Why does the Windows Portable Executable (PE) format have separate tables for import names and import addresses?, part 2

devblogs.microsoft.com/oldnewthing/20231130-00

November 30, 2023

Raymond Chen

In the Windows Portable Executable (PE) format, the image import descriptor table describes the functions imported from a specific target DLL.

```
struct IMAGE_IMPORT_DESCRIPTOR {
    DWORD   OriginalFirstThunk;
    DWORD   TimeDateStamp;
    DWORD   ForwarderChain;
    DWORD   Name;
    DWORD   FirstThunk;
};
```

The `OriginalFirstThunk` points to an array of pointer-sized `IMAGE_THUNK_DATA` structures which describe the functions being imported. The `FirstThunk` points to an array of pointers, whose initial values are a copy of the values pointed to by `OriginalFirstThunk`.

Last time, we looked at why the two identical tables can't be merged: Because the second table can be modified by binding, and then the two copies both contain distinct information that is needed at run time.

Okay, so you can't merge the two tables, but why not combine them into a single mega-table? Why split it into two mini-tables?

The tables are kept separate because one is read-only and the other is read-write. Splitting them up allows the read-only part to be combined with other read-only data, and the read-write part to be combined with other read-write data. This allows for a reduction in the number of read-write pages. This benefit could end up being small for a single DLL, but if the DLL is used by many processes, the multiplicative effect can be significant.