define OB_TYPE_TAG_TIMER          'emiT' // [Time] "Timer"
#define OB_TYPE_TAG_PROFILE        'forP' // [Prof] "Profile"
#define OB_TYPE_TAG_WINDOW_STATION 'dniW' // [Wind] "WindowStation"
#define OB_TYPE_TAG_DESKTOP        'kseD' // [Desk] "Desktop"
#define OB_TYPE_TAG_SECTION        'tceS' // [Sect] "Section"
#define OB_TYPE_TAG_KEY            ' yeK' // [Key ] "Key"
#define OB_TYPE_TAG_PORT           'troP' // [Port] "Port"
#define OB_TYPE_TAG_WAITABLE_PORT  'tiaW' // [Wait] "WaitablePort"
#define OB_TYPE_TAG_ADAPTER        'padA' // [Adap] "Adapter"
#define OB_TYPE_TAG_CONTROLLER     'tnoC' // [Cont] "Controller"
#define OB_TYPE_TAG_DEVICE         'iveD' // [Devi] "Device"
#define OB_TYPE_TAG_DRIVER         'virD' // [Driv] "Driver"
#define OB_TYPE_TAG_IO_COMPLETION  'oCoI' // [IoCo] "IoCompletion"
#define OB_TYPE_TAG_FILE           'eliF' // [File] "File"
#define OB_TYPE_TAG_WMI_GUID       'GimW' // [WmiG] "WmiGuid"
```

Object Attribute Flags

```
#define OBJ_INHERIT          0x00000002
#define OBJ_PERMANENT        0x00000010
#define OBJ_EXCLUSIVE        0x00000020
#define OBJ_CASE_INSENSITIVE 0x00000040
#define OBJ_OPENIF           0x00000080
#define OBJ_OPENLINK         0x00000100
#define OBJ_KERNEL_HANDLE    0x00000200
#define OBJ_VALID_ATTRIBUTES 0x000003F2
```

ENUMERATIONS

Some Windows 2000 constant definitions come in the form of enumerations. Following is an alphabetical collection of the most frequently used ones. The effective values of the enumeration members are shown in comments inserted before each definition line.

IO_ALLOCATION_ACTION

```
typedef enum _IO_ALLOCATION_ACTION
        {
/*001*/ KeepObject = 1,
/*002*/ DeallocateObject,
/*003*/ DeallocateObjectKeepRegisters
        }
        IO_ALLOCATION_ACTION;
```

LOOKASIDE_LIST_ID

```
typedef enum _LOOKASIDE_LIST_ID
        {
/*000*/ SmallIrpLookasideList,
/*001*/ LargeIrpLookasideList,
/*002*/ MdlLookasideList,
/*003*/ CreateInfoLookasideList,
/*004*/ NameBufferLookasideList,
/*005*/ TwilightLookasideList,
/*006*/ CompletionLookasideList
        }
        LOOKASIDE_LIST_ID;
```

MODE (SEE ALSO KPROCESSOR_MODE)

```
typedef enum _MODE
        {
/*000*/ KernelMode,
/*001*/ UserMode,
/*002*/ MaximumMode
        }
        MODE;
```

NT_PRODUCT_TYPE

```
typedef enum _NT_PRODUCT_TYPE
        {
/*000*/ NtProductInvalid,
/*001*/ NtProductWinNt,
```

```
/*002*/ NtProductLanManNt,
/*003*/ NtProductServer
        }
        NT_PRODUCT_TYPE;
```

POOL_TYPE

```
typedef enum _POOL_TYPE
        {
/*000*/ NonPagedPool,
/*001*/ PagedPool,
/*002*/ NonPagedPoolMustSucceed,
/*003*/ DontUseThisType,
/*004*/ NonPagedPoolCacheAligned,
/*005*/ PagedPoolCacheAligned,
/*006*/ NonPagedPoolCacheAlignedMustS,
/*007*/ MaxPoolType
        }
        POOL_TYPE;
```

STRUCTURES AND ALIASES

This section is an alphabetical collection of structure and alias type definitions used by kernel-mode drivers and low-level system programs. Parts of them are undocumented. Before the member names, I have inserted comments indicating the offsets of all structure members relative to the structure base address. This allows easy read-out of member values from a hex dump listing.

ANSI_STRING

```
typedef STRING ANSI_STRING;
```

CALLBACK_OBJECT

```
typedef struct _CALLBACK_OBJECT
        {
/*000*/ DWORD      Tag; // 0x6C6C6143 ("Call")
/*004*/ KSPIN_LOCK Lock;
/*008*/ LIST_ENTRY CallbackList;
/*010*/ BOOLEAN    AllowMultipleCallbacks;
/*014*/ }
        CALLBACK_OBJECT;
```

CLIENT_ID

```
typedef struct _CLIENT_ID
        {
/*000*/ HANDLE UniqueProcess;
/*004*/ HANDLE UniqueThread;
/*008*/ }
        CLIENT_ID;
```

CONTEXT

```
// base address 0xFFDFF13C

#define MAXIMUM_SUPPORTED_EXTENSION 512

typedef struct _CONTEXT
        {
/*000*/ DWORD       ContextFlags;
/*004*/ DWORD       Dr0;
/*008*/ DWORD       Dr1;
/*00C*/ DWORD       Dr2;
/*010*/ DWORD       Dr3;
/*014*/ DWORD       Dr6;
/*018*/ DWORD       Dr7;
/*01C*/ FLOATING_SAVE_AREA FloatSave;
/*08C*/ DWORD       SegGs;
/*090*/ DWORD       SegFs;
/*094*/ DWORD       SegEs;
/*098*/ DWORD       SegDs;
/*09C*/ DWORD       Edi;
/*0A0*/ DWORD       Esi;
/*0A4*/ DWORD       Ebx;
/*0A8*/ DWORD       Edx;
/*0AC*/ DWORD       Ecx;
/*0B0*/ DWORD       Eax;
/*0B4*/ DWORD       Ebp;
/*0B8*/ DWORD       Eip;
/*0BC*/ DWORD       SegCs;
/*0C0*/ DWORD       EFlags;
/*0C4*/ DWORD       Esp;
/*0C8*/ DWORD       SegSs;
/*0CC*/ BYTE        ExtendedRegisters [MAXIMUM_SUPPORTED_EXTENSION];
/*2CC*/ }
        CONTEXT;
```

CONTROLLER_OBJECT

```
typedef struct _CONTROLLER_OBJECT
        {
/*000*/ SHORT         Type; // IO_TYPE_CONTROLLER 0x02
/*002*/ SHORT         Size; // number of BYTEs
/*004*/ PVOID         ControllerExtension;
/*008*/ KDEVICE_QUEUE DeviceWaitQueue;
/*01C*/ DWORD         Spare1;
/*020*/ LARGE_INTEGER Spare2;
/*028*/ }
        CONTROLLER_OBJECT;
```

CRITICAL_SECTION

```
typedef struct _RTL_CRITICAL_SECTION
        {
/*000*/ PRTL_CRITICAL_SECTION_DEBUG DebugInfo;
/*004*/ LONG                        LockCount;
/*008*/ LONG                        RecursionCount;
/*00C*/ HANDLE                      OwningThread;
/*010*/ HANDLE                      LockSemaphore;
/*014*/ DWORD_PTR                   SpinCount;
/*018*/ }
        CRITICAL_SECTION;
```

DEVICE_OBJECT

```
typedef struct _DEVICE_OBJECT
        {
/*000*/ SHORT                    Type; // IO_TYPE_DEVICE 0x03
/*002*/ WORD                     Size; // number of BYTEs
/*004*/ LONG                     ReferenceCount;
/*008*/ struct _DRIVER_OBJECT    *DriverObject;
/*00C*/ struct _DEVICE_OBJECT    *NextDevice;
/*010*/ struct _DEVICE_OBJECT    *AttachedDevice;
/*014*/ struct _IRP              *CurrentIrp;
/*018*/ struct _PIO_TIMER        *Timer;
/*01C*/ DWORD                    Flags;           // DO_*
/*020*/ DWORD                    Characteristics; // FILE_*
/*024*/ PVPB                     Vpb;
/*028*/ PVOID                    DeviceExtension;
/*02C*/ DEVICE_TYPE              DeviceType;
/*030*/ CHAR                     StackSize;
/*034*/ union
            {
```

```
/*034*/     LIST_ENTRY         ListEntry;
/*034*/     WAIT_CONTEXT_BLOCK Wcb;
/*05C*/     } Queue;
/*05C*/ DWORD                     AlignmentRequirement;
/*060*/ KDEVICE_QUEUE             DeviceQueue;
/*074*/ KDPC                      Dpc;
/*094*/ DWORD                     ActiveThreadCount;
/*098*/ PSECURITY_DESCRIPTOR      SecurityDescriptor;
/*09C*/ KEVENT                    DeviceLock;
/*0AC*/ WORD                      SectorSize;
/*0AE*/ WORD                      Spare1;
/*0B0*/ struct _DEVOBJ_EXTENSION *DeviceObjectExtension;
/*0B4*/ PVOID                     Reserved;
/*0B8*/ }
        DEVICE_OBJECT;
```

DEVOBJ_EXTENSION

```
typedef struct _DEVOBJ_EXTENSION
        {
/*000*/ SHORT          Type; // IO_TYPE_DEVICE_OBJECT_EXTENSION 0x0D
/*002*/ WORD           Size; // number of BYTEs
/*004*/ PDEVICE_OBJECT DeviceObject;
/*008*/ }
        DEVOBJ_EXTENSION;
```

DISPATCHER_HEADER

```
typedef struct _DISPATCHER_HEADER
        {
/*000*/ BYTE       Type;         // DISP_TYPE_*
/*001*/ BYTE       Absolute;
/*002*/ BYTE       Size;         // number of DWORDs
/*003*/ BYTE       Inserted;
/*004*/ LONG       SignalState;
/*008*/ LIST_ENTRY WaitListHead;
/*010*/ }
        DISPATCHER_HEADER;
```

DRIVER_EXTENSION

```
typedef struct _DRIVER_EXTENSION
        {
/*000*/ struct _DRIVER_OBJECT *DriverObject;
/*004*/ PDRIVER_ADD_DEVICE     AddDevice;
/*008*/ DWORD                  Count;
/*00C*/ UNICODE_STRING         ServiceKeyName;
/*014*/ }
        DRIVER_EXTENSION;
```

DRIVER_OBJECT

```
typedef struct _DRIVER_OBJECT
        {
/*000*/ SHORT              Type; // IO_TYPE_DRIVER 0x04
/*002*/ SHORT              Size; // number of BYTEs
/*004*/ PDEVICE_OBJECT     DeviceObject;
/*008*/ DWORD              Flags;
/*00C*/ PVOID              DriverStart;
/*010*/ DWORD              DriverSize;
/*014*/ PVOID              DriverSection;
/*018*/ PDRIVER_EXTENSION  DriverExtension;
/*01C*/ UNICODE_STRING     DriverName;
/*024*/ PUNICODE_STRING    HardwareDatabase;
/*028*/ PFAST_IO_DISPATCH  FastIoDispatch;
/*02C*/ PDRIVER_INITIALIZE DriverInit;
/*030*/ PDRIVER_STARTIO    DriverStartIo;
/*034*/ PDRIVER_UNLOAD     DriverUnload;
/*038*/ PDRIVER_DISPATCH   MajorFunction [IRP_MJ_FUNCTIONS];
/*0A8*/ }
        DRIVER_OBJECT;
```

EPROCESS

```
typedef struct _EPROCESS
        {
/*000*/ KPROCESS               Pcb;
/*06C*/ NTSTATUS               ExitStatus;
/*070*/ KEVENT                 LockEvent;
/*080*/ DWORD                  LockCount;
/*084*/ DWORD                  d084;
/*088*/ LARGE_INTEGER          CreateTime;
/*090*/ LARGE_INTEGER          ExitTime;
/*098*/ PVOID                  LockOwner;
/*09C*/ DWORD                  UniqueProcessId;
/*0A0*/ LIST_ENTRY             ActiveProcessLinks;
/*0A8*/ DWORD                  QuotaPeakPoolUsage [2]; // NP, P
/*0B0*/ DWORD                  QuotaPoolUsage     [2]; // NP, P
/*0B8*/ DWORD                  PagefileUsage;
/*0BC*/ DWORD                  CommitCharge;
/*0C0*/ DWORD                  PeakPagefileUsage;
/*0C4*/ DWORD                  PeakVirtualSize;
/*0C8*/ LARGE_INTEGER          VirtualSize;
/*0D0*/ MMSUPPORT              Vm;
/*100*/ DWORD                  d100;
/*104*/ DWORD                  d104;
/*108*/ DWORD                  d108;
/*10C*/ DWORD                  d10C;
/*110*/ DWORD                  d110;
/*114*/ DWORD                  d114;
```

```
/*118*/ DWORD                   d118;
/*11C*/ DWORD                   d11C;
/*120*/ PVOID                   DebugPort;
/*124*/ PVOID                   ExceptionPort;
/*128*/ PHANDLE_TABLE           ObjectTable;
/*12C*/ PVOID                   Token;
/*130*/ FAST_MUTEX              WorkingSetLock;
/*150*/ DWORD                   WorkingSetPage;
/*154*/ BOOLEAN                 ProcessOutswapEnabled;
/*155*/ BOOLEAN                 ProcessOutswapped;
/*156*/ BOOLEAN                 AddressSpaceInitialized;
/*157*/ BOOLEAN                 AddressSpaceDeleted;
/*158*/ FAST_MUTEX              AddressCreationLock;
/*178*/ KSPIN_LOCK              HyperSpaceLock;
/*17C*/ DWORD                   ForkInProgress;
/*180*/ WORD                    VmOperation;
/*182*/ BOOLEAN                 ForkWasSuccessful;
/*183*/ BYTE                    MmAgressiveWsTrimMask;
/*184*/ DWORD                   VmOperationEvent;
/*188*/ HARDWARE_PTE            PageDirectoryPte;
/*18C*/ DWORD                   LastFaultCount;
/*190*/ DWORD                   ModifiedPageCount;
/*194*/ PVOID                   VadRoot;
/*198*/ PVOID                   VadHint;
/*19C*/ PVOID                   CloneRoot;
/*1A0*/ DWORD                   NumberOfPrivatePages;
/*1A4*/ DWORD                   NumberOfLockedPages;
/*1A8*/ WORD                    NextPageColor;
/*1AA*/ BOOLEAN                 ExitProcessCalled;
/*1AB*/ BOOLEAN                 CreateProcessReported;
/*1AC*/ HANDLE                  SectionHandle;
/*1B0*/ struct _PEB            *Peb;
/*1B4*/ PVOID                   SectionBaseAddress;
/*1B8*/ PQUOTA_BLOCK            QuotaBlock;
/*1BC*/ NTSTATUS                LastThreadExitStatus;
/*1C0*/ DWORD                   WorkingSetWatch;
/*1C4*/ HANDLE                  Win32WindowStation;
/*1C8*/ DWORD                   InheritedFromUniqueProcessId;
/*1CC*/ ACCESS_MASK             GrantedAccess;
/*1D0*/ DWORD                   DefaultHardErrorProcessing; // HEM_*
/*1D4*/ DWORD                   LdtInformation;
/*1D8*/ PVOID                   VadFreeHint;
/*1DC*/ DWORD                   VdmObjects;
/*1E0*/ PVOID                   DeviceMap; // 0x24 bytes
/*1E4*/ DWORD                   SessionId;
/*1E8*/ DWORD                   d1E8;
/*1EC*/ DWORD                   d1EC;
```

```
/*1F0*/ DWORD                  d1F0;
/*1F4*/ DWORD                  d1F4;
/*1F8*/ DWORD                  d1F8;
/*1FC*/ BYTE                   ImageFileName [16];
/*20C*/ DWORD                  VmTrimFaultValue;
/*210*/ BYTE                   SetTimerResolution;
/*211*/ BYTE                   PriorityClass;
/*212*/ union
            {
            struct
                {
/*212*/         BYTE               SubSystemMinorVersion;
/*213*/         BYTE               SubSystemMajorVersion;
                };
            struct
                {
/*212*/         WORD               SubSystemVersion;
                };
            };
/*214*/ struct _WIN32_PROCESS *Win32Process;
/*218*/ DWORD                  d218;
/*21C*/ DWORD                  d21C;
/*220*/ DWORD                  d220;
/*224*/ DWORD                  d224;
/*228*/ DWORD                  d228;
/*22C*/ DWORD                  d22C;
/*230*/ PVOID                  Wow64;
/*234*/ DWORD                  d234;
/*238*/ IO_COUNTERS            IoCounters;
/*268*/ DWORD                  d268;
/*26C*/ DWORD                  d26C;
/*270*/ DWORD                  d270;
/*274*/ DWORD                  d274;
/*278*/ DWORD                  d278;
/*27C*/ DWORD                  d27C;
/*280*/ DWORD                  d280;
/*284*/ DWORD                  d284;
/*288*/ }
        EPROCESS;
```

ERESOURCE

```
typedef struct _ERESOURCE
        {
/*000*/ LIST_ENTRY   SystemResourcesList;
/*008*/ POWNER_ENTRY OwnerTable;
```

```
/*00C*/ SHORT         ActiveCount;
/*00E*/ WORD          Flag;
/*010*/ PKSEMAPHORE   SharedWaiters;
/*014*/ PKEVENT       ExclusiveWaiters;
/*018*/ OWNER_ENTRY   OwnerThreads [2];
/*028*/ DWORD         ContentionCount;
/*02C*/ WORD          NumberOfSharedWaiters;
/*02E*/ WORD          NumberOfExclusiveWaiters;
/*030*/ union
            {
/*030*/     PVOID     Address;
/*030*/     DWORD_PTR CreatorBackTraceIndex;
/*034*/     };
/*034*/ KSPIN_LOCK    SpinLock;
/*038*/ }
        ERESOURCE;
```

ERESOURCE_OLD

```
typedef struct _ERESOURCE_OLD
        {
/*000*/ LIST_ENTRY         SystemResourcesList;
/*008*/ PERESOURCE_THREAD  OwnerThreads;
/*00C*/ PBYTE              OwnerCounts;
/*010*/ WORD               TableSize;
/*012*/ WORD               ActiveCount;
/*014*/ WORD               Flag;
/*016*/ WORD               TableRover;
/*018*/ BYTE               InitialOwnerCounts  [4];
/*01C*/ ERESOURCE_THREAD   InitialOwnerThreads [4];
/*02C*/ DWORD              Spare1;
/*030*/ DWORD              ContentionCount;
/*034*/ WORD               NumberOfExclusiveWaiters;
/*036*/ WORD               NumberOfSharedWaiters;
/*038*/ KSEMAPHORE         SharedWaiters;
/*04C*/ KEVENT             ExclusiveWaiters;
/*05C*/ KSPIN_LOCK         SpinLock;
/*060*/ DWORD              CreatorBackTraceIndex;
/*064*/ WORD               Depth;
/*066*/ WORD               Reserved;
/*068*/ PVOID              OwnerBackTrace [4];
/*078*/ }
        ERESOURCE_OLD;
```

ERESOURCE_THREAD

```
typedef DWORD_PTR ERESOURCE_THREAD;
```

ETHREAD

```
typedef struct _ETHREAD
        {
/*000*/ KTHREAD       Tcb;
/*1B0*/ LARGE_INTEGER CreateTime;
/*1B8*/ union
            {
/*1B8*/     LARGE_INTEGER ExitTime;
/*1B8*/     LIST_ENTRY    LpcReplyChain;
            };
/*1C0*/ union
            {
/*1C0*/     NTSTATUS      ExitStatus;
/*1C0*/     DWORD         OfsChain;
            };
/*1C4*/ LIST_ENTRY    PostBlockList;
/*1CC*/ LIST_ENTRY    TerminationPortList;
/*1D4*/ PVOID         ActiveTimerListLock;
/*1D8*/ LIST_ENTRY    ActiveTimerListHead;
/*1E0*/ CLIENT_ID     Cid;
/*1E8*/ KSEMAPHORE    LpcReplySemaphore;
/*1FC*/ DWORD         LpcReplyMessage;
/*200*/ DWORD         LpcReplyMessageId;
/*204*/ DWORD         PerformanceCountLow;
/*208*/ DWORD         ImpersonationInfo;
/*20C*/ LIST_ENTRY    IrpList;
/*214*/ PVOID         TopLevelIrp;
/*218*/ PVOID         DeviceToVerify;
/*21C*/ DWORD         ReadClusterSize;
/*220*/ BOOLEAN       ForwardClusterOnly;
/*221*/ BOOLEAN       DisablePageFaultClustering;
/*222*/ BOOLEAN       DeadThread;
/*223*/ BOOLEAN       Reserved;
/*224*/ BOOL          HasTerminated;
/*228*/ ACCESS_MASK   GrantedAccess;
/*22C*/ PEPROCESS     ThreadsProcess;
/*230*/ PVOID         StartAddress;
/*234*/ union
            {
/*234*/     PVOID         Win32StartAddress;
/*234*/     DWORD         LpcReceivedMessageId;
            };
/*238*/ BOOLEAN       LpcExitThreadCalled;
/*239*/ BOOLEAN       HardErrorsAreDisabled;
/*23A*/ BOOLEAN       LpcReceivedMsgIdValid;
/*23B*/ BOOLEAN       ActiveImpersonationInfo;
/*23C*/ DWORD         PerformanceCountHigh;
/*240*/ DWORD         d240;
/*244*/ DWORD         d244;
/*248*/ }
        ETHREAD;
```

ETIMER

```
typedef struct _ETIMER
        {
/*000*/ KTIMER     Tcb;
/*028*/ KAPC       Apc;
/*058*/ KDPC       Dpc;
/*078*/ LIST_ENTRY ActiveTimerList;
/*080*/ KSPIN_LOCK Lock;
/*084*/ LONG       Period;
/*088*/ BOOLEAN    Active;
/*089*/ BOOLEAN    Resume;
/*08C*/ LIST_ENTRY WakeTimerList;
/*098*/ }
        ETIMER;
```

FAST_MUTEX

```
typedef struct _FAST_MUTEX
        {
/*000*/ LONG             Count;
/*004*/ struct _KTHREAD *Owner;
/*008*/ DWORD            Contention;
/*00C*/ KEVENT           Event;
/*01C*/ DWORD            OldIrql;
/*020*/ }
        FAST_MUTEX;
```

FILE_OBJECT

```
typedef struct _FILE_OBJECT
        {
/*000*/ SHORT                    Type; // IO_TYPE_FILE 0x05
/*002*/ SHORT                    Size; // number of BYTEs
/*004*/ PDEVICE_OBJECT           DeviceObject;
/*008*/ PVPB                     Vpb;
/*00C*/ PVOID                    FsContext;
/*010*/ PVOID                    FsContext2;
/*014*/ PSECTION_OBJECT_POINTERS SectionObjectPointer;
/*018*/ PVOID                    PrivateCacheMap;
/*01C*/ NTSTATUS                 FinalStatus;
/*020*/ struct _FILE_OBJECT     *RelatedFileObject;
/*024*/ BOOLEAN                  LockOperation;
/*025*/ BOOLEAN                  DeletePending;
/*026*/ BOOLEAN                  ReadAccess;
/*027*/ BOOLEAN                  WriteAccess;
/*028*/ BOOLEAN                  DeleteAccess;
/*029*/ BOOLEAN                  SharedRead;
```

```
/*02A*/ BOOLEAN                 SharedWrite;
/*02B*/ BOOLEAN                 SharedDelete;
/*02C*/ DWORD                   Flags; // FO_*
/*030*/ UNICODE_STRING          FileName;
/*038*/ LARGE_INTEGER           CurrentByteOffset;
/*040*/ DWORD                   Waiters;
/*044*/ DWORD                   Busy;
/*048*/ PVOID                   LastLock;
/*04C*/ KEVENT                  Lock;
/*05C*/ KEVENT                  Event;
/*06C*/ PIO_COMPLETION_CONTEXT  CompletionContext;
/*070*/ }
        FILE_OBJECT;
```

FLOATING_SAVE_AREA

```
// base address 0xFFDFF158

#define SIZE_OF_80387_REGISTERS 80

typedef struct _FLOATING_SAVE_AREA
        {
/*000*/ DWORD ControlWord;
/*004*/ DWORD StatusWord;
/*008*/ DWORD TagWord;
/*00C*/ DWORD ErrorOffset;
/*010*/ DWORD ErrorSelector;
/*014*/ DWORD DataOffset;
/*018*/ DWORD DataSelector;
/*01C*/ BYTE  RegisterArea [SIZE_OF_80387_REGISTERS];
/*06C*/ DWORD Cr0NpxState;
/*070*/ }
        FLOATING_SAVE_AREA;
```

HANDLE_ENTRY

```
#define HANDLE_ATTRIBUTE_INHERIT 0x00000002
#define HANDLE_ATTRIBUTE_MASK    0x00000007
#define HANDLE_OBJECT_MASK       0xFFFFFFF8

typedef struct _HANDLE_ENTRY // cf. OBJECT_HANDLE_INFORMATION
        {
/*000*/ union
            {
/*000*/     DWORD          HandleAttributes;// HANDLE_ATTRIBUTE_MASK
/*000*/     POBJECT_HEADER ObjectHeader;    // HANDLE_OBJECT_MASK
/*004*/     };
```

```
/*004*/ union
            {
/*004*/     ACCESS_MASK    GrantedAccess;   // if used entry
/*004*/     DWORD          NextEntry;       // if free entry
/*008*/     };
/*008*/ }
        HANDLE_ENTRY;
```

HANDLE_LAYER1, HANDLE_LAYER2, HANDLE_LAYER3

```
#define HANDLE_LAYER_SIZE 0x00000100

typedef struct _HANDLE_LAYER1
        {
/*000*/ PHANDLE_LAYER2 Layer2 [HANDLE_LAYER_SIZE]; // bits 18 to 25
/*400*/ }
        HANDLE_LAYER1;

typedef struct _HANDLE_LAYER2
        {
/*000*/ PHANDLE_LAYER3 Layer3 [HANDLE_LAYER_SIZE]; // bits 10 to 17
/*400*/ }
        HANDLE_LAYER2;

typedef struct _HANDLE_LAYER3
        {
/*000*/ HANDLE_ENTRY Entries [HANDLE_LAYER_SIZE]; // bits 2 to 9
/*800*/ }
        HANDLE_LAYER3;
```

HANDLE_TABLE

```
typedef struct _HANDLE_TABLE
        {
/*000*/ DWORD             Reserved;
/*004*/ DWORD             HandleCount;
/*008*/ PHANDLE_LAYER1    Layer1;
/*00C*/ struct _EPROCESS *Process; // passed to PsChargePoolQuota ()
/*010*/ HANDLE            UniqueProcessId;
/*014*/ DWORD             NextEntry;
/*018*/ DWORD             TotalEntries;
/*01C*/ ERESOURCE         HandleTableLock;
/*054*/ LIST_ENTRY        HandleTableList;
/*05C*/ KEVENT            Event;
/*06C*/ }
        HANDLE_TABLE;
```

HARDWARE_PTE

```
typedef struct _HARDWARE_PTE
        {
/*000*/ unsigned Valid           :  1;
        unsigned Write           :  1;
        unsigned Owner           :  1;
        unsigned WriteThrough    :  1;
        unsigned CacheDisable    :  1;
        unsigned Accessed        :  1;
        unsigned Dirty           :  1;
        unsigned LargePage       :  1;
/*001*/ unsigned Global          :  1;
        unsigned CopyOnWrite     :  1;
        unsigned Prototype       :  1;
        unsigned reserved        :  1;
        unsigned PageFrameNumber : 20;
/*004*/ }
        HARDWARE_PTE;
```

IMAGE_DATA_DIRECTORY

```
typedef struct _IMAGE_DATA_DIRECTORY
        {
/*000*/ DWORD VirtualAddress;
/*004*/ DWORD Size;
/*008*/ }
        IMAGE_DATA_DIRECTORY;
```

IMAGE_EXPORT_DIRECTORY

```
typedef struct _IMAGE_EXPORT_DIRECTORY
        {
/*000*/ DWORD Characteristics;
/*004*/ DWORD TimeDateStamp;
/*008*/ WORD  MajorVersion;
/*00A*/ WORD  MinorVersion;
/*00C*/ DWORD Name;
/*010*/ DWORD Base;
/*014*/ DWORD NumberOfFunctions;
/*018*/ DWORD NumberOfNames;
/*01C*/ DWORD AddressOfFunctions;
/*020*/ DWORD AddressOfNames;
/*024*/ DWORD AddressOfNameOrdinals;
/*028*/ }
        IMAGE_EXPORT_DIRECTORY;
```

IMAGE_FILE_HEADER

```
typedef struct _IMAGE_FILE_HEADER
        {
/*000*/ WORD  Machine;
/*002*/ WORD  NumberOfSections;
/*004*/ DWORD TimeDateStamp;
/*008*/ DWORD PointerToSymbolTable;
/*00C*/ DWORD NumberOfSymbols;
/*010*/ WORD  SizeOfOptionalHeader;
/*012*/ WORD  Characteristics;
/*014*/ }
        IMAGE_FILE_HEADER;
```

IMAGE_NT_HEADERS

```
typedef struct _IMAGE_NT_HEADERS
        {
/*000*/ DWORD                 Signature;
/*004*/ IMAGE_FILE_HEADER     FileHeader;
/*018*/ IMAGE_OPTIONAL_HEADER OptionalHeader;
/*0F8*/ }
        IMAGE_NT_HEADERS;
```

IMAGE_OPTIONAL_HEADER

```
#define IMAGE_NUMBEROF_DIRECTORY_ENTRIES        16

typedef struct _IMAGE_OPTIONAL_HEADER
        {
/*000*/ WORD                Magic;
/*002*/ BYTE                MajorLinkerVersion;
/*003*/ BYTE                MinorLinkerVersion;
/*004*/ DWORD               SizeOfCode;
/*008*/ DWORD               SizeOfInitializedData;
/*00C*/ DWORD               SizeOfUninitializedData;
/*010*/ DWORD               AddressOfEntryPoint;
/*014*/ DWORD               BaseOfCode;
/*018*/ DWORD               BaseOfData;
/*01C*/ DWORD               ImageBase;
/*020*/ DWORD               SectionAlignment;
/*024*/ DWORD               FileAlignment;
/*028*/ WORD                MajorOperatingSystemVersion;
/*02A*/ WORD                MinorOperatingSystemVersion;
/*02C*/ WORD                MajorImageVersion;
/*02E*/ WORD                MinorImageVersion;
/*030*/ WORD                MajorSubsystemVersion;
/*032*/ WORD                MinorSubsystemVersion;
/*034*/ DWORD               Win32VersionValue;
```

```
/*038*/ DWORD                SizeOfImage;
/*03C*/ DWORD                SizeOfHeaders;
/*040*/ DWORD                CheckSum;
/*044*/ WORD                 Subsystem;
/*046*/ WORD                 DllCharacteristics;
/*048*/ DWORD                SizeOfStackReserve;
/*04C*/ DWORD                SizeOfStackCommit;
/*050*/ DWORD                SizeOfHeapReserve;
/*054*/ DWORD                SizeOfHeapCommit;
/*058*/ DWORD                LoaderFlags;
/*05C*/ DWORD                NumberOfRvaAndSizes;
/*060*/ IMAGE_DATA_DIRECTORY DataDirectory
                             [IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
/*0E0*/ }
        IMAGE_OPTIONAL_HEADER;
```

IO_COMPLETION

```
typedef struct _IO_COMPLETION
        {
/*000*/ KQUEUE Queue;
/*028*/ }
        IO_COMPLETION;
```

IO_COMPLETION_CONTEXT

```
typedef struct _IO_COMPLETION_CONTEXT
        {
/*000*/ PVOID Port;
/*004*/ PVOID Key;
/*008*/ }
        IO_COMPLETION_CONTEXT;
```

IO_ERROR_LOG_ENTRY

```
typedef struct _IO_ERROR_LOG_ENTRY
        {
/*000*/ SHORT               Type; // IO_TYPE_ERROR_LOG 0x0B
/*002*/ SHORT               Size; // number of BYTEs
/*004*/ LIST_ENTRY          ErrorLogList;
/*00C*/ PDEVICE_OBJECT      DeviceObject;
/*010*/ PDRIVER_OBJECT      DriverObject;
/*014*/ DWORD               Reserved;
/*018*/ LARGE_INTEGER       TimeStamp;
/*020*/ IO_ERROR_LOG_PACKET EntryData;
/*050*/ }
        IO_ERROR_LOG_ENTRY;
```

IO_ERROR_LOG_MESSAGE

```
typedef struct _IO_ERROR_LOG_MESSAGE
        {
/*000*/ WORD                Type; // IO_TYPE_ERROR_MESSAGE 0x0C
/*002*/ WORD                Size; // number of BYTEs
/*004*/ WORD                DriverNameLength;
/*008*/ LARGE_INTEGER       TimeStamp;
/*010*/ DWORD               DriverNameOffset;
/*018*/ IO_ERROR_LOG_PACKET EntryData;
/*048*/ }
        IO_ERROR_LOG_MESSAGE;
```

IO_ERROR_LOG_PACKET

```
typedef struct _IO_ERROR_LOG_PACKET
        {
/*000*/ BYTE          MajorFunctionCode;
/*001*/ BYTE          RetryCount;
/*002*/ WORD          DumpDataSize;
/*004*/ WORD          NumberOfStrings;
/*006*/ WORD          StringOffset;
/*008*/ WORD          EventCategory;
/*00C*/ NTSTATUS      ErrorCode;
/*010*/ DWORD         UniqueErrorValue;
/*014*/ NTSTATUS      FinalStatus;
/*018*/ DWORD         SequenceNumber;
/*01C*/ DWORD         IoControlCode;
/*020*/ LARGE_INTEGER DeviceOffset;
/*028*/ DWORD         DumpData [1];
/*030*/ }
        IO_ERROR_LOG_PACKET;
```

IO_STATUS_BLOCK

```
typedef struct _IO_STATUS_BLOCK
        {
/*000*/ NTSTATUS Status;
/*004*/ ULONG    Information;
/*008*/ }
        IO_STATUS_BLOCK;
```

IO_TIMER

```
typedef struct _IO_TIMER
        {
/*000*/ SHORT             Type;       // IO_TYPE_TIMER 0x09
/*002*/ WORD              TimerState; // 0 = stopped, 1 = started
/*004*/ LIST_ENTRY        TimerQueue;
/*00C*/ PIO_TIMER_ROUTINE TimerRoutine;
```

```
/*010*/ PVOID             Context;
/*014*/ PDEVICE_OBJECT    DeviceObject;
/*018*/ }
        IO_TIMER;
```

KAFFINITY

```
typedef DWORD KAFFINITY;
```

KAPC

```
typedef struct _KAPC
        {
/*000*/ SHORT             Type; // IO_TYPE_APC 0x12
/*002*/ SHORT             Size; // number of BYTEs
/*004*/ DWORD             Spare0;
/*008*/ struct _KTHREAD  *Thread;
/*00C*/ LIST_ENTRY        ApcListEntry;
/*014*/ PKKERNEL_ROUTINE  KernelRoutine;  // KiSuspendNop
/*018*/ PKRUNDOWN_ROUTINE RundownRoutine;
/*01C*/ PKNORMAL_ROUTINE  NormalRoutine;  // KiSuspendThread
/*020*/ PVOID             NormalContext;
/*024*/ PVOID             SystemArgument1;
/*028*/ PVOID             SystemArgument2;
/*02C*/ CHAR              ApcStateIndex;
/*02D*/ KPROCESSOR_MODE   ApcMode;
/*02E*/ BOOLEAN           Inserted;
/*030*/ }
        KAPC;
```

KAPC_STATE

```
typedef struct _KAPC_STATE
        {
/*000*/ LIST_ENTRY        ApcListHead [2];
/*010*/ struct _KPROCESS *Process;
/*014*/ BOOLEAN           KernelApcInProgress;
/*015*/ BOOLEAN           KernelApcPending;
/*016*/ BOOLEAN           UserApcPending;
/*018*/ }
        KAPC_STATE;
```

KDEVICE_QUEUE

```
typedef struct _KDEVICE_QUEUE
        {
/*000*/ SHORT      Type; // IO_TYPE_DEVICE_QUEUE 0x14
/*002*/ SHORT      Size; // number of BYTEs
```

```
/*004*/ LIST_ENTRY DeviceListHead;
/*00C*/ KSPIN_LOCK Lock;
/*010*/ BOOLEAN    Busy;
/*014*/ }
        KDEVICE_QUEUE;
```

KDEVICE_QUEUE_ENTRY

```
typedef struct _KDEVICE_QUEUE_ENTRY
        {
/*000*/ LIST_ENTRY DeviceListEntry;
/*008*/ DWORD      SortKey;
/*00C*/ BOOLEAN    Inserted;
/*010*/ }
        KDEVICE_QUEUE_ENTRY;
```

KDPC

```
typedef struct _KDPC
        {
/*000*/ SHORT              Type; // IO_TYPE_DPC 0x13
/*002*/ BYTE               Number;
/*003*/ BYTE               Importance;
/*004*/ LIST_ENTRY         DpcListEntry;
/*00C*/ PKDEFERRED_ROUTINE DeferredRoutine;
/*010*/ PVOID              DeferredContext;
/*014*/ PVOID              SystemArgument1;
/*018*/ PVOID              SystemArgument2;
/*01C*/ PDWORD_PTR         Lock;
/*020*/ }
        KDPC;
```

KEVENT

```
typedef struct _KEVENT
        {
/*000*/ DISPATCHER_HEADER Header; // DISP_TYPE_*_EVENT 0x00, 0x01
/*010*/ }
        KEVENT;
```

KEVENT_PAIR

```
typedef struct _KEVENT_PAIR
        {
/*000*/ SHORT  Type; // IO_TYPE_EVENT_PAIR 0x15
```

```
/*002*/ WORD   Size; // number of BYTEs
/*004*/ KEVENT Event1;
/*014*/ KEVENT Event2;
/*024*/ }
        KEVENT_PAIR;
```

KGDTENTRY

```
typedef struct _KGDTENTRY

        {
/*000*/ WORD  LimitLow;
/*002*/ WORD  BaseLow;
/*004*/ DWORD HighWord;
/*008*/ }
        KGDTENTRY;
```

KIDTENTRY

```
typedef struct _KIDTENTRY
        {
/*000*/ WORD Offset;
/*002*/ WORD Selector;
/*004*/ WORD Access;
/*006*/ WORD ExtendedOffset;
/*008*/ }
        KIDTENTRY;
```

KIRQL

```
typedef BYTE KIRQL;
```

KMUTANT, KMUTEX

```
typedef struct _KMUTANT
        {
/*000*/ DISPATCHER_HEADER Header; // DISP_TYPE_MUTANT 0x02
/*010*/ LIST_ENTRY        MutantListEntry;
/*018*/ struct _KTHREAD  *OwnerThread;
/*01C*/ BOOLEAN           Abandoned;
/*01D*/ BYTE              ApcDisable;
/*020*/ }
        KMUTANT, KMUTEX;
```

KPCR

```
// base address 0xFFDFF000


typedef struct _KPCR // processor control region
        {
/*000*/ NT_TIB             NtTib;
/*01C*/ struct _KPCR      *SelfPcr;
/*020*/ PKPRCB             Prcb;
/*024*/ KIRQL              Irql;
/*028*/ DWORD              IRR;
/*02C*/ DWORD              IrrActive;
/*030*/ DWORD              IDR;
/*034*/ DWORD              Reserved2;
/*038*/ struct _KIDTENTRY *IDT;
/*03C*/ struct _KGDTENTRY *GDT;
/*040*/ struct _KTSS      *TSS;
/*044*/ WORD               MajorVersion;
/*046*/ WORD               MinorVersion;
/*048*/ KAFFINITY          SetMember;
/*04C*/ DWORD              StallScaleFactor;
/*050*/ BYTE               DebugActive;
/*051*/ BYTE               Number;
/*054*/ }
        KPCR;
```

KPRCB

```
// base address 0xFFDFF120

typedef struct _KPRCB // processor control block
        {
/*000*/ WORD                    MinorVersion;
/*002*/ WORD                    MajorVersion;
/*004*/ struct _KTHREAD        *CurrentThread;
/*008*/ struct _KTHREAD        *NextThread;
/*00C*/ struct _KTHREAD        *IdleThread;
/*010*/ CHAR                    Number;
/*011*/ CHAR                    Reserved;
/*012*/ WORD                    BuildType;
/*014*/ KAFFINITY               SetMember;
/*018*/ struct _RESTART_BLOCK *RestartBlock;
/*01C*/ }
        KPRCB;
```

KPROCESS

```
typedef struct _KPROCESS
        {
/*000*/ DISPATCHER_HEADER Header; // DO_TYPE_PROCESS (0x1B)
/*010*/ LIST_ENTRY        ProfileListHead;
/*018*/ DWORD             DirectoryTableBase;
/*01C*/ DWORD             PageTableBase;
/*020*/ KGDTENTRY         LdtDescriptor;
/*028*/ KIDTENTRY         Int21Descriptor;
/*030*/ WORD              IopmOffset;
/*032*/ BYTE              Iopl;
/*033*/ BOOLEAN           VdmFlag;
/*034*/ DWORD             ActiveProcessors;
/*038*/ DWORD             KernelTime; // ticks
/*03C*/ DWORD             UserTime;   // ticks
/*040*/ LIST_ENTRY        ReadyListHead;
/*048*/ LIST_ENTRY        SwapListEntry;
/*050*/ LIST_ENTRY        ThreadListHead; // KTHREAD.ThreadListEntry
/*058*/ PVOID             ProcessLock;
/*05C*/ KAFFINITY         Affinity;
/*060*/ WORD              StackCount;
/*062*/ BYTE              BasePriority;
/*063*/ BYTE              ThreadQuantum;
/*064*/ BOOLEAN           AutoAlignment;
/*065*/ BYTE              State;
/*066*/ BYTE              ThreadSeed;
/*067*/ BOOLEAN           DisableBoost;
/*068*/ DWORD             d068;
/*06C*/ }
        KPROCESS;
```

KPROCESSOR_MODE

```
typedef CHAR KPROCESSOR_MODE;
```

KQUEUE

```
typedef struct _KQUEUE
        {
/*000*/ DISPATCHER_HEADER Header; // DISP_TYPE_QUEUE 0x04
/*010*/ LIST_ENTRY        EntryListHead;
/*018*/ DWORD             CurrentCount;
/*01C*/ DWORD             MaximumCount;
/*020*/ LIST_ENTRY        ThreadListHead;
/*028*/ }
        KQUEUE;
```

KSEMAPHORE

```
typedef struct _KSEMAPHORE
        {
/*000*/ DISPATCHER_HEADER Header; // DISP_TYPE_SEMAPHORE 0x05
/*010*/ LONG              Limit;
/*014*/ }
        KSEMAPHORE;
```

KTHREAD

```
typedef struct _KTHREAD
        {
/*000*/ DISPATCHER_HEADER         Header; // DO_TYPE_THREAD (0x6C)
/*010*/ LIST_ENTRY                MutantListHead;
/*018*/ PVOID                     InitialStack;
/*01C*/ PVOID                     StackLimit;
/*020*/ struct _TEB              *Teb;
/*024*/ PVOID                     TlsArray;
/*028*/ PVOID                     KernelStack;
/*02C*/ BOOLEAN                   DebugActive;
/*02D*/ BYTE                      State; // THREAD_STATE_*
/*02E*/ BOOLEAN                   Alerted;
/*02F*/ BYTE                      bReserved01;
/*030*/ BYTE                      Iopl;
/*031*/ BYTE                      NpxState;
/*032*/ BYTE                      Saturation;
/*033*/ BYTE                      Priority;
/*034*/ KAPC_STATE                ApcState;
/*04C*/ DWORD                     ContextSwitches;
/*050*/ DWORD                     WaitStatus;
/*054*/ BYTE                      WaitIrql;
/*055*/ BYTE                      WaitMode;
/*056*/ BYTE                      WaitNext;
/*057*/ BYTE                      WaitReason;
/*058*/ PLIST_ENTRY               WaitBlockList;
/*05C*/ LIST_ENTRY                WaitListEntry;
/*064*/ DWORD                     WaitTime;
/*068*/ BYTE                      BasePriority;
/*069*/ BYTE                      DecrementCount;
/*06A*/ BYTE                      PriorityDecrement;
/*06B*/ BYTE                      Quantum;
/*06C*/ KWAIT_BLOCK               WaitBlock [4];
/*0CC*/ DWORD                     LegoData;
/*0D0*/ DWORD                     KernelApcDisable;
/*0D4*/ KAFFINITY                 UserAffinity;
/*0D8*/ BOOLEAN                   SystemAffinityActive;
/*0D9*/ BYTE                      Pad [3];
/*0DC*/ PSERVICE_DESCRIPTOR_TABLE pServiceDescriptorTable;
```

```
/*0E0*/ PVOID                    Queue;
/*0E4*/ PVOID                    ApcQueueLock;
/*0E8*/ KTIMER                   Timer;
/*110*/ LIST_ENTRY               QueueListEntry;
/*118*/ KAFFINITY                Affinity;
/*11C*/ BOOLEAN                  Preempted;
/*11D*/ BOOLEAN                  ProcessReadyQueue;
/*11E*/ BOOLEAN                  KernelStackResident;
/*11F*/ BYTE                     NextProcessor;
/*120*/ PVOID                    CallbackStack;
/*124*/ struct _WIN32_THREAD     *Win32Thread;
/*128*/ PVOID                    TrapFrame;
/*12C*/ PKAPC_STATE              ApcStatePointer;
/*130*/ PVOID                    p130;
/*134*/ BOOLEAN                  EnableStackSwap;
/*135*/ BOOLEAN                  LargeStack;
/*136*/ BYTE                     ResourceIndex;
/*137*/ KPROCESSOR_MODE          PreviousMode;
/*138*/ DWORD                    KernelTime; // ticks
/*13C*/ DWORD                    UserTime;   // ticks
/*140*/ KAPC_STATE               SavedApcState;
/*157*/ BYTE                     bReserved02;
/*158*/ BOOLEAN                  Alertable;
/*159*/ BYTE                     ApcStateIndex;
/*15A*/ BOOLEAN                  ApcQueueable;
/*15B*/ BOOLEAN                  AutoAlignment;
/*15C*/ PVOID                    StackBase;
/*160*/ KAPC                     SuspendApc;
/*190*/ KSEMAPHORE               SuspendSemaphore;
/*1A4*/ LIST_ENTRY               ThreadListEntry;  // see KPROCESS
/*1AC*/ BYTE                     FreezeCount;
/*1AD*/ BYTE                     SuspendCount;
/*1AE*/ BYTE                     IdealProcessor;
/*1AF*/ BOOLEAN                  DisableBoost;
/*1B0*/ }
        KTHREAD;
```

KTIMER

```
typedef struct _KTIMER
        {
/*000*/ DISPATCHER_HEADER Header; // DISP_TYPE_*_TIMER 0x08, 0x09
/*010*/ ULARGE_INTEGER    DueTime;
/*018*/ LIST_ENTRY        TimerListEntry;
/*020*/ struct _KDPC     *Dpc;
/*024*/ LONG              Period;
/*028*/ }
        KTIMER;
```

KWAIT_BLOCK

```
typedef struct _KWAIT_BLOCK
        {
/*000*/ LIST_ENTRY           WaitListEntry;
/*008*/ struct _KTHREAD     *Thread;
/*00C*/ PVOID                Object;
/*010*/ struct _KWAIT_BLOCK *NextWaitBlock;
/*004*/ WORD                 WaitKey;
/*006*/ WORD                 WaitType;
/*018*/ }
        KWAIT_BLOCK;
```

LARGE_INTEGER

```
typedef union _LARGE_INTEGER
        {
/*000*/ struct
            {
/*000*/     ULONG LowPart;
/*004*/     LONG  HighPart;
/*008*/     };
/*000*/ LONGLONG  QuadPart;
/*008*/ }
        LARGE_INTEGER;
```

LIST_ENTRY

```
typedef struct _LIST_ENTRY
        {
/*000*/ struct _LIST_ENTRY *Flink;
/*004*/ struct _LIST_ENTRY *Blink;
/*008*/ }
        LIST_ENTRY;
```

MMSUPPORT

```
typedef struct _MMSUPPORT
        {
/*000*/ LARGE_INTEGER LastTrimTime;
/*008*/ DWORD         LastTrimFaultCount;
/*00C*/ DWORD         PageFaultCount;
/*010*/ DWORD         PeakWorkingSetSize;
/*014*/ DWORD         WorkingSetSize;
/*018*/ DWORD         MinimumWorkingSetSize;
```

```
/*01C*/ DWORD         MaximumWorkingSetSize;
/*020*/ PVOID         VmWorkingSetList;
/*024*/ LIST_ENTRY    WorkingSetExpansionLinks;
/*02C*/ BOOLEAN       AllowWorkingSetAdjustment;
/*02D*/ BOOLEAN       AddressSpaceBeingDeleted;
/*02E*/ BYTE          ForegroundSwitchCount;
/*02F*/ BYTE          MemoryPriority;
/*030*/ }
        MMSUPPORT;
```

NT_TIB (THREAD INFORMATION BLOCK)

```
typedef struct _NT_TIB // see winnt.h / ntddk.h
        {
/*000*/ struct _EXCEPTION_REGISTRATION_RECORD *ExceptionList;
/*004*/ PVOID                                  StackBase;
/*008*/ PVOID                                  StackLimit;
/*00C*/ PVOID                                  SubSystemTib;
/*010*/ union
            {
/*010*/     PVOID FiberData;
/*010*/     ULONG Version;
            };
/*014*/ PVOID           ArbitraryUserPointer;
/*018*/ struct _NT_TIB *Self;
/*01C*/ }
        NT_TIB;
```

NTSTATUS

```
typedef LONG NTSTATUS;
```

OBJECT_ATTRIBUTES

```
typedef struct _OBJECT_ATTRIBUTES
        {
/*000*/ DWORD                        Length; // 0x18
/*004*/ HANDLE                       RootDirectory;
/*008*/ PUNICODE_STRING              ObjectName;
/*00C*/ DWORD                        Attributes;
/*010*/ PSECURITY_DESCRIPTOR         SecurityDescriptor;
/*014*/ PSECURITY_QUALITY_OF_SERVICE SecurityQualityOfService;
/*018*/ }
        OBJECT_ATTRIBUTES;
```

OBJECT_CREATE_INFO

```
typedef struct _OBJECT_CREATE_INFO
        {
/*000*/ DWORD                          Attributes; // OBJ_*
/*004*/ HANDLE                         RootDirectory;
/*008*/ DWORD                          Reserved;
/*00C*/ KPROCESSOR_MODE                AccessMode;
/*010*/ DWORD                          PagedPoolCharge;
/*014*/ DWORD                          NonPagedPoolCharge;
/*018*/ DWORD                          SecurityCharge;
/*01C*/ PSECURITY_DESCRIPTOR           SecurityDescriptor;
/*020*/ PSECURITY_QUALITY_OF_SERVICE SecurityQualityOfService;
/*024*/ SECURITY_QUALITY_OF_SERVICE  SecurityQualityOfServiceBuffer;
/*030*/ }
        OBJECT_CREATE_INFO;
```

OBJECT_CREATOR_INFO

```
typedef struct _OBJECT_CREATOR_INFO
        {
/*000*/ LIST_ENTRY ObjectList;      // OBJECT_CREATOR_INFO
/*008*/ HANDLE     UniqueProcessId;
/*00C*/ WORD       Reserved1;
/*00E*/ WORD       Reserved2;
/*010*/ }
        OBJECT_CREATOR_INFO;
```

OBJECT_DIRECTORY

```
#define OBJECT_HASH_TABLE_SIZE 37

typedef struct _OBJECT_DIRECTORY
        {
/*000*/ POBJECT_DIRECTORY_ENTRY HashTable [OBJECT_HASH_TABLE_SIZE];
/*094*/ POBJECT_DIRECTORY_ENTRY CurrentEntry;
/*098*/ BOOLEAN                 CurrentEntryValid;
/*099*/ BYTE                    Reserved1;
/*09A*/ WORD                    Reserved2;
/*09C*/ DWORD                   Reserved3;
/*0A0*/ }
        OBJECT_DIRECTORY;
```

OBJECT_DIRECTORY_ENTRY

```
typedef struct _OBJECT_DIRECTORY_ENTRY
        {
/*000*/ struct _OBJECT_DIRECTORY_ENTRY *NextEntry;
```

```
/*004*/ POBJECT                         Object;
/*008*/ }
        OBJECT_DIRECTORY_ENTRY;
```

OBJECT_HANDLE_DB

```
typedef struct _OBJECT_HANDLE_DB
        {
/*000*/ union
            {
/*000*/     struct _EPROCESS              *Process;
/*000*/     struct _OBJECT_HANDLE_DB_LIST *HandleDBList;
/*004*/     };
/*004*/ DWORD HandleCount;
/*008*/ }
        OBJECT_HANDLE_DB;
```

OBJECT_HANDLE_DB_LIST

```
typedef struct _OBJECT_HANDLE_DB_LIST
        {
/*000*/ DWORD            Count;
/*004*/ OBJECT_HANDLE_DB Entries [];
/*???*/ }
        OBJECT_HANDLE_DB_LIST;
```

OBJECT_HANDLE_INFORMATION

```
#define OBJ_HANDLE_TAGBITS 0x00000003

typedef struct _OBJECT_HANDLE_INFORMATION // cf. HANDLE_ENTRY
        {
/*000*/ DWORD       HandleAttributes; // cf. HANDLE_ATTRIBUTE_MASK
/*004*/ ACCESS_MASK GrantedAccess;
/*008*/ }
        OBJECT_HANDLE_INFORMATION;
```

OBJECT_HEADER

```
typedef struct _OBJECT_HEADER
        {
/*000*/ DWORD        PointerCount;       // number of references
/*004*/ DWORD        HandleCount;        // number of open handles
/*008*/ POBJECT_TYPE ObjectType;
/*00C*/ BYTE         NameOffset;         // -> OBJECT_NAME
/*00D*/ BYTE         HandleDBOffset;     // -> OBJECT_HANDLE_DB
/*00E*/ BYTE         QuotaChargesOffset; // -> OBJECT_QUOTA_CHARGES
```

```
/*00F*/ BYTE         ObjectFlags;        // OB_FLAG_*
/*010*/ union
            { // OB_FLAG_CREATE_INFO ? ObjectCreateInfo : QuotaBlock
/*010*/     PQUOTA_BLOCK        QuotaBlock;
/*010*/     POBJECT_CREATE_INFO ObjectCreateInfo;
/*014*/     };
/*014*/ PSECURITY_DESCRIPTOR SecurityDescriptor;
/*018*/ }
        OBJECT_HEADER;
```

OBJECT_NAME

```
typedef struct _OBJECT_NAME
        {
/*000*/ POBJECT_DIRECTORY Directory;
/*004*/ UNICODE_STRING    Name;
/*00C*/ DWORD             Reserved;
/*010*/ }
        OBJECT_NAME;
```

OBJECT_NAME_INFORMATION

```
typedef struct _OBJECT_NAME_INFORMATION
        {
/*000*/ UNICODE_STRING Name; // points to Buffer[]
/*008*/ WORD           Buffer [];
/*???*/ }
        OBJECT_NAME_INFORMATION;
```

OBJECT_QUOTA_CHARGES

```
#define OB_SECURITY_CHARGE 0x00000800

typedef struct _OBJECT_QUOTA_CHARGES
        {
/*000*/ DWORD PagedPoolCharge;
/*004*/ DWORD NonPagedPoolCharge;
/*008*/ DWORD SecurityCharge;
/*00C*/ DWORD Reserved;
/*010*/ }
        OBJECT_QUOTA_CHARGES;
```

OBJECT_TYPE

```
typedef struct _OBJECT_TYPE
        {
/*000*/ ERESOURCE      Lock;
/*038*/ LIST_ENTRY     ObjectListHead; // OBJECT_CREATOR_INFO
```

```
/*040*/ UNICODE_STRING ObjectTypeName; // see above
/*048*/ union
            {
/*048*/      PVOID DefaultObject; // ObpDefaultObject
/*048*/      DWORD Code;          // File: 5C, WaitablePort: A0
            };
/*04C*/ DWORD                   ObjectTypeIndex; // OB_TYPE_INDEX_*
/*050*/ DWORD                   ObjectCount;
/*054*/ DWORD                   HandleCount;
/*058*/ DWORD                   PeakObjectCount;
/*05C*/ DWORD                   PeakHandleCount;
/*060*/ OBJECT_TYPE_INITIALIZER ObjectTypeInitializer;
/*0AC*/ DWORD                   ObjectTypeTag;   // OB_TYPE_TAG_*
/*0B0*/ }
        OBJECT_TYPE;
```

OBJECT_TYPE_ARRAY

```
typedef struct _OBJECT_TYPE_ARRAY
        {
/*000*/ DWORD                ObjectCount;
/*004*/ POBJECT_CREATOR_INFO ObjectList [];
/*???*/ }
        OBJECT_TYPE_ARRAY;
```

OBJECT_TYPE_INFO

```
typedef struct _OBJECT_TYPE_INFO
        {
/*000*/ UNICODE_STRING  ObjectTypeName; // points to Buffer[]
/*008*/ DWORD           ObjectCount;
/*00C*/ DWORD           HandleCount;
/*010*/ DWORD           Reserved1 [4];
/*020*/ DWORD           PeakObjectCount;
/*024*/ DWORD           PeakHandleCount;
/*028*/ DWORD           Reserved2 [4];
/*038*/ DWORD           InvalidAttributes;
/*03C*/ GENERIC_MAPPING GenericMapping;
/*04C*/ ACCESS_MASK     ValidAccessMask;
/*050*/ BOOLEAN         SecurityRequired;
/*051*/ BOOLEAN         MaintainHandleCount;
/*052*/ WORD            Reserved3;
/*054*/ BOOL            PagedPool;
/*058*/ DWORD           DefaultPagedPoolCharge;
/*05C*/ DWORD           DefaultNonPagedPoolCharge;
/*060*/ WORD            Buffer [];
/*???*/ }
        OBJECT_TYPE_INFO;
```

OBJECT_TYPE_INITIALIZER

```
typedef struct _OBJECT_TYPE_INITIALIZER
        {
/*000*/ WORD            Length;          //0x004C
/*002*/ BOOLEAN         UseDefaultObject;//OBJECT_TYPE.DefaultObject
/*003*/ BOOLEAN         Reserved1;
/*004*/ DWORD           InvalidAttributes;
/*008*/ GENERIC_MAPPING GenericMapping;
/*018*/ ACCESS_MASK     ValidAccessMask;
/*01C*/ BOOLEAN         SecurityRequired;
/*01D*/ BOOLEAN         MaintainHandleCount; // OBJECT_HANDLE_DB
/*01E*/ BOOLEAN         MaintainTypeList;    // OBJECT_CREATOR_INFO
/*01F*/ BYTE            Reserved2;
/*020*/ BOOL            PagedPool;
/*024*/ DWORD           DefaultPagedPoolCharge;
/*028*/ DWORD           DefaultNonPagedPoolCharge;
/*02C*/ NTPROC          DumpProcedure;
/*030*/ NTPROC          OpenProcedure;
/*034*/ NTPROC          CloseProcedure;
/*038*/ NTPROC          DeleteProcedure;
/*03C*/ NTPROC_VOID     ParseProcedure;
/*040*/ NTPROC_VOID     SecurityProcedure; // SeDefaultObjectMethod
/*044*/ NTPROC_VOID     QueryNameProcedure;
/*048*/ NTPROC_BOOLEAN  OkayToCloseProcedure;
/*04C*/ }
        OBJECT_TYPE_INITIALIZER;
```

OEM_STRING

```
typedef STRING OEM_STRING;
```

OWNER_ENTRY

```
typedef struct _OWNER_ENTRY
        {
/*000*/ ERESOURCE_THREAD OwnerThread;
/*004*/ union
            {
/*004*/     LONG  OwnerCount;
/*004*/     DWORD TableSize;
/*008*/     };
/*008*/ }
        OWNER_ENTRY;
```

PEB (PROCESS ENVIRONMENT BLOCK)

```
// located at 0x7FFDF000

typedef struct _PEB
        {
/*000*/ BOOLEAN              InheritedAddressSpace;
/*001*/ BOOLEAN              ReadImageFileExecOptions;
/*002*/ BOOLEAN              BeingDebugged;
/*003*/ BYTE                 b003;
/*004*/ DWORD                d004;
/*008*/ PVOID                SectionBaseAddress;
/*00C*/ PPROCESS_MODULE_INFO ProcessModuleInfo;
/*010*/ PPROCESS_PARAMETERS  ProcessParameters;
/*014*/ DWORD                SubSystemData;
/*018*/ HANDLE               ProcessHeap;
/*01C*/ PCRITICAL_SECTION    FastPebLock;
/*020*/ PVOID                AcquireFastPebLock; // function
/*024*/ PVOID                ReleaseFastPebLock; // function
/*028*/ DWORD                d028;
/*02C*/ PPVOID               User32Dispatch;     // function
/*030*/ DWORD                d030;
/*034*/ DWORD                d034;
/*038*/ DWORD                d038;
/*03C*/ DWORD                TlsBitMapSize;      // number of bits
/*040*/ PRTL_BITMAP          TlsBitMap;          // ntdll!TlsBitMap
/*044*/ DWORD                TlsBitMapData [2];  // 64 bits
/*04C*/ PVOID                p04C;
/*050*/ PVOID                p050;
/*054*/ PTEXT_INFO           TextInfo;
/*058*/ PVOID                InitAnsiCodePageData;
/*05C*/ PVOID                InitOemCodePageData;
/*060*/ PVOID                InitUnicodeCaseTableData;
/*064*/ DWORD                KeNumberProcessors;
/*068*/ DWORD                NtGlobalFlag;
/*06C*/ DWORD                d6C;
/*070*/ LARGE_INTEGER        MmCriticalSectionTimeout;
/*078*/ DWORD                MmHeapSegmentReserve;
/*07C*/ DWORD                MmHeapSegmentCommit;
/*080*/ DWORD                MmHeapDeCommitTotalFreeThreshold;
/*084*/ DWORD                MmHeapDeCommitFreeBlockThreshold;
/*088*/ DWORD                NumberOfHeaps;
/*08C*/ DWORD                AvailableHeaps; // 16, *2 if exhausted
/*090*/ PHANDLE              ProcessHeapsListBuffer;
/*094*/ DWORD                d094;
/*098*/ DWORD                d098;
/*09C*/ DWORD                d09C;
```

```
/*0A0*/ PCRITICAL_SECTION    LoaderLock;
/*0A4*/ DWORD                NtMajorVersion;
/*0A8*/ DWORD                NtMinorVersion;
/*0AC*/ WORD                 NtBuildNumber;
/*0AE*/ WORD                 CmNtCSDVersion;
/*0B0*/ DWORD                PlatformId;
/*0B4*/ DWORD                Subsystem;
/*0B8*/ DWORD                MajorSubsystemVersion;
/*0BC*/ DWORD                MinorSubsystemVersion;
/*0C0*/ KAFFINITY            AffinityMask;
/*0C4*/ DWORD                ad0C4 [35];
/*150*/ PVOID                p150;
/*154*/ DWORD                ad154 [32];
/*1D4*/ HANDLE               Win32WindowStation;
/*1D8*/ DWORD                d1D8;
/*1DC*/ DWORD                d1DC;
/*1E0*/ PWORD                CSDVersion;
/*1E4*/ DWORD                d1E4;
/*1E8*/ }
        PEB;
```

PHYSICAL_ADDRESS

```
typedef LARGE_INTEGER PHYSICAL_ADDRESS;
```

PROCESS_PARAMETERS

```
typedef struct _PROCESS_PARAMETERS
        {
/*000*/ DWORD          Allocated;
/*004*/ DWORD          Size;
/*008*/ DWORD          Flags; // bit 0: all pointers normalized
/*00C*/ DWORD          Reserved1;
/*010*/ LONG           Console;
/*014*/ DWORD          ProcessGroup;
/*018*/ HANDLE         StdInput;
/*01C*/ HANDLE         StdOutput;
/*020*/ HANDLE         StdError;
/*024*/ UNICODE_STRING WorkingDirectoryName;
/*02C*/ HANDLE         WorkingDirectoryHandle;
/*030*/ UNICODE_STRING SearchPath;
/*038*/ UNICODE_STRING ImagePath;
/*040*/ UNICODE_STRING CommandLine;
/*048*/ PWORD          Environment;
/*04C*/ DWORD          X;
/*050*/ DWORD          Y;
/*054*/ DWORD          XSize;
/*058*/ DWORD          YSize;
/*05C*/ DWORD          XCountChars;
```

```
/*060*/ DWORD          YCountChars;
/*064*/ DWORD          FillAttribute;
/*068*/ DWORD          Flags2;
/*06C*/ WORD           ShowWindow;
/*06E*/ WORD           Reserved2;
/*070*/ UNICODE_STRING Title;
/*078*/ UNICODE_STRING Desktop;
/*080*/ UNICODE_STRING Reserved3;
/*088*/ UNICODE_STRING Reserved4;
/*090*/ }
        PROCESS_PARAMETERS;
```

QUOTA_BLOCK

```
typedef struct _QUOTA_BLOCK
        {
/*000*/ DWORD Flags;
/*004*/ DWORD ChargeCount;
/*008*/ DWORD PeakPoolUsage [2]; // NonPagedPool, PagedPool
/*010*/ DWORD PoolUsage     [2]; // NonPagedPool, PagedPool
/*018*/ DWORD PoolQuota     [2]; // NonPagedPool, PagedPool
/*020*/ }
        QUOTA_BLOCK;
```

RTL_BITMAP

```
typedef struct _RTL_BITMAP
        {
/*000*/ DWORD  SizeOfBitMap;
/*004*/ PDWORD Buffer;
/*008*/ }
        RTL_BITMAP;
```

RTL_CRITICAL_SECTION_DEBUG

```
#define RTL_CRITSECT_TYPE 0
#define RTL_RESOURCE_TYPE 1

typedef struct _RTL_CRITICAL_SECTION_DEBUG
        {
/*000*/ WORD                          Type;
/*002*/ WORD                          CreatorBackTraceIndex;
/*004*/ struct _RTL_CRITICAL_SECTION *CriticalSection;
/*008*/ LIST_ENTRY                    ProcessLocksList;
/*010*/ DWORD                         EntryCount;
/*014*/ DWORD                         ContentionCount;
/*018*/ DWORD                         Spare [2];
/*020*/ }
        RTL_CRITICAL_SECTION_DEBUG;
```

SECTION_OBJECT_POINTERS

```
typedef struct _SECTION_OBJECT_POINTERS
        {
/*000*/ PVOID DataSectionObject;
/*004*/ PVOID SharedCacheMap;
/*008*/ PVOID ImageSectionObject;
/*00C*/ }
        SECTION_OBJECT_POINTERS;
```

SECURITY_DESCRIPTOR

```
typedef struct _SECURITY_DESCRIPTOR
        {
/*000*/ BYTE                        Revision;
/*001*/ BYTE                        Sbz1;
/*002*/ SECURITY_DESCRIPTOR_CONTROL Control;
/*004*/ PSID                        Owner;
/*008*/ PSID                        Group;
/*00C*/ PACL                        Sacl;
/*010*/ PACL                        Dacl;
/*014*/ }
         SECURITY_DESCRIPTOR;
```

SECURITY_DESCRIPTOR_CONTROL

```
typedef WORD SECURITY_DESCRIPTOR_CONTROL;
```

SERVICE_DESCRIPTOR_TABLE

```
typedef struct _SERVICE_DESCRIPTOR_TABLE
        {
/*000*/ SYSTEM_SERVICE_TABLE ntoskrnl;  // ntoskrnl.exe (native api)
/*010*/ SYSTEM_SERVICE_TABLE win32k;    // win32k.sys   (gdi/user)
/*020*/ SYSTEM_SERVICE_TABLE Table3;    // not used
/*030*/ SYSTEM_SERVICE_TABLE Table4;    // not used
/*040*/ }
        SERVICE_DESCRIPTOR_TABLE;
```

STRING

```
typedef struct _STRING
        {
/*000*/ WORD  Length;
/*002*/ WORD  MaximumLength;
```

```
/*004*/ PBYTE Buffer;
/*008*/ }
        STRING;
```

SYSTEM_SERVICE_TABLE

```
typedef NTSTATUS (NTAPI *NTPROC) ();
typedef NTPROC *PNTPROC;

typedef struct _SYSTEM_SERVICE_TABLE
        {
/*000*/ PNTPROC ServiceTable;          // array of entry points
/*004*/ PDWORD  CounterTable;          // array of usage counters
/*008*/ DWORD   ServiceLimit;          // number of table entries
/*00C*/ PBYTE   ArgumentTable;         // array of byte counts
/*010*/ }
        SYSTEM_SERVICE_TABLE;
```

TEB (THREAD ENVIRONMENT BLOCK)

```
// located at 0x7FFDE000, 0x7FFDD000, ...

typedef struct _TEB
        {
/*000*/ NT_TIB    Tib;
/*01C*/ PVOID     EnvironmentPointer;
/*020*/ CLIENT_ID Cid;
/*028*/ HANDLE    RpcHandle;
/*02C*/ PPVOID    ThreadLocalStorage;
/*030*/ PPEB      Peb;
/*034*/ DWORD     LastErrorValue;
/*038*/ }
        TEB;
```

TIME_FIELDS

```
typedef struct _TIME_FIELDS
        {
/*000*/ SHORT Year;
/*002*/ SHORT Month;
/*004*/ SHORT Day;
/*006*/ SHORT Hour;
/*008*/ SHORT Minute;
/*00A*/ SHORT Second;
/*00C*/ SHORT Milliseconds;
/*00E*/ SHORT Weekday; // 0 = sunday
/*010*/ }
        TIME_FIELDS;
```

ULARGE_INTEGER

```
typedef union _ULARGE_INTEGER
        {
/*000*/ struct
            {
/*000*/     ULONG LowPart;
/*004*/     ULONG HighPart;
/*008*/     };
/*000*/ ULONGLONG QuadPart;
/*008*/ }
        ULARGE_INTEGER;
```

UNICODE_STRING

```
typedef struct _UNICODE_STRING
        {
/*000*/ WORD  Length;
/*002*/ WORD  MaximumLength;
/*004*/ PWORD Buffer;
/*008*/ }
        UNICODE_STRING;
```

VPB (VOLUME PARAMETER BLOCK)

```
#define MAXIMUM_VOLUME_LABEL        32
#define MAXIMUM_VOLUME_LABEL_LENGTH (MAXIMUM_VOLUME_LABEL * WORD_)

typedef struct _VPB // volume parameter block
        {
/*000*/ SHORT                  Type; // IO_TYPE_VPB 0x0A
/*002*/ SHORT                  Size; // number of BYTEs
/*004*/ WORD                   Flags;
/*006*/ WORD                   VolumeLabelLength; // bytes (no term.)
/*008*/ struct _DEVICE_OBJECT *DeviceObject;
/*00C*/ struct _DEVICE_OBJECT *RealDevice;
/*010*/ DWORD                  SerialNumber;
/*014*/ DWORD                  ReferenceCount;
/*018*/ WORD                   VolumeLabel [MAXIMUM_VOLUME_LABEL];
/*058*/ }
        VPB;
```

WAIT_CONTEXT_BLOCK

```
typedef struct _WAIT_CONTEXT_BLOCK
        {
/*000*/ KDEVICE_QUEUE_ENTRY WaitQueueEntry;
/*010*/ PDRIVER_CONTROL     DeviceRoutine;
/*014*/ PVOID               DeviceContext;
/*018*/ DWORD               NumberOfMapRegisters;
/*01C*/ PVOID               DeviceObject;
/*020*/ PVOID               CurrentIrp;
/*024*/ PKDPC               BufferChainingDpc;
/*028*/ }
        WAIT_CONTEXT_BLOCK;
```