The format of string resources

devblogs.microsoft.com/oldnewthing/20040130-00

January 30, 2004

Raymond Chen

Unlike the other resource formats, where the resource identifier is the same as the value listed in the *.rc file, string resources are packaged in “bundles”. There is a rather terse description of this in Knowledge Base article Q196774. Today we’re going to expand that terse description into actual code.

The strings listed in the *.rc file are grouped together in bundles of sixteen. So the first bundle contains strings 0 through 15, the second bundle contains strings 16 through 31, and so on. In general, bundle N contains strings (N-1)*16 through (N-1)*16+15.

The strings in each bundle are stored as **counted** UNICODE strings, not null-terminated strings. If there are gaps in the numbering, null strings are used. So for example if your string table had only strings 16 and 31, there would be one bundle (number 2), which consists of string 16, fourteen null strings, then string 31.

(Note that this means there is no way to tell the difference between “string 20 is a string that has length zero” and “string 20 doesn’t exist”.)

The LoadString function is rather limiting in a few ways:

- You can’t pass a language ID. If your resources are multilingual, you can’t load strings from a nondefault language.
- You can’t query the length of a resource string.

Let’s write some functions that remove these limitations.

```
LPCWSTR FindStringResourceEx(HINSTANCE hinst,
 UINT uId, UINT langId)
{
 // Convert the string ID into a bundle number
 LPCWSTR pwsz = NULL;
 HRSRC hrsrc = FindResourceEx(hinst, RT_STRING,
                     MAKEINTRESOURCE(uId / 16 + 1),
                     langId);
 if (hrsrc) {
  HGLOBAL hglob = LoadResource(hinst, hrsrc);
  if (hglob) {
   pwsz = reinterpret_cast<LPCWSTR>
              (LockResource(hglob));
   if (pwsz) {
    // okay now walk the string table
    for (int i = 0; i < uId & 15; i++) {
     pwsz += 1 + (UINT)*pwsz;
    }
    UnlockResource(pwsz);
   }
   FreeResource(hglob);
  }
 }
 return pwsz;
}
```

After converting the string ID into a bundle number, we find the bundle, load it, and lock it. (That's an awful lot of paperwork just to access a resource. It's a throwback to the Windows 3.1 way of managing resources; more on that in a future entry.)

We then walk through the table skipping over the desired number of strings until we find the one we want. The first WCHAR in each string entry is the length of the string, so adding 1 skips over the count and adding the count skips over the string.

When we finish walking, pwsz is left pointing to the counted string.

With this basic function we can create fancier functions.

The function FindStringResource is a simple wrapper that searches for the string in the default thread language.

```
LPCWSTR FindStringResource(HINSTANCE hinst, UINT uId)
{
 return FindStringResourceEx(hinst, uId,
     MAKELANGID(LANG_NEUTRAL, SUBLANG_NEUTRAL));
}
```

The function GetResourceStringLengthEx returns the length of the corresponding string, including the null terminator.

```
UINT GetStringResourceLengthEx(HINSTANCE hinst,
 UINT uId, UINT langId)
{
 LPCWSTR pwsz = FindStringResourceEx
                       (hinst, uId, langId);
 return 1 + (pwsz ? *pwsz : 0);
}
```

And the function AllocStringFromResourceEx loads the entire string resource into a heap-allocated memory block.

```
LPWSTR AllocStringFromResourceEx(HINSTANCE hinst,
 UINT uId, UINT langId)
{
 LPCWSTR pwszRes = FindStringResourceEx
                       (hinst, uId, langId);
 if (!pwszRes) pwszRes = L"";
 LPWSTR pwsz = new WCHAR[(UINT)*pwszRes+1];
 if (pwsz) {
   pwsz[(UINT)*pwszRes] = L'\0';
   CopyMemory(pwsz, pwszRes+1,
              *pwszRes * sizeof(WCHAR));
 }
 return pwsz;
}
```

(Writing the non-Ex functions GetStringResourceLength and AllocStringFromResource is left as an exercise.)

Note that we must explicitly null-terminate the string since the string in the resource is not null-terminated. Note also that the string returned by AllocStringFromResourceEx must be freed with delete[]. For example:

```
LPWSTR pwsz = AllocStringFromResource(hinst, uId);
if (pwsz) {
  ... use pwsz ...
  delete[] pwsz;
}
```

Mismatching vector “new[]” and scalar “delete” is an error I’ll talk about in a future entry.

Exercise: Discuss how the /n flag to rc.exe affects these functions.

Raymond Chen

Follow