How to dynamically add data to an executable file (part 1 of 5)

Contents, introduction and overview

Contents

Introduction

In *article #2* we discussed how it is often useful to distribute data that is embedded in a program's executable file. That article solved the problem by writing the data to resource files and linking them into the program at compile time. This article solves the same problem in a different way – by appending the data to the executable file. This has the advantage that the data doesn't need to be linked in at compile time and can be added or updated dynamically. Data can be attached to an already compiled file. The only disadvantage of this method is that it's slightly harder to read the data at run time than it is when using resources.

A typical use for this technique would be in creating install programs. We have the actual installer as a program stub and append the files to be installed to the end of the stub. The installer stub contains code that can extract the files.

Overview

The Windows PE file format permits additional data to be appended to the executable file. This data is ignored by the Windows program loader, so we can use the data for our own purposes without affecting the program code. For the purposes of this article let's call this data the *payload*.

Credits

The concepts used in this article are based on, and extend, those presented by *Daniel Polistchuck* in *his article* on the Borland Community Forum.

The key problem is how we denote that an executable file has a payload and, if so, what size it is. We must be able to do this without modifying the executable portion of the file. Following *Daniel Polistchuck* we will use a special record to identify the payload. This record will follow the payload data.

So, an executable file that contains a payload has three main components (in order):

1. The original executable file.
2. The payload data.
3. A *footer record* that identifies that a payload is present and records the size of the executable code and the payload.

Our task, then, is to create code that can add create, modify or delete a payload, check for existence of a payload and read a payload's data. To enable this we must be able to detect, read and update the payload

footer. We begin our discussion in the *next part* by looking at how to handle this footer.
