Reverse-engineering what a “short” section is

devblogs.microsoft.com/oldnewthing/20241029-00

October 29, 2024

There is a linker error LNK2004 that says that the “short” section is too long. What does this mean? What’s a “short” section? What makes it shorter than a “long” section?

You actually have enough information to figure this out.

Look closely at the error message, and you’ll see a big clue:

```
gp relative fixup overflow to 'target'; short section 'section' is too large or out
of range.
```

The error message says “gp relative fixup”, and “gp” is the traditional name of a dedicated register on Itanium and PowerPC that points to a module’s global variables.

The reach of the `gp` register is 4MB on Itanium and 64KB on PowerPC, and the limited reach is consistent with the “relative fixup” part of the error message. It’s saying that the linker needs to fix up an indexed indirect instruction, but the offset is too large for the instruction’s reach.

Recall that on Itanium and PowerPC, global variables are divided into two classes: “small” and “large”. All the “small” variables are placed in a block of memory that the `gp` register points to. All the “large” variables are placed elsewhere in the process, and a pointer to the “large” variable is added to the list of “small” variables.

gp	→	Small A		
		•	→	Large B
		Small C		
		Small D		
		•	→	Large E

In the above example, we have the `gp` register pointing to a block of memory (known in PowerPC as the Table of Contents). This block of memory contains some small variables (Small A, Small C, and Small D), and it also has pointers to large variables (Large B and Large E).

Loading a small variable can be done by indirecting from the `gp` register.

```
// Itanium
    addl    r30 = 16, gp ;; // Calculate address of Small C variable
    ld4     r30 = [r30]     // load 32-bit value from Small C

// PowerPC
    lwz     r3, 16(r2)      // load 32-bit value from Small C (relative to r2)
```

But loading a large variable requires two steps, one to load the pointer as a small variable, and then another to dereference that pointer to get to the real data.

```
// Itanium
    addl    r30 = 8, gp ;;  // Calculate address of Large B pointer (in small data)
    ld8     r30 = [r30] ;;  // Load the pointer to Large B
    ld4     r30 = [r30] ;;  // Load 32-bit value from Large B

// PowerPC
    lwz     r3, 16(r2)      // load pointer to Large B
    lwz     r3, 0(r3)       // Load 32-bit value from Large B
```

Variables placed in the “small variables” block are faster to access, but you have space for only a limited number of them, and large variables each consume a pointer in the small variables block as well.

Now the error message makes sense: You tried to pack too much data into the “small variables” block. The linker apparently uses the terms “short” and “long” to refer to what we have been informally calling “small” and “large”.

Windows dropped support for both Itanium and PowerPC quite some time ago, so you’re not going to see these errors coming from any version of the linker less than a decade old. The error message is vestigial. Nevertheless, the error message number has already been assigned, so you’ll see it in a list of error codes, even though there is no longer any way to cause it to be generated.

Related reading: What is the thread reaper?