

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Windows 2000/NT Native API. Leksykon

Autor: Gary Nebbett

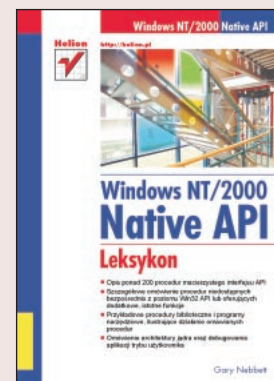
Tłumaczenie: Paweł Koronkiewicz

ISBN: 83-7197-508-2

Tytuł oryginału: [Windows NT/2000 Native API](#)

Reference

Format: B5, stron: 632



Native API, czyli macierzysty interfejs programowania aplikacji systemu Windows NT, to zestaw usług systemowych dostarczanych przez moduł wykonawczy NT programom trybu użytkownika i trybu jądra. Leksykon Windows NT/2000 Native API jest próbą całościowego omówienia tego nieudokumentowanego zbioru procedur. Stanowi niezastąpione narzędzie projektanta oprogramowania, zawierając:

- opis ponad 200 procedur macierzystego interfejsu API;
- szczegółowe omówienie procedur niedostępnych bezpośrednio z poziomu Win32 API lub oferujących dodatkowe, istotne funkcje;
- przykładowe procedury biblioteczne i programy narzędziowe, ilustrujące działanie omawianych procedur;
- omówienie architektury jądra oraz debugowania aplikacji trybu użytkownika.

Jako programiście pracującym na poziomie systemu, a także aplikacji, Leksykon Windows NT/2000 Native API pomoże Ci w:

- tworzeniu programów narzędziowych i mechanizmów systemowych, takich jak debugery, narzędzia analityczne i biblioteki wykonawcze;
- określeniu czy funkcje, których brakuje w systemie, choć należałoby oczekiwać ich obecności, faktycznie nie istnieją, czy jedynie nie zostały udokumentowane;
- zapoznaniu się ze zmianami w API wprowadzonymi w Windows 2000;
- pogłębieniu wiedzy o tajnikach funkcjonowania Windows NT.



Spis treści

<i>O Autorze</i>	19
<i>Wstęp</i>	21
Korzystanie z macierzystego interfejsu API	22
Zależność między Win32 API a API macierzystym	23
Przykład W.1. Typowe osłonięcie procedury API macierzystego przez interfejs Win32	23
Przykład W.2. Najprostszy przykład osłonięcia przez Win32 procedury API macierzystego	25
Wywoływanie macierzystych usług systemu z trybu jądra.....	25
Przykład W.3. Typowa preambuła macierzystej usługi systemowej	26
Wartości zwracane przez usługi systemowe	27
Względna częstość wywołań usług systemowych.....	27
Tabela W.1. Częstość wywołań macierzystych usług systemowych	27
O niniejszej książce	31
 <i>Rozdział 1. Informacje systemowe</i>	 33
ZwQuerySystemInformation	33
ZwSetSystemInformation	35
SYSTEM_INFORMATION_CLASS	36
SystemBasicInformation.....	37
SystemProcessorInformation	38
SystemPerformanceInformation	39
SystemTimeOfDayInformation	47
SystemProcessesAndThreadsInformation	48
SystemCallCounts.....	53
SystemConfigurationInformation	53
SystemProcessorTimes	54
SystemGlobalFlag.....	55

SystemModuleInformation	56
SystemLockInformation	57
SystemHandleInformation	59
SystemObjectInformation	60
SystemPagefileInformation	63
SystemInstructionEmulationCounts	64
SystemCacheInformation	64
SystemPoolTagInformation	66
SystemProcessorStatistics	67
SystemDpcInformation	68
SystemLoadImage	68
SystemUnloadImage	69
SystemTimeAdjustment	70
SystemCrashDumpInformation	71
SystemExceptionInformation	71
SystemCrashDumpStateInformation	72
SystemKernelDebuggerInformation	73
SystemContextSwitchInformation	73
SystemRegistryQuotaInformation	74
SystemLoadAndCallImage	74
SystemPrioritySeparation	75
SystemTimeZoneInformation	76
SystemLookasideInformation	77
SystemSetTimeSlipEvent	78
SystemCreateSession	79
SystemDeleteSession	79
SystemRangeStartInformation	80
SystemVerifierInformation	80
SystemAddVerifier	80
SystemSessionProcessesInformation	81
SystemPoolBlocksInformation	81
SystemMemoryUsageInformation	83
Przykład 1.1. Częściowa implementacja biblioteki ToolHelp	84
Przykład 1.2. Lista otwartych uchwytów procesu	88
ZwQuerySystemEnvironmentValue	90
ZwSetSystemEnvironmentValue	92
ZwShutdownSystem	93
ZwSystemDebugControl	94
Przykład 1.3. Ustawianie wewnętrznego punktu przerwania	98
Przykład 1.4. Pobieranie danych śledzenia	100

Rozdział 2. Obiekty, katalogi i łącza symboliczne	103
OBJECT_ATTRIBUTES	103
ZwQueryObject	105
ZwSetInformationObject	107
OBJECT_INFORMATION_CLASS	108
ObjectBasicInformation.....	108
ObjectNameInformation	110
ObjectTypeInformation	110
ObjectAllTypesInformation.....	112
ObjectHandleInformation	113
ZwDuplicateObject.....	113
ZwMakeTemporaryObject	115
ZwClose	115
Przykład 2.1. Lista otwartych uchwytów procesu	116
ZwQuerySecurityObject	118
ZwSetSecurityObject	119
ZwCreateDirectoryObject.....	120
ZwOpenDirectoryObject	121
ZwQueryDirectoryObject	123
ZwCreateSymbolicLinkObject	124
ZwOpenSymbolicLinkObject.....	125
ZwQuerySymbolicLinkObject	126
Rozdział 3. Pamięć wirtualna	129
ZwAllocateVirtualMemory	129
ZwFreeVirtualMemory	131
ZwQueryVirtualMemory	132
MEMORY_INFORMATION_CLASS.....	134
MemoryBasicInformation.....	134
MemoryWorkingSetList	135
MemorySectionName	136
ZwLockVirtualMemory.....	137
ZwUnlockVirtualMemory	138
ZwReadVirtualMemory.....	139
ZwWriteVirtualMemory	140
ZwProtectVirtualMemory.....	142
ZwFlushVirtualMemory	143
ZwAllocateUserPhysicalPages	144
ZwFreeUserPhysicalPages	145
ZwMapUserPhysicalPages	146
ZwMapUserPhysicalPagesScatter	148

ZwGetWriteWatch.....	149
ZwResetWriteWatch.....	150
Rozdział 4. Sekcje	153
ZwCreateSection.....	153
ZwOpenSection	155
ZwQuerySection	156
SECTION_INFORMATION_CLASS	158
SectionBasicInformation	158
SectionImageInformation	159
ZwExtendSection.....	160
ZwMapViewOfSection.....	161
ZwUnmapViewOfSection	164
ZwAreMappedFilesTheSame	164
Rozdział 5. Wątki	167
ZwCreateThread	167
ZwOpenThread	170
ZwTerminateThread	171
ZwQueryInformationThread.....	172
ZwSetInformationThread	174
THREADINFOCLASS	175
ThreadBasicInformation	175
ThreadTimes	176
ThreadPriority	177
ThreadBasePriority	177
ThreadAffinityMask	177
ThreadImpersonationToken.....	177
ThreadEnableAlignmentFaultFixup	177
ThreadEventPair	178
ThreadQuerySetWin32StartAddress	178
ThreadZeroTlsCell.....	179
ThreadPerformanceCount.....	179
ThreadAmILastThread	179
ThreadIdealProcessor	179
ThreadPriorityBoost	179
ThreadSetTlsArrayAddress	179
ThreadIsIoPending.....	180
ThreadHideFromDebugger.....	180
ZwSuspendThread	180
ZwResumeThread.....	181

ZwGetContextThread	182
ZwSetContextThread	183
ZwQueueApcThread.....	184
ZwTestAlert	185
ZwAlertThread.....	186
ZwAlertResumeThread.....	187
ZwRegisterThreadTerminatePort	188
ZwImpersonateThread	188
ZwImpersonateAnonymousToken	189
Rozdział 6. Procesy	191
ZwCreateProcess	191
ZwOpenProcess	193
ZwTerminateProcess	195
ZwQueryInformationProcess.....	196
ZwSetInformationProcess.....	197
PROCESSINFOCLASS	198
ProcessBasicInformation	199
ProcessQuotaLimits	200
ProcessIoCounters	201
ProcessVmCounters.....	202
ProcessTimes	203
ProcessBasePriority	204
ProcessRaisePriority	204
ProcessDebugPort.....	205
ProcessExceptionPort	205
ProcessAccessToken.....	205
ProcessDefaultHardErrorMode	206
ProcessPooledUsageAndLimits.....	206
ProcessWorkingSetWatch	207
ProcessUserModeIOPL	208
ProcessEnableAlignmentFaultFixup	208
ProcessPriorityClass	208
ProcessWx86Information	209
ProcessHandleCount.....	209
ProcessAffinityMask	209
ProcessPriorityBoost.....	210
ProcessDeviceMap.....	210
ProcessSessionInformation.....	211
ProcessForegroundInformation	211
ProcessWow64Information	211

RtlCreateProcessParameters	212
RtlDestroyProcessParameters	214
PROCESS_PARAMETERS	214
RtlCreateQueryDebugBuffer	218
RtlQueryProcessDebugInformation	218
RtlDestroyQueryDebugBuffer	220
DEBUG_BUFFER	220
DEBUG_MODULE_INFORMATION	221
DEBUG_HEAP_INFORMATION	223
DEBUG_LOCK_INFORMATION	224
Przykład 6.1. Rozwidlenie procesu Win32	226
Przykład 6.2. Tworzenie procesu Win32	230
Przykład 6.3. Rozszerzenie implementacji biblioteki ToolHelp z użyciem RtlQueryProcessDebugInformation	234
Rozdział 7. Zadania (jobs)	241
ZwCreateJobObject	241
ZwOpenJobObject	242
ZwTerminateJobObject	243
ZwAssignProcessToJobObject	244
ZwQueryInformationJobObject	245
ZwSetInformationJobObject	246
JOBOBJECTINFOCLASS	247
JobObjectBasicAccountingInformation	247
JobObjectBasicLimitInformation	249
JobObjectBasicProcessIdList	251
JobObjectBasicUIRestrictions	252
JobObjectSecurityLimitInformation	252
JobObjectEndOfJobTimeInformation	254
JobObjectAssociateCompletionPortInformation	254
JobObjectBasicAndIoAccountingInformation	255
JobObjectExtendedLimitInformation	255
Rozdział 8. Żetony	257
ZwCreateToken	257
ZwOpenProcessToken	260
ZwOpenThreadToken	261
ZwDuplicateToken	262
ZwFilterToken	264
ZwAdjustPrivilegesToken	265
ZwAdjustGroupsToken	267

ZwQueryInformationToken.....	268
ZwSetInformationToken.....	269
TOKEN_INFORMATION_CLASS.....	270
TokenUser.....	271
TokenGroups i TokenRestrictedSids.....	271
TokenPrivileges.....	272
TokenOwner.....	272
TokenPrimaryGroup.....	273
TokenDefaultDacl.....	273
TokenSource.....	273
TokenType.....	274
TokenImpersonationLevel.....	274
TokenStatistics.....	274
TokenSessionId.....	276
Przykład 8.1. Tworzenie okna poleceń dla użytkownika SYSTEM.....	276
Rozdział 9. Synchronizacja.....	279
ZwWaitForSingleObject.....	279
ZwSignalAndWaitForSingleObject.....	280
ZwWaitForMultipleObjects.....	281
ZwCreateTimer.....	283
ZwOpenTimer.....	284
ZwCancelTimer.....	285
ZwSetTimer.....	286
ZwQueryTimer.....	287
TIMER_INFORMATION_CLASS.....	289
TimerBasicInformation.....	289
ZwCreateEvent.....	289
ZwOpenEvent.....	291
ZwSetEvent.....	292
ZwPulseEvent.....	292
ZwResetEvent.....	293
ZwClearEvent.....	294
ZwQueryEvent.....	295
EVENT_INFORMATION_CLASS.....	296
EventBasicInformation.....	296
ZwCreateSemaphore.....	297
ZwOpenSemaphore.....	298
ZwReleaseSemaphore.....	299
ZwQuerySemaphore.....	300
SEMAPHORE_INFORMATION_CLASS.....	301

SemaphoreBasicInformation	301
ZwCreateMutant	302
ZwOpenMutant	303
ZwReleaseMutant	304
ZwQueryMutant.....	305
MUTANT_INFORMATION_CLASS.....	306
MutantBasicInformation.....	306
ZwCreateIoCompletion	307
ZwOpenIoCompletion	308
ZwSetIoCompletion.....	309
ZwRemoveIoCompletion	310
ZwQueryIoCompletion.....	312
IO_COMPLETION_INFORMATION_CLASS.....	313
IoCompletionBasicInformation	313
ZwCreateEventPair.....	313
ZwOpenEventPair.....	314
ZwWaitLowEventPair	315
ZwWaitHighEventPair	316
ZwSetLowWaitHighEventPair	317
ZwSetHighWaitLowEventPair	318
ZwSetLowEventPair.....	318
ZwSetHighEventPair	319
Rozdział 10. Data i godzina	321
ZwQuerySystemTime	321
ZwSetSystemTime.....	322
ZwQueryPerformanceCounter.....	323
ZwSetTimerResolution.....	323
ZwQueryTimerResolution	324
ZwDelayExecution	325
ZwYieldExecution	326
ZwGetTickCount	327
Rozdział 11. Profilowanie wykonania.....	329
KPROFILE_SOURCE	329
ZwCreateProfile.....	329
ZwSetIntervalProfile.....	331
ZwQueryIntervalProfile.....	332
ZwStartProfile.....	333
ZwStopProfile	333
Przykład 11.1. Profilowanie jądra	334

Rozdział 12. Porty (LPC)	337
PORT_MESSAGE.....	337
PORT_SECTION_WRITE.....	339
PORT_SECTION_READ	340
ZwCreatePort	341
ZwCreateWaitablePort	342
ZwConnectPort	343
ZwSecureConnectPort	345
ZwListenPort	346
ZwAcceptConnectPort.....	347
ZwCompleteConnectPort	349
ZwRequestPort.....	349
ZwRequestWaitReplyPort	350
ZwReplyPort.....	351
ZwReplyWaitReplyPort	352
ZwReplyWaitReceivePort	353
ZwReplyWaitReceivePortEx.....	354
ZwReadRequestData	355
ZwWriteRequestData	357
ZwQueryInformationPort	358
PORT_INFORMATION_CLASS.....	359
PortBasicInformation.....	359
ZwImpersonateClientOfPort.....	360
Przykład 12.1. Przyłączenie do portu nazwanego	360
Rozdział 13. Pliki	365
ZwCreateFile	365
ZwOpenFile	368
ZwDeleteFile	371
ZwFlushBuffersFile.....	371
ZwCancelIoFile	372
ZwReadFile.....	373
ZwWriteFile.....	375
ZwReadFileScatter.....	376
ZwWriteFileGather	378
ZwLockFile.....	380
ZwUnlockFile	382
ZwDeviceIoControlFile.....	383
ZwFsControlFile.....	385
ZwNotifyChangeDirectoryFile.....	387
FILE_NOTIFY_INFORMATION	389

ZwQueryEaFile.....	389
ZwSetEaFile.....	391
FILE_FULL_EA_INFORMATION.....	392
FILE_GET_EA_INFORMATION.....	393
ZwCreateNamedPipeFile.....	394
ZwCreateMailslotFile.....	396
ZwQueryVolumeInformationFile.....	398
ZwSetVolumeInformationFile.....	399
FS_INFORMATION_CLASS.....	400
FileFsVolumeInformation.....	401
FileFsLabelInformation.....	402
FileFsSizeInformation.....	402
FileFsDeviceInformation.....	403
FileFsAttributeInformation.....	404
FileFsControlInformation.....	405
FileFsFullSizeInformation.....	405
FileFsObjectIdInformation.....	406
ZwQueryQuotaInformationFile.....	407
ZwSetQuotaInformationFile.....	408
FILE_USER_QUOTA_INFORMATION.....	410
FILE_QUOTA_LIST_INFORMATION.....	411
ZwQueryAttributesFile.....	411
ZwQueryFullAttributesFile.....	412
ZwQueryInformationFile.....	413
ZwSetInformationFile.....	414
ZwQueryDirectoryFile.....	415
ZwQueryOleDirectoryFile.....	417
FILE_INFORMATION_CLASS.....	419
FileDirectoryInformation.....	420
FileFullDirectoryInformation.....	422
FileBothDirectoryInformation.....	424
FileBasicInformation.....	426
FileStandardInformation.....	427
FileInternalInformation.....	428
FileEaInformation.....	428
FileAccessInformation.....	429
FileNameInformation.....	429
FileRenameInformation i FileLinkInformation.....	430
FileNamesInformation.....	430
FileDispositionInformation.....	431
FilePositionInformation.....	432
FileModeInformation.....	432

FileAlignmentInformation	432
FileAllInformation	433
FileAllocationInformation	433
FileEndOfFileInformation	434
FileStreamInformation.....	434
FilePipeInformation.....	435
FilePipeLocalInformation.....	436
FilePipeRemoteInformation	437
FileMailslotQueryInformation.....	437
FileMailslotSetInformation.....	438
FileCompressionInformation.....	439
FileObjectIdInformation	440
FileCompletionInformation	440
FileMoveClusterInformation	440
FileQuotaInformation	440
FileReparsePointInformation.....	441
FileNetworkOpenInformation	441
FileAttributeTagInformation	442
Przykład 13.1. Otwieranie pliku określonego identyfikatorem.....	443
Rozdział 14. Klucze Rejestru.....	445
ZwCreateKey	445
ZwOpenKey.....	447
ZwDeleteKey	448
ZwFlushKey.....	449
ZwSaveKey.....	449
ZwSaveMergedKeys.....	450
ZwRestoreKey	451
ZwLoadKey	452
ZwLoadKey2	453
ZwUnloadKey.....	454
ZwQueryOpenSubKeys.....	455
ZwReplaceKey.....	456
ZwSetInformationKey	457
KEY_SET_INFORMATION_CLASS.....	458
KeyWriteTimeInformation	458
ZwQueryKey	459
ZwEnumerateKey	460
KEY_INFORMATION_CLASS.....	461
KeyBasicInformation.....	461
KeyNodeInformation	462

KeyFullInformation	463
KeyNameInformation	464
ZwNotifyChangeKey	465
ZwNotifyChangeMultipleKeys	467
ZwDeleteValueKey	469
ZwSetValueKey	470
ZwQueryValueKey	471
ZwEnumerateValueKey	473
KEY_VALUE_INFORMATION_CLASS	474
KeyValueBasicInformation	474
KeyValueFullInformation i KeyValueFullInformationAlign64	475
KeyValuePartialInformation	476
ZwQueryMultipleValueKey	477
KEY_VALUE_ENTRY	479
ZwInitializeRegistry	479
Rozdział 15. Zabezpieczenia i inspekcja (auditing)	481
ZwPrivilegeCheck	481
ZwPrivilegeObjectAuditAlarm	482
ZwPrivilegedServiceAuditAlarm	483
ZwAccessCheck	485
ZwAccessCheckAndAuditAlarm	486
ZwAccessCheckByType	488
ZwAccessCheckByTypeAndAuditAlarm	490
ZwAccessCheckByTypeResultList	493
ZwAccessCheckByTypeResultListAndAuditAlarm	495
ZwAccessCheckByTypeResultListAndAuditAlarmByHandle	497
ZwOpenObjectAuditAlarm	500
ZwCloseObjectAuditAlarm	502
ZwDeleteObjectAuditAlarm	503
Rozdział 16. Plug and Play i zarządzanie energią	505
ZwRequestWakeupLatency	505
ZwRequestDeviceWakeup	506
ZwCancelDeviceWakeupRequest	507
ZwIsSystemResumeAutomatic	508
ZwSetThreadExecutionState	508
ZwGetDevicePowerState	509
ZwSetSystemPowerState	510
ZwInitiatePowerAction	512
ZwPowerInformation	513

POWER_INFORMATION_LEVEL.....	515
SystemPowerPolicyAc, SystemPowerPolicyDc, SystemPowerPolicyCurrent.....	515
SystemPowerCapabilities	516
SystemBatteryState.....	516
SystemPowerStateHandler	516
ProcessorStateHandler.....	517
AdministratorPowerPolicy	517
ProcessorInformation.....	517
SystemPowerInformation	517
ZwPlugPlayControl	518
ZwGetPlugPlayEvent	519
Rozdział 17. Inne usługi systemowe	521
ZwRaiseException	521
ZwContinue	522
ZwW32Call.....	523
ZwCallbackReturn	524
ZwSetLowWaitHighThread	526
ZwSetHighWaitLowThread	527
ZwLoadDriver	527
ZwUnloadDriver.....	528
ZwFlushInstructionCache.....	529
ZwFlushWriteBuffer.....	530
ZwQueryDefaultLocale	530
ZwSetDefaultLocale	531
ZwQueryDefaultUILanguage	532
ZwSetDefaultUILanguage.....	533
ZwQueryInstallUILanguage.....	533
ZwAllocateLocallyUniqueId	534
ZwAllocateUuids	535
ZwSetUuidSeed	536
ZwRaiseHardError.....	537
ZwSetDefaultHardErrorPort.....	538
ZwDisplayString.....	539
ZwCreatePagingFile	540
ZwAddAtom	541
ZwFindAtom.....	542
ZwDeleteAtom.....	543
ZwQueryInformationAtom.....	543
ATOM_INFORMATION_CLASS	545
AtomBasicInformation	545

AtomListInformation	545
ZwSetLdtEntries	546
ZwVdmControl	547
Nie zaimplementowane usługi systemowe	548
<i>Dodatek A Wywoływanie usług systemowych z trybu jądra</i>	549
Przykład A.1. Implementacja NtQueryEvent dla trybu jądra	555
Przykład A.2. Dynamiczne wiązanie z ntdll.dll	556
<i>Dodatek B Punkty wejścia trybu jądra na platformie Intel</i>	559
KiTrap03	559
KiTrap04	560
KiGetTickCount	560
KiCallbackReturn	560
KiSetLowWaitHighThread	560
KiDebugService	560
KiSystemService	561
<i>Dodatek C Wyjątki i debuggowanie kodu</i>	563
Przykład C.1. Pseudokod procedury KiDispatchException	563
Przykład C.2. Pseudokod procedury KiUserExceptionDispatcher	566
Debugger trybu jądra	566
Przykład C.3. Pseudokod procedury DebugService	567
Debuggery trybu jądra	568
DEBUG_MESSAGE	568
Przekazywanie komunikatów debuggowania	569
Konsekwencje mechanizmu przekazywania komunikatów	569
OutputDebugString	570
Śledzenie wywołań procedur eksportowanych bibliotek DLL	570
Przykład C.4. Narzędzie śledzenia	570
<i>Dodatek D Struktura dysku NTFS</i>	581
NTFS_RECORD_HEADER	581
FILE_RECORD_HEADER	582
ATTRIBUTE	584
RESIDENT_ATTRIBUTE	585
NONRESIDENT_ATTRIBUTE	586
AttributeStandardInformation	587
AttributeAttributeList	589
AttributeFileName	590

AttributeObjectId	592
AttributeSecurityDescriptor	593
AttributeVolumeName	593
AttributeVolumeInformation	593
AttributeData	594
AttributeIndexRoot	594
AttributeIndexAllocation	595
DIRECTORY_INDEX	596
DIRECTORY_ENTRY	596
AttributeBitmap	597
AttributeReparsePoint	598
AttributeEaInformation	598
AttributeEA	599
AttributePropertySet	600
AttributeLoggedUtilityStream	600
Pliki specjalne	600
Otwieranie plików specjalnych	602
Przywracanie danych plików usuniętych	603
Przykład D.1. Odzyskiwanie danych z pliku	603
Przykład D.2. Dekompresowanie odzyskanych danych	609
Skorowidz	611

3

Pamięć wirtualna

Usługi systemowe, które opisujemy w tym rozdziale, wykonują operacje dotyczące pamięci wirtualnej.

ZwAllocateVirtualMemory

Alokuje pamięć wirtualną w zakresie adresowym trybu użytkownika.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwAllocateVirtualMemory(
    IN HANDLE ProcessHandle,
    IN OUT PVOID *BaseAddress,
    IN ULONG ZeroBits,
    IN OUT PULONG AllocationSize,
    IN ULONG AllocationType,
    IN ULONG Protect
);
```

Parametry

ProcessHandle

Uchwyt obiektu typu proces, reprezentujący proces, dla którego zaalokowana zostanie pamięć. Wymagany jest dostęp `PROCESS_VM_OPERATION`.

BaseAddress

Wskaźnik do zmiennej, w której umieszczony zostanie adres bazowy zaalokowanego obszaru pamięci. Jeżeli przy wywołaniu parametr ten nie jest pusty, początkiem alokacji będzie wskazany adres, zaokrąglony (gdy jest taka potrzeba) do najbliższego progu ziarnistości alokacji.

ZeroBits

Określa liczbę starszych bitów adresu, które muszą pozostać równe zeru w adresie bazowym alokowanej pamięci. Wartość tego parametru musi być mniejsza od 21. Stosowana wyłącznie, gdy system operacyjny określa adres alokacji (parametr `BaseAddress` pozostaje pusty)

AllocationSize

Wskaźnik do zmiennej określającej rozmiar alokacji w bajtach. Po zakończeniu procedury — wielkość obszaru faktycznie zaalokowanego. Jeżeli wartość `BaseAddress` pozostała pusta, rozmiar alokacji może zostać powiększony odpowiednio do wielkości strony. W pozostałych przypadkach wartość `AllocationSize` obejmuje wszystkie strony, w których znajduje się jeden lub więcej bajtów z zakresu od `BaseAddress` do `BaseAddress+AllocationSize`.

AllocationType

Zestaw znaczników opisujących wykonywaną operację alokowania zakresu stron. Stosowane są wybrane kombinacje znaczników:

<code>MEM_COMMIT</code>	<code>0x001000</code>	Alokowanie pamięci
<code>MEM_RESERVE</code>	<code>0x002000</code>	Rezerwowanie bez alokowania
<code>MEM_RESET</code>	<code>0x080000</code>	Oznaczenie danych jako nieaktualnych
<code>MEM_TOP_DOWN</code>	<code>0x100000</code>	Użycie najwyższego dostępnego adresu
<code>MEM_WRITE_WATCH</code>	<code>0x200000</code>	Śledzenie operacji zapisu do pamięci
<code>MEM_PHYSICAL</code>	<code>0x400000</code>	Tworzenie widoku pamięci fizycznej

Protect

Określa poziom ochrony alokowanych stron. Wartości poniższej listy można łączyć z `PAGE_GUARD` lub `PAGE_NOCACHE`:

```

PAGE_NOACCESS
PAGE_READONLY
PAGE_READWRITE
PAGE_EXECUTE
PAGE_EXECUTE_READ
PAGE_EXECUTE_READWRITE

```

Zwracana wartość

Zwraca `STATUS_SUCCESS` lub kod błędu, na przykład, `STATUS_NO_MEMORY`, `STATUS_CONFLICTING_ADDRESS`, `STATUS_ALREADY_COMMITTED`, `STATUS_INVALID_PAGE_PROTECTION` lub `STATUS_PROCESS_IS_TERMINATING`.

Funkcje Win32

`VirtualAlloc`, `VirtualAllocEx`.

Uwagi

Prawie wszystkie wywołania `ZwAllocateVirtualMemory` może zrealizować funkcja `VirtualAllocEx`.

Przy alokowaniu pamięci wymagane jest jej wcześniejsze zarezerwowanie lub uwzględnienie w parametrze `AllocationType` zarówno znacznika `MEM_COMMIT`, jak i `MEM_RESERVE` (niezależnie od możliwości łączenia ich z `MEM_TOP_DOWN`).

Znacznik `MEM_RESET` został opisany w artykule Microsoft Knowledge Base Q162104 oraz nowszych wersjach Platform SDK.

Znacznik `MEM_WRITE_WATCH` dostępny jest wyłącznie w Windows 2000. Jeżeli skorzystamy z niego w systemie, który nie obsługuje śledzenia operacji zapisu, procedura nie wykonuje żadnej operacji i zwraca kod błędu `STATUS_NOT_SUPPORTED`.

Również znacznik `MEM_PHYSICAL` dostępny jest tylko w Windows 2000. Występuje on wyłącznie w połączeniu z `MEM_RESERVE` (i żadnym innym). Powoduje, że zarezerwowany zostaje zakres adresów wirtualnych, które zostaną wykorzystane do mapowania widoków pamięci fizycznej alokowanych procedurą `ZwAllocateUserPhysicalPages`.

ZwFreeVirtualMemory

Zwalnia pamięć wirtualną w zakresie adresowym trybu użytkownika.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwFreeVirtualMemory(
    IN HANDLE ProcessHandle,
    IN OUT PVOID *BaseAddress,
    IN OUT PULONG FreeSize,
    IN ULONG FreeType
);
```

Parametry

ProcessHandle

Uchwyt obiektu typu `proces`, reprezentujący proces, którego pamięć zostanie zwolniona. Wymagany jest dostęp `PROCESS_VM_OPERATION`.

BaseAddress

Wskaźnik do zmiennej określającej adres bazowy zwalnianego obszaru pamięci.

FreeSize

Wskaźnik do zmiennej, która określa (w bajtach) ilość zwalnianej pamięci i w której, po wykonaniu operacji, umieszczana jest informacja o liczbie bajtów faktycznie zwolnionych. Jeżeli wartością FreeType jest MEM_RELEASE, FreeSize musi zawierać 0.

FreeType

Zestaw znaczników opisujących wykonywaną operację zwalniania zakresu stron. Stosowane są wybrane kombinacje znaczników.

MEM_DECOMMIT	Zwalnia ale pozostawia rezerwację.
MEM_RELEASE	Zwalnia (również rezerwację)

Zwracana wartość

Zwraca STATUS_SUCCESS lub kod błędu, na przykład, STATUS_UNABLE_TO_FREE_VM, STATUS_UNABLE_TO_DELETE_SECTION, STATUS_FREE_VM_NOT_AT_BASE, STATUS_MEMORY_NOT_ALLOCATED lub STATUS_PROCESS_IS_TERMINATING.

Funkcje Win32

VirtualFree, VirtualFreeEx.

Uwagi

Prawie wszystkie wywołania ZwFreeVirtualMemory może zrealizować funkcja VirtualFreeEx.

ZwQueryVirtualMemory

Pobiera informacje o pamięci wirtualnej w zakresie adresowym trybu użytkownika.

```

NTSYSAPI
NTSTATUS
NTAPI
ZwQueryVirtualMemory(
    IN HANDLE ProcessHandle,
    IN PVOID BaseAddress,
    IN MEMORY_INFORMATION_CLASS MemoryInformationClass,
    OUT PVOID MemoryInformation,
    IN ULONG MemoryInformationLength,
    OUT PULONG ReturnLength OPTIONAL
);

```

Parametry

ProcessHandle

Uchwyt reprezentujący proces, którego pamięci będą dotyczyły zwracane informacje. Wymagany jest dostęp `PROCESS_QUERY_INFORMATION`.

BaseAddress

Wskaźnik do zmiennej określającej adres bazowy obszaru pamięci, którego dotyczyły będą zwracane informacje. Wartość zaokrąglana w dół do wielokrotności rozmiaru strony. Jeżeli pobierana klasa informacji nie jest związana z zakresem adresów, parametr może mieć wartość 0.

MemoryInformationClass

Rodzaj pobieranych informacji o pamięci wirtualnej. Listę dopuszczalnych wartości parametru określa wyliczenie `MEMORY_INFORMATION_CLASS`, które opisujemy na kolejnych stronach.

MemoryInformation

Wskazuje alokowany przez procedurę wywołującą bufor lub zmienną, w której umieszczone zostaną odpowiednie dane o pamięci wirtualnej.

MemoryInformationLength

Określa wielkość `MemoryInformation` w bajtach. Musi być odpowiednia dla klasy informacji, określonej przez `MemoryInformationClass`.

ReturnLength

Opcjonalny wskaźnik do zmiennej, w której umieszczana jest liczba bajtów faktycznie zwracanych parametrem `MemoryInformation`, o ile wywołanie jest udane. Jeżeli informacja ta nie jest potrzebna, można użyć wskaźnika pustego.

Zwracana wartość

Zwraca `STATUS_SUCCESS` albo kod błędu w rodzaju `STATUS_INVALID_INFO_CLASS`, `STATUS_INFO_LENGTH_MISMATCH`, `STATUS_INVALID_ADDRESS`, `STATUS_FILE_INVALID` lub `STATUS_PROCESS_IS_TERMINATING`.

Funkcje Win32

`VirtualQuery`, `VirtualQueryEx`.

Uwagi

Nie ma.

MEMORY_INFORMATION_CLASS

```
typedef enum _MEMORY_INFORMATION_CLASS {
    MemoryBasicInformation,
    MemoryWorkingSetList,
    MemorySectionName,
    MemoryBasicVlmInformation
} MEMORY_INFORMATION_CLASS;
```

MemoryBasicInformation

```
typedef struct _MEMORY_BASIC_INFORMATION { // Information Class 0
    PVOID BaseAddress;
    PVOID AllocationBase;
    DWORD AllocationProtect;
    SIZE_T RegionSize;
    DWORD State;
    DWORD Protect;
    DWORD Type;
} MEMORY_BASIC_INFORMATION, *PMEMORY_BASIC_INFORMATION;
```

Elementy

BaseAddress

Wirtualny adres bazowy obszaru pamięci, którego dotyczy zapytanie.

AllocationBase

Wirtualny adres bazowy alokowanego wcześniej obszaru pamięci, który obejmuje badany obszar.

AllocationProtect

Poziom ochrony stron, określony przy alokowaniu obszaru. Wartości poniższej listy mogą być łączone z PAGE_GUARD lub PAGE_NOCACHE:

```
PAGE_NOACCESS
PAGE_READONLY
PAGE_READWRITE
PAGE_EXECUTE
PAGE_EXECUTE_READ
PAGE_EXECUTE_READWRITE
```

RegionSize

Wielkość (w bajtach) obszaru, który zaczyna się od adresu bazowego alokowanego wcześniej obszaru pamięci, w którym wszystkie strony mają identyczne atrybuty ochrony i stanu.

State

Stan stron obszaru. Wykorzystuje się wymienione niżej wartości.

MEM_COMMIT	Pamięć zarezerwowana i zaalokowana
MEM_RESERVE	Pamięć zarezerwowana, nie zaalokowana
MEM_FREE	Pamięć wolna

Protect

Bieżący poziom ochrony stron obszaru.

Type

Typ stron w obszarze. MEM_FREE odpowiada wartości 0. Pozostałe wyliczono niżej.

MEM_PRIVATE	Pamięć prywatna
MEM_MAPPED	Pamięć podlegająca współużytkowaniu, mapowana z sekcji danych
MEM_IMAGE	Pamięć podlegająca współużytkowaniu, mapowana z sekcji obrazu

Uwagi

Struktura MEMORY_BASIC_INFORMATION jest identyczna ze strukturą o tej samej nazwie, zwracana przez funkcję Win32 VirtualQueryEx.

MemoryWorkingSetList

```
typedef struct _MEMORY_WORKING_SET_LIST { // Information Class 1
    ULONG NumberOfPages;
    ULONG WorkingSetList[1];
} MEMORY_WORKING_SET_LIST, *PMEMORY_WORKING_SET_LIST;
```

Elementy

NumberOfPages

Liczba stron na liście zestawu roboczego.

WorkingSetList

Tablica wpisów stron zestawu roboczego. 20 starszych bitów wpisu odpowiada wirtualnemu adresowi strony, 20 młodszych to maska znaczników. Zdefiniowane zostały poniżej.

WSLE_PAGE_READONLY	0x001	// Strona tylko-do-odczytu.
WSLE_PAGE_EXECUTE	0x002	// Strona z kodem wykonywalnym.
WSLE_PAGE_READWRITE	0x004	// Strona zapisywalna.
WSLE_PAGE_EXECUTE_READ	0x003	
WSLE_PAGE_WRITECOPY	0x005	// Strona kopiowana przy zapisie (copy-on-write).
WSLE_PAGE_EXECUTE_READWRITE	0x006	
WSLE_PAGE_EXECUTE_WRITECOPY	0x007	// Strona kopiowana po zapisie.
WSLE_PAGE_SHARE_COUNT_MASK	0x0E0	
WSLE_PAGE_SHAREABLE	0x100	// Strona ze współużytkowaniem.

Uwagi

Procedura `ZwQueryVirtualMemory` z klasą `MemoryWorkingSetList` zawsze zwraca `STATUS_SUCCESS`. Faktycznym potwierdzeniem udanej operacji pobrania danych jest upewnienie się, że wartość `MemoryInformationLength` jest większa od `ReturnLength`.

Bity znaczników niewykorzystywanych nie są ani ustawiane, ani zerowane. Zaleca się więc, przed wywołaniem `ZwQueryVirtualMemory`, wypełnić bufor `MemoryInformation` zerami.

Lista zestawu roboczego procesu obejmuje również informację o tym, czy strona jest zablokowana (w pamięci lub zestawie roboczym). Nie jest zwracana w opisywanej tu strukturze.

Klasę `MemoryWorkingSetList` wykorzystuje funkcja `PSAPI QueryWorkingSet`.

`WSLE_PAGE_SHARE_COUNT_MASK` to element dostępny wyłącznie w Windows 2000. Wartość 7 oznacza, że stronę współużytkuje co najmniej siedem procesów.

MemorySectionName

```
typedef struct _MEMORY_SECTION_NAME { // Information Class 2
    UNICODE_STRING SectionFileName;
} MEMORY_SECTION_NAME, *PMEMORY_SECTION_NAME;
```

Elementy

SectionFileName

Nazwa mapowanego pliku.

Uwagi

Parametr `BaseAddress` musi wskazywać adres bazowy mapowanej sekcji danych. Nie zostanie zwrócona nazwa mapowanej sekcji obrazu (jest to, jak się zdaje, ograniczenie możliwości procedury).

W strukturze `MemoryInformation` umieszczana jest zarówno struktura `UNICODE_STRING`, jak i sam ciąg.

Z klasy korzysta funkcja PSAPI `GetMappedFileName`.

ZwLockVirtualMemory

Blokuje pamięć wirtualną zakresu adresowego trybu użytkownika, zapewniając, że kolejne operacje dostępu do obszaru nie wywołają błędów strony.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwLockVirtualMemory(
    IN HANDLE ProcessHandle,
    IN OUT PVOID *BaseAddress,
    IN OUT PULONG LockSize,
    IN ULONG LockType
);
```

Parametry

ProcessHandle

Uchwyt reprezentujący proces, którego pamięć zostanie zablokowana. Wymagany jest dostęp `PROCESS_VM_OPERATION`.

BaseAddress

Wskaźnik do zmiennej określającej adres bazowy blokowanej pamięci wirtualnej. Zostaje w niej zwrócony adres bazowy pamięci faktycznie zablokowanej.

LockSize

Wskaźnik do zmiennej określającej wielkość obszaru pamięci wirtualnej, który zostanie zablokowany. Procedura zwraca w niej wielkość obszaru faktycznie zablokowanego.

LockType

Zestaw znaczników opisujących wykonywaną operację blokowania stron. Można łączyć następujące:

```
LOCK_VM_IN_WSL 0x01 // Blokowanie stron na liście zestawu roboczego
LOCK_VM_IN_RAM 0x02 // Blokowanie stron w pamięci fizycznej
```

Zwracana wartość

Zwraca STATUS_SUCCESS, STATUS_WAS_LOCKED lub kod błędu, na przykład, STATUS_PRIVILEGE_NOT_HELD, STATUS_WORKING_SET_QUOTA lub STATUS_PROCESS_IS_TERMINATING.

Funkcje Win32

VirtualLock.

Uwagi

Blokowanie stron w pamięci fizycznej wymaga uprawnienia SeLockMemoryPrivilege.

Blokowane są wszystkie strony zawierające jeden lub więcej bajtów z zakresu od BaseAddress do BaseAddress+LockSize.

ZwUnlockVirtualMemory

Odblokowuje pamięć wirtualną zakresu adresowego trybu użytkownika.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwUnlockVirtualMemory(
    IN HANDLE ProcessHandle,
    IN OUT PVOID *BaseAddress,
    IN OUT PULONG LockSize,
    IN ULONG LockType
);
```

Parametry

ProcessHandle

Uchwyt reprezentujący proces, którego pamięć zostanie odblokowana. Wymagany jest dostęp PROCESS_VM_OPERATION.

BaseAddress

Wskaźnik do zmiennej określającej adres bazowy odblokowywanej pamięci wirtualnej. Zostaje w niej zwrócony adres bazowy pamięci faktycznie odblokowanej.

LockSize

Wskaźnik do zmiennej określającej wielkość obszaru pamięci wirtualnej, który zostanie odblokowany. Procedura zwraca w niej wielkość obszaru faktycznie odblokowanego.

LockType

Zestaw znaczników opisujących wykonywaną operację. Można łączyć następujące:

```
LOCK_VM_IN_WSL 0x01 // Odblokowanie stron z listy zestawu roboczego
LOCK_VM_IN_RAM 0x02 // Odblokowanie stron w pamięci fizycznej
```

Zwracana wartość

Zwraca STATUS_SUCCESS albo kod błędu w rodzaju STATUS_PRIVILEGE_NOT_HELD, STATUS_NOT_LOCKED lub STATUS_PROCESS_IS_TERMINATING.

Funkcje Win32

VirtualUnlock.

Uwagi

Odblokowanie stron w pamięci fizycznej wymaga uprawnienia SeLockMemoryPrivilege.

Odblokowane zostają wszystkie strony zawierające jeden lub więcej bajtów z zakresu od BaseAddress do BaseAddress+LockSize. Wymagane jest, aby każda z nich została uprzednio zablokowana.

ZwReadVirtualMemory

Przeprowadza operację odczytu pamięci wirtualnej zakresu adresowego trybu użytkownika, należącej do innego procesu.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwReadVirtualMemory(
    IN HANDLE ProcessHandle,
    IN PVOID BaseAddress,
    OUT PVOID Buffer,
    IN ULONG BufferLength,
    OUT PULONG ReturnLength OPTIONAL
);
```

Parametry

ProcessHandle

Uchwyt reprezentujący proces, którego pamięci dotyczy operacja. Wymagany jest dostęp PROCESS_VM_READ.

BaseAddress

Adres bazowy odczytywanej pamięci wirtualnej.

Buffer

Wskazuje alokowany przez procedurę wywołującą bufor lub zmienną, w której umieszczona zostanie odczytana zawartość pamięci.

BufferLength

Rozmiar bufora Buffer w bajtach i liczba pobieranych bajtów pamięci wirtualnej.

ReturnLength

Opcjonalny wskaźnik do zmiennej, w której umieszczana jest liczba bajtów faktycznie zwracanych parametrem Buffer, o ile wywołanie jest udane. Jeżeli informacja ta nie jest potrzebna, można użyć wskaźnika pustego.

Zwracana wartość

Zwraca STATUS_SUCCESS lub kod błędu, na przykład, STATUS_ACCESS_VIOLATION lub STATUS_PROCESS_IS_TERMINATING.

Funkcje Win32

ReadProcessMemory.

Uwagi

Funkcjonalnym odpowiednikiem procedury ZwReadVirtualMemory jest funkcja Win32 ReadProcessMemory.

ZwWriteVirtualMemory

Przeprowadza operację zapisu do pamięci wirtualnej zakresu adresowego trybu użytkownika, należącej do innego procesu.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwWriteVirtualMemory(
    IN HANDLE ProcessHandle,
    IN PVOID BaseAddress,
    IN PVOID Buffer,
```

```
IN ULONG BufferLength,  
OUT PULONG ReturnLength OPTIONAL  
);
```

Parametry

ProcessHandle

Uchwyt reprezentujący proces, którego pamięci dotyczy operacja. Wymagany jest dostęp PROCESS_VM_WRITE.

BaseAddress

Adres bazowy zapisywanej pamięci wirtualnej.

Buffer

Wskazuje alokowany przez procedurę wywołującą bufor lub zmienną, w której umieszczone są zapisywane dane.

BufferLength

Rozmiar bufora Buffer w bajtach i liczba zapisywanych bajtów.

ReturnLength

Opcjonalny wskaźnik do zmiennej, w której umieszczana jest liczba bajtów faktycznie odczytanych ze zmiennej Buffer, o ile wywołanie jest udane. Jeżeli informacja ta nie jest potrzebna, można użyć wskaźnika pustego.

Zwracana wartość

Zwraca STATUS_SUCCESS lub kod błędu, na przykład, STATUS_ACCESS_VIOLATION lub STATUS_PROCESS_IS_TERMINATING.

Funkcje Win32

WriteProcessMemory.

Uwagi

Funkcjonalnym odpowiednikiem procedury ZwWriteVirtualMemory jest funkcja Win32 WriteProcessMemory.

Funkcja `WriteProcessMemory` podejmuje próbę zmiany poziomu ochrony pamięci wirtualnej, aby zapewnić dostęp z prawem zapisu. Po wykonaniu operacji opróżnia pamięć podręczną instrukcji (wywołując `ZwFlushInstructionCache`).

ZwProtectVirtualMemory

Zmienia poziom ochrony pamięci wirtualnej w zakresie adresowym trybu użytkownika.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwProtectVirtualMemory(
    IN HANDLE ProcessHandle,
    IN OUT PVOID *BaseAddress,
    IN OUT PULONG ProtectSize,
    IN ULONG NewProtect,
    OUT PULONG OldProtect
);
```

Parametry

ProcessHandle

Uchwyt reprezentujący proces, którego pamięci dotyczy operacja. Wymagany jest dostęp `PROCESS_VM_OPERATION`.

BaseAddress

Wskaźnik do zmiennej określającej adres bazowy obszaru pamięci, którego dotyczy operacja. Procedura umieszcza w niej adres bazowy obszaru, dla którego poziom ochrony faktycznie został ustawiony.

ProtectSize

Wskaźnik do zmiennej określającej w bajtach wielkość obszaru pamięci wirtualnej, której dotyczy operacja. Procedura umieszcza w niej wielkość obszaru, dla którego poziom ochrony faktycznie został ustawiony.

NewProtect

Nowy poziom ochrony stron. Wartości poniższej listy mogą być łączone z `PAGE_GUARD` lub `PAGE_NOCACHE`:

```
PAGE_NOACCESS
PAGE_READONLY
PAGE_READWRITE
PAGE_WRITECOPY
PAGE_EXECUTE
PAGE_EXECUTE_READ
PAGE_EXECUTE_READWRITE
PAGE_EXECUTE_WRITECOPY
```

OldProtect

Wskaźnik do zmiennej, w której umieszczana jest informacja o wcześniejszym poziomie ochrony pierwszej strony ze wskazanego obszaru.

Zwracana wartość

Zwraca STATUS_SUCCESS lub kod błędu, na przykład, STATUS_NOT_COMMITED lub STATUS_PROCESS_IS_TERMINATING.

Funkcje Win32

VirtualProtect, VirtualProtectEx.

Uwagi

Funkcjonalnym odpowiednikiem procedury ZwProtectVirtualMemory jest funkcja Win32 VirtualProtectEx.

ZwFlushVirtualMemory

Opróżnia mapowaną do pliku pamięć wirtualną w zakresie adresowym trybu użytkownika.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwFlushVirtualMemory(
    IN HANDLE ProcessHandle,
    IN OUT PVOID *BaseAddress,
    IN OUT PULONG FlushSize,
    OUT PIO_STATUS_BLOCK IoStatusBlock
);
```

Parametry

ProcessHandle

Uchwyt reprezentujący proces, którego pamięci dotyczy operacja. Wymagany jest dostęp PROCESS_VM_OPERATION.

BaseAddress

Wskaźnik do zmiennej określającej adres bazowy obszaru pamięci, którego dotyczy operacja. Procedura umieszcza w niej adres bazowy obszaru, który faktycznie został opróżniony. Adres powinien wskazywać obszar mapowany do sekcji danych pliku.

FlushSize

Wskaźnik do zmiennej określającej w bajtach wielkość obszaru pamięci wirtualnej, który powinien zostać opróżniony i zwracającej informację o rozmiarze obszaru, dla którego operacja faktycznie została wykonana. Jeżeli początkową wartością FlushSize jest 0, opróżniany jest zakres od BaseAddress do końca sekcji.

IoStatusBlock

Wskaźnik do zmiennej, w której umieszczany jest kod stanu operacji we-wy wymaganej do opróżnienia pamięci wirtualnej (o ile jest wykonywana).

Zwracana wartość

Zwraca STATUS_SUCCESS lub kod błędu, na przykład, STATUS_NOT_MAPPED_DATA lub STATUS_PROCESS_IS_TERMINATING.

Funkcje Win32

FlushViewOfFile.

Uwagi

Nie ma.

ZwAllocateUserPhysicalPages

Alokuje strony pamięci fizycznej.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwAllocateUserPhysicalPages(
    IN HANDLE ProcessHandle,
    IN PULONG NumberOfPages,
    OUT PULONG PageFrameNumbers
);
```

Parametry

ProcessHandle

Uchwyt reprezentujący proces, dla którego strony będą alokowane. Wymagany jest dostęp PROCESS_VM_OPERATION.

NumberOfPages

Wskaźnik do zmiennej określającej liczbę stron pamięci fizycznej do zaalokowania.

PageFrameNumbers

Wskazuje alokowany przez procedurę wywołującą bufor lub zmienną, w której umieszczone zostaną numery ramek alokowanych stron.

Zwracana wartość

Zwraca STATUS_SUCCESS albo kod błędu w rodzaju STATUS_PRIVILEGE_NOT_HELD lub STATUS_PROCESS_IS_TERMINATING.

Funkcje Win32

AllocateUserPhysicalPages.

Uwagi

Alokowanie stron pamięci fizycznej wymaga uprawnienia SeLockMemoryPrivilege.

Funkcjonalnym odpowiednikiem procedury ZwAllocateUserPhysicalPages jest funkcja Win32 AllocateUserPhysicalPages.

Funkcja AllocateUserPhysicalPages jest częścią „rozszerzeń okien adresowania” (AWE, Address Windowing Extensions), rodzaju interfejsu API, który pozwala aplikacjom korzystać z 64 GB fizycznej, niestronicowanej pamięci w 32-bitowej wirtualnej przestrzeni adresowej. Na platformie Intel, jeżeli system wyposażony jest w więcej niż 4 GB pamięci fizycznej, przy uruchamianiu ustawiany jest znacznik Physical Address Extension (PAE) w rejestrze CR4, włączający 36-bitowe adresowanie fizyczne.

Procedura ZwAllocateUserPhysicalPages dostępna jest wyłącznie w systemie Windows 2000.

ZwFreeUserPhysicalPages

Zwalnia strony pamięci fizycznej.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwFreeUserPhysicalPages(
    IN HANDLE ProcessHandle,
    IN OUT PULONG NumberOfPages,
    IN PULONG PageFrameNumbers
);
```

Parametry

ProcessHandle

Uchwyt reprezentujący proces, którego strony mają zostać zwolnione. Wymagany jest dostęp PROCESS_VM_OPERATION.

NumberOfPages

Wskaźnik do zmiennej określającej liczbę stron pamięci fizycznej, które mają zostać zwolnione. Jest w niej umieszczana liczba stron faktycznie zwolnionych.

PageFrameNumbers

Wskazuje alokowany przez procedurę wywołującą bufor lub zmienną, w której umieszczone są numery ramek zwalnianych stron.

Zwracana wartość

Zwraca STATUS_SUCCESS lub kod błędu, na przykład, STATUS_CONFLICTING_ADDRESS lub STATUS_PROCESS_IS_TERMINATING.

Funkcje Win32

FreeUserPhysicalPages.

Uwagi

Funkcjonalnym odpowiednikiem procedury ZwFreeUserPhysicalPages jest funkcja Win32 FreeUserPhysicalPages.

Procedura ZwFreeUserPhysicalPages dostępna jest wyłącznie w systemie Windows 2000.

ZwMapUserPhysicalPages

Mapuje strony pamięci fizycznej do widoku pamięci fizycznej.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwMapUserPhysicalPages(
    IN PVOID BaseAddress,
    IN PULONG NumberOfPages,
    IN PULONG PageFrameNumbers
);
```

Parametry

BaseAddress

Adres należący do widoku pamięci fizycznej, do którego wykonane zostanie mapowanie. Jeżeli jest to potrzebne, zapewnione jest zaokrąglenie w dół do wielokrotności rozmiaru strony. Widok pamięci fizycznej tworzony jest wywołaniem `ZwAllocateVirtualMemory` z parametrem `AllocationType` o wartości `MEM_PHYSICAL|MEM_RESERVE`.

NumberOfPages

Wskaźnik do zmiennej określającej liczbę stron pamięci fizycznej do mapowania.

PageFrameNumbers

Wskazuje alokowany przez procedurę wywołującą bufor lub zmienną, w której umieszczone są numery ramek mapowanych stron. Jeżeli będzie to wskaźnik pusty, usuwane jest mapowanie pamięci fizycznej do adresu `BaseAddress`.

Zwracana wartość

Zwraca `STATUS_SUCCESS` lub kod błędu, na przykład, `STATUS_CONFLICTING_ADDRESS` lub `STATUS_PROCESS_IS_TERMINATING`.

Funkcje Win32

`MapUserPhysicalPages`.

Uwagi

Funkcjonalnym odpowiednikiem procedury `ZwMapUserPhysicalPages` jest funkcja Win32 `MapUserPhysicalPages`.

Procedura `ZwMapUserPhysicalPages` dostępna jest wyłącznie w systemie Windows 2000.

Wymagane jest, aby strony fizyczne zostały wcześniej zaalokowane procedurą `ZwAllocateUserPhysicalPages`.

Z niewiadomych przyczyn procedura `ZwMapUserPhysicalPages` nie umożliwia wskazania procesu, dla którego wykonane ma zostać mapowanie. Jest to istotna różnica w stosunku do wszystkich innych podobnych procedur, które uwzględniają parametr `ProcessHandle`.

ZwMapUserPhysicalPagesScatter

Mapuje strony pamięci fizycznej do widoku pamięci fizycznej.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwMapUserPhysicalPagesScatter(
    IN PVOID *BaseAddresses,
    IN PULONG NumberOfPages,
    IN PULONG PageFrameNumbers
);
```

Parametry

BaseAddresses

Wskazuje alokowany przez procedurę wywołującą bufor lub zmienną, w której umieszczona jest tablica adresów wirtualnych (należących do widoku pamięci fizycznej), do których mapowana będzie pamięć fizyczna. Jeżeli jest to potrzebne, zapewnione jest zaokrąglenie w dół do wielokrotności rozmiaru strony. Widok pamięci fizycznej tworzony jest wywołaniem `ZwAllocateVirtualMemory` z parametrem `AllocationType` o wartości `MEM_PHYSICAL|MEM_RESERVE`.

NumberOfPages

Wskaźnik do zmiennej określającej liczbę stron pamięci fizycznej do mapowania.

PageFrameNumbers

Wskazuje alokowany przez procedurę wywołującą bufor lub zmienną, w której umieszczone są numery ramek mapowanych stron. Jeżeli będzie to wskaźnik pusty, usuwane jest mapowanie pamięci fizycznej do adresu `BaseAddress`.

Zwracana wartość

Zwraca `STATUS_SUCCESS` lub kod błędu, na przykład, `STATUS_CONFLICTING_ADDRESS` lub `STATUS_PROCESS_IS_TERMINATING`.

Funkcje Win32

`MapUserPhysicalPagesScatter`.

Uwagi

Funkcjonalnym odpowiednikiem procedury `ZwMapUserPhysicalPagesScatter` jest funkcja Win32 `MapUserPhysicalPagesScatter`.

Procedura `ZwMapUserPhysicalPagesScatter` dostępna jest wyłącznie w systemie Windows 2000.

Wymagane jest, aby strony fizyczne zostały wcześniej zaalokowane procedurą `ZwAllocatUserPhysicalPages`.

ZwGetWriteWatch

Pobiera adresy stron zapisanych do obszaru pamięci wirtualnej.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwGetWriteWatch(
    IN HANDLE ProcessHandle,
    IN ULONG Flags,
    IN PVOID BaseAddress,
    IN ULONG RegionSize,
    OUT PULONG Buffer,
    IN OUT PULONG BufferEntries,
    OUT PULONG Granularity
);
```

Parametry

ProcessHandle

Uchwyt reprezentujący proces, którego pamięci dotyczyć będą dane śledzenia operacji zapisu do pamięci wirtualnej. Wymagany jest dostęp `PROCESS_VM_OPERATION`.

Flags

Tablica znaczników. Zdefiniowana jest wartość:

```
WRITE_WATCH_RESET_FLAG 0x01 // Zeruj dane śledzenia zapisu.
```

BaseAddress

Adres bazowy obszaru pamięci, dla którego pobrane będą dane śledzenia.

RegionSize

Wielkość (w bajtach) obszaru, dla którego pobrane będą dane śledzenia.

Buffer

Wskazuje alokowany przez procedurę wywołującą bufor lub zmienną, w której umieszczona zostanie tablica adresów stron, w których dokonano zapisu od momentu alokacji obszaru lub zerowania danych śledzenia.

BufferEntries

Wskazuje zmienną, która określa największą dopuszczalną ilość zwracanych adresów stron i w której umieszczana jest liczba adresów faktycznie zwróconych.

Granularity

Wskazuje zmienną, w której podawana jest (liczona w bajtach) ziarnistość śledzenia operacji zapisu. Normalną wartością jest rozmiar strony fizycznej.

Zwracana wartość

Zwraca STATUS_SUCCESS albo kod błędu w rodzaju STATUS_PROCESS_IS_TERMINATING, STATUS_INVALID_PARAMETER_1, STATUS_INVALID_PARAMETER_2, STATUS_INVALID_PARAMETER_3 lub STATUS_INVALID_PARAMETER_5.

Funkcje Win32

GetWriteWatch.

Uwagi

Większość wywołań ZwGetWriteWatch może zrealizować funkcja GetWriteWatch.

Procedura ZwGetWriteWatch dostępna jest wyłącznie w systemie Windows 2000.

ZwResetWriteWatch

Zeruje dane śledzenia operacji zapisu do pamięci wirtualnej dla wskazanego obszaru tej pamięci.

```
NTSYSAPI
NTSTATUS
NTAPI
ZwResetWriteWatch(
    IN HANDLE ProcessHandle,
    IN PVOID BaseAddress,
    IN ULONG RegionSize
);
```

Parametry

ProcessHandle

Uchwyt reprezentujący proces, którego pamięci dotyczy będzie operacja zerowania danych śledzenia operacji zapisu do pamięci wirtualnej. Wymagany jest dostęp PROCESS_VM_OPERATION.

BaseAddress

Adres bazowy obszaru pamięci, dla którego dane śledzenia zostaną wyzerowane.

RegionSize

Wielkość (w bajtach) obszaru, dla którego dane śledzenia zostaną wyzerowane.

Zwracana wartość

Zwraca STATUS_SUCCESS albo kod błędu w rodzaju STATUS_PROCESS_IS_TERMINATING, STATUS_INVALID_PARAMETER_1, STATUS_INVALID_PARAMETER_2 lub STATUS_INVALID_PARAMETER_3.

Funkcje Win32

ResetWriteWatch.

Uwagi

Większość wywołań ZwResetWriteWatch może zrealizować funkcja ResetWriteWatch.

Procedura ZwResetWriteWatch dostępna jest wyłącznie w systemie Windows 2000.