How can I reserve a range of address space and receive notifications when the program first reads or writes a page in the range?

devblogs.microsoft.com/oldnewthing/20180124-00

January 24, 2018

Raymond Chen

A customer wanted to reserve a range of address space and be notified when the program first reads or writes a page in the range.

It's not clear what the customer's goal is, but if it's true that all they want is to be notified of the access, without affecting the underlying memory, then it's not so hard.

In the simplest case, you can mark the page as `PAGE_GUARD`. This will raise a guard page violation the first time the program reads from or writes to the memory. You can log whatever you need and then indicate that you handled the exception and want execution to continue as if no exception had occurred. The guard page violation is raised only once per page. After it's done, the memory behaves normally, either as a normal read-only page or a normal read-write page, depending on how you allocated the memory.

In the more complicated case where you want to detect reads and writes separately, you can mark the page as `PAGE_NOACCESS`. If that's all you do, then this will raise an access violation every time the program reads from or writes to the memory. But what you can do is to inspect the exception reason, and if it's "read", then change the protection from `PAGE_NOACCESS` `PAGE_READONLY` to upgrade the page from no-access to read-only. If it's "write", then upgrade all the way to `PAGE_READWRITE`. Log the information, change the page protections, and indicate that execution should continue.

Watch out for the multithreaded case, if two threads take access violations simultaneously on the same page.

If you want this fancy memory management only for the duration of a function call, then you can install a structured exception handler around the code whose access is being monitored. If you need this beyond the scope of a single function, then you can use a vectored exception handler.

A variation of this is where you want to commit empty pages on demand. In that case, you the same technique that the `FormatMessage` function used to use: Reserve a bunch of memory, and then install an exception handler that creates the memory on demand in response to an accss violation on one of the pages you're managing.

There is a gotcha here: Your custom page fault handler will be called only for page faults incurred by user mode. If the program passes a buffer to kernel mode (say as the source of a `WriteFile` or the destination of a `ReadFile`), then kernel mode will complain that the buffer is invalid because not all the pages are committed with appropriate access. To work around this, you'll have to manually fault in the pages with the appropriate protections before using then as source or destination buffers in kernel calls.

Okay, so this works in the case where the program merely wants to be notified of the access, or if it wants to swoop in and allocate blank pages. Next time, we'll look at a more complicated scenario.

Raymond Chen

Follow