

Изменение атрибутов доступа к виртуальной странице

Изменить атрибуты доступа к области виртуальной памяти можно при помощи вызова функции `VirtualProtect`, которая имеет следующий прототип:

```
BOOL VirtualProtect(  
    LPVOID lpAddress, // адрес области памяти  
    SIZE_T dwSize, // размер области памяти  
    DWORD flNewProtect, // флаги новых атрибутов доступа  
    PDWORD lpflOldProtect // указатель на старые атрибуты доступа  
);
```

В случае успешного завершения эта функция возвращает значение, а в случае неудачи — `FALSE`. Параметры этой функции имеют тот же смысл, что и при распределении виртуальной памяти функцией `VirtualAlloc`. Единственное отличие состоит в том, что старые атрибуты доступа возвращаются функцией `VirtualProtect` по адресу `lpflOldProtect`, который должна установить вызывающая программа.

Инициализация и копирование блоков виртуальной памяти

Чтобы заполнить блок памяти определенным значением, используется функция `FillMemory`, которая имеет следующий прототип:

```
VOID FillMemory(  
    PVOID Destination, // адрес блока памяти  
    SIZE_T Length, // длина блока  
    BYTE Fill // символ-заполнитель  
);
```

Эта функция заполняет блок памяти, длина в байтах и базовый адрес которого задаются соответственно параметрами `Length` и `Destination`, символом, заданным в параметре `Fill`.

Если блок памяти необходимо заполнить нулями, то для этого можно использовать функцию `ZeroMemory`, которая имеет следующий прототип:

```
VOID ZeroMemory(  
    PVOID Destination, // адрес блока памяти  
    SIZE_T Length, // длина блока  
);
```

Параметры этой функции имеют то же назначение, что и параметры функции `FillMemory`, исключая символ-заполнитель.

Для копирования блока виртуальной памяти используется функция `CopyMemory`, которая имеет следующий прототип:

```
VOID CopyMemory(  
    PVOID Destination, // адрес области назначения  
    CONST VOID *Source, // адрес исходной области  
    SIZE_T Length // длина блока памяти  
);
```

Эта функция копирует блок памяти, длина в байтах и базовый адрес которого задаются соответственно параметрами Length и Source в область памяти по адресу Destination. Результат выполнения функции CopyMemory непредсказуем, если исходный и результирующий блоки памяти перекрываются.

Для копирования перекрывающихся блоков памяти используется функция MoveMemory, которая имеет следующий прототип:

```
VOID MoveMemory(  
    PVOID Destination, // адрес области назначения  
    CONST VOID *Source, // адрес исходной области  
    SIZE_T Length // длина блока памяти  
);
```

Параметры этой функции полностью совпадают с параметрами функции CopyMemory.

Инициализация и копирование блоков виртуальной памяти

```
CopyVirMemory (Глобальная область)  
1  #include <windows.h>  
2  #include <iostream>  
3  using namespace std;  
4  int main()  
5  {  
6      BYTE* a, * b; // указатели на блоки памяти  
7      const int size = 1000; // размерность массива  
8      // распределяем виртуальную память  
9      a = (BYTE*)VirtualAlloc(NULL, size, MEM_COMMIT, PAGE_READWRITE);  
10     if (!a)  
11     {  
12         cout << "Virtual allocation failed." << endl;  
13         return GetLastError();  
14     }  
15     b = (BYTE*)VirtualAlloc(NULL, size, MEM_COMMIT, PAGE_READWRITE);  
16     if (!b)  
17     {  
18         cout << "Virtual allocation failed." << endl;  
19         return GetLastError();  
20     }  
21     // инициализируем символом X  
22     FillMemory(a, size, 'X');  
23     // копируем блок A в блок B  
24     CopyMemory(b, a, size);  
25     // распечатываем результат  
26     cout << "b[10] = " << b[10] << endl;  
27     // освобождаем виртуальную память  
28     if (!VirtualFree(a, 0, MEM_RELEASE))  
29     {  
30         cout << "Memory release failed." << endl;  
31         return GetLastError();  
32     }  
33     return 0;  
34 }
```