Using virtual memory placeholders to allocate contiguous address space for multiple purposes

devblogs.microsoft.com/oldnewthing/20240201-00

February 1, 2024

Raymond Chen

Windows 10 Version 1803 added support for virtual memory *placeholders* which let you reserve address space in a way that can be replaced by another virtual memory allocation.

Suppose you want to create two adjacent memory mappings of 64KB. Before the advent of placeholders, you would have to do something like this (pseudocode):

```
bool retry = true;
while (retry) {
    addr = VirtualAlloc(MEM_RESERVE, 128KB);
    if (!addr) fail();

    VirtualFree(addr, 0, MEM_RELEASE);

    view1 = MapViewOfFileEx(hMapping1, addr, 64KB);
    if (view1) {
        view2 = MapViewOfFileEx(hMapping2, addr + 64KB, 64KB);
        if (view2) {
            retry = false;
        } else {
            UnmapViewOfFile(view1);
        }
    }
}
```

(The pseudocode would be a little simpler with RAII types to manage the cleanup, but I did things the manual way.)

First, we reserve 128KB of contiguous address space, looking for a place we can put our two adjacent 64KB memory blocks. If that fails, then there is no contiguous 128KB memory block, and we give up.

If it succeeds, then we free that memory and then try to map the two 64KB blocks into the space that we recently freed up. If either one fails (due to a multithreaded race where another thread allocated that address space out from under us), then we unwind all the work we did and start over.

Virtual memory placeholders let you reserve memory in a way that allows a later memory-mapping operation to take over that address space without you having to free it first. It closes the race window where you have to temporarily free the address space so you can allocate something else.

The categories of memory functions that support allocating address space into an existing placeholder are currently virtual memory allocation (using new flags added to `VirtualAlloc2`) and file mapping (using new flags added to `MapViewOfFile3`).

With placeholders, the algorithm for creating adjacent memory mappings is much simpler (still pseudocode):

```
addr = VirtualAlloc2(MEM_RESERVE | MEM_RESERVE_PLACEHOLDER, 128KB);
if (!addr) fail();

VirtualFree(addr, 64KB, MEM_RELEASE | MEM_PRESERVE_PLACEHOLDER);

view1 = MapViewOfFile3(hMapping1, addr, 64KB, MEM_REPLACE_PLACEHOLDER);
if (!view1) {
    VirtualFree(addr, 0, MEM_RELEASE);
    fail();
}

view2 = MapViewOfFile3(hMapping2, addr + 64KB, 64KB, MEM_REPLACE_PLACEHOLDER);
if (!view2) {
    VirtualFree(addr, 0, MEM_RELEASE);
    UnmapViewOfFile(view1);
    fail();
}
```

First, we use the new `MEM_RESERVE_PLACEHOLDER` flag to allocate a 128KB placeholder.

Next, we use `VirtualFree` with the new `MEM_PRESERVE_PLACEHOLDER` to say that we want to split the original placeholder into two placeholders, splitting at the 64KB mark. This splits the 128KB block into two 64KB blocks.

Finally, we use `MapViewOfFile3` with the new `MEM_REPLACE_PLACEHOLDER` flag to indicate that we want the newly-mapped views to go into a space that currently holds a placeholder. Note that when replacing a placeholder, the new allocation must exactly match the position and size of the existing placeholder. No partial replacements allowed. It’s all or nothing. (If you want to do a partial replacement, then split the placeholder like we did here.)

There are also flags for merging two adjacent placeholders into one big placeholder, or for freeing virtual memory or file mappings and leaving a placeholder behind. There’s a full sample in the documentation for `VirtualAlloc2`, so I’ll defer to that page.

Bonus chatter: Peter Cooper Jr. pointed out in a comment that I didn't provide any motivation for placeholders.

One scenario is the "scatter/gather" case, where you want to map multiple files into adjacent blocks so you can treat them as if they were one giant file. Another is to simplify implementation of a ring buffer, where you map the same physical buffer into two adjacent blocks so that structures which straddle the boundary do not require special treatment.