Why would the incremental linker insert padding between section fragments?

devblogs.microsoft.com/oldnewthing/20190114-00

January 14, 2019

Raymond Chen

Last year, I briefly discussed the subtleties of inter-fragment section padding, and noted that the incremental linker is a common source of this padding. Commenter DanStur wondered why the incremental linker inserts padding between section fragments.

The goal of the incremental linker is to permit rapid generation of a new binary given an old binary and a small number of changes, so that only a relatively small number of bytes in the binary need to be modified. Among other things, the linker adds padding in places that would permit it to accommodate objects that change size.

For example, the incremental linker adds padding after each function (which is represented as a code section fragment). That way, if you make a change to the function that causes it to become a little bit bigger, the linker can plop the modified function into the space used by the original function and its padding, and it's done. Not only did the function's start address not change, neither did the start addresses of any other functions change. This means that the linker doesn't have to chase down all the places that call a function and update the target addresses.

The incremental linker does the same thing with data section fragments. If your object file declares 16 bytes of data, the incremental linker will add padding after each data fragment. If you add a new variable to the source file, the incremental linker can put it in the padding (assuming there's room). This means that the linker doesn't need to move any data variables around, which means that no data addresses need to be updated.

Section fragment padding is just a special case of object padding. In the case of a section fragment holding data, if you add a new variable to the section fragment, and the new variable fits inside the padding, then the linker can just plop the new variable into the padding and not have to move any existing data around.

If the linker cannot accommodate the change with the existing padding, then it has to perform a lot more work to re-link the program and update all the addresses. But the linker will add *new* padding to the re-linked program, so as to improve the likelihood that the next link request can be performed incrementally if the change is minor.

Raymond Chen

Follow