Лекция

Управление памятью в Windows

LPVOID VirtualAlloc(

LPVOID lpvAddress, // адрес области

DWORD cbSize, // размер области

DWORD fdwAllocationType, // способ получения памяти

DWORD fdwProtect); // тип доступа

Для параметра fdwAllocationType вы можете использовать одно из следующих значений:

|  |  |
| --- | --- |
| Значение | Описание |
| MEM\_RESERVE | Функция VirtualAlloc XE "VirtualAlloc" выполняет резервирование диапазона адресов в адресном пространстве приложения |
| MEM\_COMMIT | Выполняется выделение страниц памяти для непосредственной работы с ними. Выделенные страницы заполняются нулями |
| MEM\_TOP\_DOWN | Память выделяется в области верхних адресов адресного пространства приложения |

С помощью параметра fdwProtect приложение может установить желаемый тип доступа для заказанных страниц. Можно использовать одно из следующих значений:

|  |  |
| --- | --- |
| Значение | Разрешенный доступ |
| PAGE\_READWRITE | Чтение и запись |
| PAGE\_READONLY | Только чтение |
| PAGE\_EXECUTE | Только исполнение программного кода |
| PAGE\_EXECUTE\_READ | Исполнение и чтение |
| PAGE\_EXECUTE\_READWRITE | Исполнение, чтение и запись |
| PAGE\_NOACCESS | Запрещен любой вид доступа |
| PAGE\_GUARD | Сигнализация доступа к странице. Это значение можно использовать вместе с любыми другими, кроме PAGE\_NOACCESS XE "PAGE\_NOACCESS" |
| PAGE\_NOCACHE | Отмена кэширования для страницы памяти. Используется драйверами устройств. Это значение можно использовать вместе с любыми другими, кроме PAGE\_NOACCESS XE "PAGE\_NOACCESS" |

**Освобождение виртуальной памяти**

После использования вы должны освободить полученную ранее виртуальную память, вызвав функцию VirtualFree XE "VirtualFree" :

BOOL VirtualFree(

LPVOID lpvAddress, // адрес области

DWORD cbSize, // размер области

DWORD fdwFreeType); // выполняемая операция

Через параметры lpvAddress и cbSize передаются, соответственно, адрес и размер освобождаемой области.

Если вы зарезервировали область виртуальной памяти функцией VirtualAllocс параметром MEM\_RESERVE для последующего получения страниц в пользование и затем вызвали эту функцию еще раз с параметром MEM\_COMMIT, вы можете либо совсем освободить область памяти, обозначив соответствующие страницы как свободные, либо оставить их зарезервированными, но не используемыми.

В первом случае вы должны вызвать функцию VirtualFree с параметром fdwFreeType, равным MEM\_RELEASE, во втором - с параметром MEM\_DECOMMIT.

### Три состояния страниц виртуальной памяти

Страницы виртуальной памяти, принадлежащие адресному пространству процесса в Microsoft Windows NT, могут находиться в одном из трех состояний. Они могут быть свободными (free XE "free" ), зарезервированными (reserved) или выделенными для использования (committed). В адресном пространстве приложения есть также относительно небольшое количество страниц, зарезервированных для себя операционной системой. Эти страницы недоступны приложению.

Функция VirtualAlloc XE "VirtualAlloc" может либо зарезервировать свободные страницы памяти (для чего ее нужно вызвать с параметром MEM\_RESERVE), либо выделить свободные или зарезервированные страницы для непосредственного использования (для этого функция вызывается с параметром MEM\_COMMIT). Приложение может либо сразу получить страницы памяти в использование, либо предварительно зарезервировать их, обеспечив доступное сплошное адресное пространство достаточного размера.

Для того чтобы зарезервированная или используемая область памяти стала свободной, вы должны вызвать для нее функцию VirtualFree с параметром MEM\_RELEASE.

Вы можете перевести страницы используемой области памяти в зарезервированное состояние, не освобождая соответствующего адресного пространства. Это можно сделать при помощи функции VirtualFree XE "VirtualFree" с параметром MEM\_DECOMMIT XE "MEM\_DECOMMIT" .

На рис. 1.11 мы показали три состояния страниц виртуальной памяти и способы перевода страниц из одного состояния в другое.

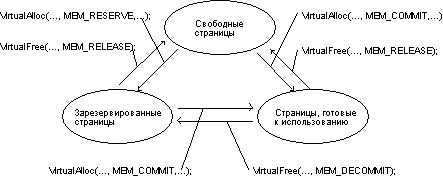


Рис. 1.11. Три состояния страниц виртуальной памяти

### Фиксирование страниц виртуальной памяти

Вирутальная память по сравнению с физической оперативной памятью обладает одним существенным недостатком - невысоким быстродействием. Это из-за того, что файл страниц расположен на диске. В том случае, когда вы, например, создаете драйвер периферийного устройства, обрабатывающий аппаратные прерывания, вам могут потребоваться области памяти, гарантированно размещенные в физической оперативной памяти.

В программном интерфейсе Microsoft Windows NT есть функция VirtualLock, с помощью которой нетрудно зафиксировать нужное вам количество страниц в физической памяти.

Прототип функции VirtualLock XE "VirtualLock" представлен ниже:

BOOL VirtualLock(

LPVOID lpvAddress, // адрес начала фиксируемой

// области памяти

DWORD cbSize); // размер области в байтах

Через параметр lpvAddress вы должны передать адрес фиксируемой области памяти, расположенной в страницах, готовых к использованию.

Параметр cbSize, задающий размер фиксируемой области памяти, может иметь значение, которое не кратно размеру страницы. В результате вызова функции будет зафиксировано столько страниц, сколько нужно для размещения указанной области.

Для расфиксирования страниц памяти следует вызвать функцию VirtualUnlock, имеющую аналогичное назначение параметров:

BOOL VirtualUnlock(

LPVOID lpvAddress, // адрес начала расфиксируемой

// области памяти

DWORD cbSize); // размер области в байтах

Сколько страниц памяти можно зафиксировать функцией VirtualLock?

Не очень много. По умолчанию приложение может зафиксировать не более 30 страниц виртуальной памяти.

Однако при необходимости вы можете увеличить это значение при помощи функции SetProcessWorkingSetSize, описанной в SDK.

### Изменение типа разрешенного доступа для страниц памяти

При получении страниц памяти в пользование функцией VirtualAlloc вы можете в последнем параметре указать тип доступа, разрешенного для этих страниц. В процессе работы приложение может изменять тип доступа для полученных им страниц при помощи функции VirtualProtect, прототип которой представлен ниже:

BOOL VirtualProtect(

LPVOID lpvAddress, // адрес области памяти

DWORD cbSize, // размер области памяти в байтах

DWORD fdwNewProtect, // новый тип разрешенного доступа

PDWORD pfdwOldProtect); // указатель на переменную,

// в которую будет записан прежний код доступа

Через параметр lpvAddress вы должны передать адрес области памяти, расположенной в готовых для использования страницах (а не в зарезервированных страницах).

Новый тип доступа передается через параметр fdwNewProtect. Здесь вы можете использовать все константы, что и для последнего параметра функции VirtualAlloc, например, PAGE\_READWRITE или PAGE\_READONLY.

Заметим, что функция VirtualProtect позволяет изменить код доступа только для тех страниц, которые созданы вызывающим ее процессом. При необходимости приложение может изменить код доступа страниц другого процесса, имеющего код доступа PROCESS\_VM\_OPERATION (например, процесса, созданного приложением). Это можно сделать при помощи функции VirtualProtectEx, прототип которой представлен ниже:

BOOL VirtualProtectEx(

HANDLE hProcess, // идентификатор процесса

LPVOID lpvAddress, // адрес области памяти

DWORD cbSize, // размер области памяти в байтах

DWORD fdwNewProtect, // новый тип разрешенного доступа

PDWORD pfdwOldProtect); // указатель на переменную,

// в которую будет записан прежний код доступа

Через параметр hProcess функции VirtualProtectEx XE "VirtualProtectEx" следует передать идентификатор процесса. Подробнее об этом идентификаторе вы узнаете из главы, посвященной мультизадачности в операционной системе Microsoft Windows NT.

### Получение информации об использовании виртуальной памяти

В программном интерфейсе Microsoft Windows NT есть средства для получения справочной информации об использовании процессами виртуальной памяти. Это функции VirtualQuery XE "VirtualQuery" и VirtualQueryEx XE "VirtualQueryEx" . С помощью них процесс может исследовать, соответственно, собственное адресное пространство и адресное пространство других процессов, определяя такие характеристики областей виртуальной памяти, как размеры, тип доступа, состояние и так далее.