The format of accelerator table resources

devblogs.microsoft.com/oldnewthing/20070316-00

March 16, 2007

Raymond Chen

Continuing in the extremely sporadic series on the format of resources, today we'll take a look at accelerator tables. This topic is so simple, I'll cover both 16-bit and 32-bit resources on the same day!

In 16-bit Windows, the format of an accelerator table resource was simply an array of `ACCEL` structures.

```
typedef struct tagACCEL {
    BYTE fVirt;
    BYTE bPadding; /* making the padding explicit */
    WORD key;
    WORD cmd;
} ACCEL, *LPACCEL;
```

This array is the same array you would pass to the `CreateAcceleratorTable`, with one important difference: The `fVirt` of the last entry in the accelerator resource has its high bit set to indicate that it is the end of the table.

The format of 32-bit accelerator table resources is nearly identical to its 16-bit counterpart. The only difference is the addition of an additional word of padding to bring the size of the structure up to a multiple of four bytes.

```
typedef struct tagACCEL_RESOURCE {
    BYTE fVirt;
    BYTE bPadding; /* making the padding explicit */
    WORD key;
    WORD cmd;
    WORD wPadding; /* making the padding explicit */
} ACCEL_RESOURCE;
```

Once again, the last entry is marked by setting the high bit of the `fVirt` member. The extra word of padding adds a second obstacle to taking the resource data and passing it to the `CreateAcceleratorTable` function to create the accelerator table manually. Not only do you have to strip off the high bit of the `fVirt`, you also have to convert the table to an array of `ACCEL` structures and pass the converted table to the `CreateAcceleratorTable` function.

That’s all there is to the format of accelerator table resources. I told you it was pretty simple.

Raymond Chen

Follow