What’s the difference between setting a page’s protection to PAGE_NOACCESS and freeing it?

devblogs.microsoft.com/oldnewthing/20231019-00

October 19, 2023

Raymond Chen

A customer wanted to know if, after using `VirtualAlloc` with `MEM_COMMIT` to commit some memory, they could free parts of it by calling `VirtualProtect` with `PAGE_NOACCESS`.

No, changing a page to no-access does not free it. It makes the page inaccessible, but it’s still there. If you can restore access later, the old contents will still be there.

```
SYSTEM_INFO info;
GetSystemInfo(&info);

// Allocate two pages of read-write data
auto p = (BYTE*)VirtualAlloc(
    nullptr, info.dwPageSize * 2,
    MEM_RESERVE | MEM_COMMIT, PAGE_READWRITE);

// Set the first byte to 1
p[0] = 1;

// Remove access from the first page
DWORD prev;
VirtualProtect(p, info.dwPageSize, PAGE_NOACCESS, &prev);

// At this point, any access to p[0] will crash with an access violation.

// Restore access to the memory.
VirtualProtect(p, info.dwPageSize, PAGE_READWRITE, &prev);

// The old values are still there!
ASSERT(p[0] == 1);
```

If you want to free the memory, you can use `VirtualFree` with the `MEM_DECOMMIT` flag.

```
// Allocate two pages of read-write data
auto p = (BYTE*)VirtualAlloc(
    nullptr, info.dwPageSize * 2,
    MEM_RESERVE | MEM_COMMIT, PAGE_READWRITE);

// Set the first byte of each page to 1
p[0] = 1;
p[info.dwPageSize] = 1;


// Decommit the first page
DWORD prev;
VirtualFree(p, info.dwPageSize, MEM_DECOMMIT);

// The memory is now gone!
// At this point, any access to p[0] will crash with an access violation.

// Commit a new page.
VirtualAlloc(p, info.dwPageSize,
    MEM_COMMIT, PAGE_READWRITE);

// The old values are gone!
ASSERT(p[0] != 1);

// But the second page is still good.
ASSERT(p[info.dwPageSize] == 1);
```

If you unreserve a region of address space with `VirtualFree(..., MEM_RELEASE)`, then all the associated memory in the region is decommitted as part of the release.