

## Re: CListCtrl unicode doesn't display korean characters correctly

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2007-04/msg00360.html>

---

- *From:* Dansk <dansk@xxxxxxxxxxxxxx>
  - *Date:* Wed, 04 Apr 2007 13:54:08 +0200
- 

Thank you *\*very\** much for your time and deep analysis.

Even if my problem is not yet solved, lots of your comments make sense (and made me change my code).

Here are my coments, remarks and questions:

– Your assumptions are ok about the not described types, Strings are CString, not arrays of TCHARs

– The prototype for LoadString is  
bool LoadString(CString &String, DWORD StringId);  
I have not written it, I have to trust it.

– strsafe.h is not in my VC6 installation, Where can I find it? I'll try to include it, but do you *\*really\** think it could change something? In which way is it safe?

– pItem->cchTextMax : I added a check on that value, and replaced the hard-coded 256 by the use of that value. That way, the code looks better, but does not change anything.

– You are right, there is a bug in my test  
if(iItemIndx<=\_DisplayedStringElements.GetSize() && \_DisplayedStringElements[iItemIndx])

it is much better like  
if(iItemIndx<\_DisplayedStringElements.GetSize())

– The font. The app is for internal use, so, the deployment is not really a problem. I do have "Arial Unicode MS" on my system, and a step by step execution showed me that it is used. It is a font from Microsoft. If "Arial" is used, then many characters in every languages are missing, I can see it immediatly.

I don't think the font itself is a problem as I use it on the CEdit which works perfectly, and the tooltips on the ListCtrl are displayed correctrly too.

Again, thanks for your time.

Dansk

Re: CListCtrl unicode doesn't display korean characters correctly

MrAsm a écrit :

On Wed, 04 Apr 2007 11:14:34 +0200, Dansk <dansk@xxxxxxxxxxxxxx> wrote:

Here is the code that gives the string to the ListCtrl  
The korean string is in case 2.  
The LoadString method gets it from the resources.

From your code, I take that you have an array  
\_DisplayedStringElements, that stores the data you display to the list  
control. This array seems to store pointers to a data structure.  
It would be interesting to see how this is defined.  
Reading your code, it seems that it stores fields like \_NumberStr  
(string), \_ID (string), \_Value (string), \_Number (unsigned long).

For example, I hope the string fields are defined as CString and not  
raw arrays of TCHARs (or 'char's).

More below:

```
////////////////////////////////////  
void CMyDlg::OnGetdispinfoList(NMHDR* pNMHDR, LRESULT*  
pResult)  
{  
LV_DISPINFO* pDispInfo = (LV_DISPINFO*)pNMHDR;  
LV_ITEM* pItem = &(pDispInfo->item);  
  
int iItemIdx = pItem->iItem;  
ASSERT(iItemIdx >= 0 &&  
iItemIdx < _DisplayedStringElements.GetSize());
```

\*\*\*

OK with this check, but you kind of duplicated it also in case 2.

```
if (pItem->mask & LVIF_TEXT) //valid text buffer?  
{  
switch(pItem->iSubItem){  
case 0: //fill in main text  
lstrcpy(pItem->pszText,  
_DisplayedStringElements[iItemIdx]->_NumberStr);
```

\*\*\*

## Re: CListCtrl unicode doesn't display korean characters correctly

This code is not robust, and it may be broken by buffer overruns.  
You should forget about functions like `lstrcpy` & friends.  
You should only use \*safe\* string functions, like the functions from  
Safe String Library (`strsafe.h`).

In this case, I would use `StringCchCopy`. It protects you from buffer  
overruns.

```
StringCchCopy( pItem->pszText, // destination  
pItem->cchTextMax // destination buffer size in TCHARs  
_DisplayStringElements[iItemIdx]->_NumberStr // source  
);
```

From the description of `LVITEM` structure here:

<http://msdn2.microsoft.com/en-us/library/ms670569.aspx>

you can read:

<cite>

NOTE: Never copy more than `cchTextMax` TCHARs where `cchTextMax`  
includes the terminating NULL into `pszText` during an `LVN_`  
notification, otherwise your program can fail.

</cite>

And you are processing an `LVN_GETDISPINFO`.

```
case 1: //fill in sub item 1 text  
lstrcpy(pItem->pszText,  
_DisplayedStringElements[iItemIdx]->_ID);
```

\*\*\*

Again, you should not use `lstrcpy`.

```
case 2: //fill in sub item 2 text  
{  
if(iItemIdx<=_DisplayedStringElements.GetSize() &&  
_DisplayedStringElements[iItemIdx])
```

\*\*\*

I don't understand this.

You first checked `iItemIdx` at the beginning of the function.

So, why are you checking again?

Moreover, it seems to me that this second check has a bug: you should  
substitute the `<=` with `<`

## Re: CListCtrl unicode doesn't display korean characters correctly

If array size is 10, only valid indexes are 0,1,2,3,...,9 (not 10! 10 is out-of-bounds).

Moreover, why do you put the "&& \_DisplayedStringElements[iItemIdx]" ??

Do you want to check for non-NULL pointers?

Maybe you should do:

```
ASSERT( _DisplayedStringElements[iItemIdx] != NULL );
```

```
{
  CString *Value = &_DisplayedStringElements[iItemIdx]->_Value;
  if (Value->IsEmpty())
  {
    if (!LoadString(*Value, _DisplayedStringElements[iItemIdx]->_Number))
```

\*\*\*

I understand that this LoadString is neither Win32 ::LoadString, nor you are using CString::LoadString...

So, there could be a bug in this function.

Could you post the code of your LoadString function?

Or at least the prototype...

You are passing a CString (not CString \*) to LoadString.

Is the first LoadString parameter a \*reference\* to CString (CString &)? Or is it just a CString? This could be a bug...

```
try {
  if(Value->GetLength()>256)
    lstrcpy(pItem->pszText, Value->Left(256)+_T("..."));
  else
    lstrcpy(pItem->pszText, *Value);
} catch(...) {
  lstrcpy(pItem->pszText, _T("--> Unexpected error <--"));
}
```

\*\*\*

Why this try/catch block here??

I see no need for it...

Also, I have suspects about the operator+ you used to concatenate strings.

I would make the code more explicit and don't trust C++ copy constructors, temporary objects, automatic casts, etc....

## Re: CListCtrl unicode doesn't display korean characters correctly

Something like this:

```
if ( Value->GetLength() > 256 )
{
  CString s = Value->Left(256);
  s += _T("...");

  StringCchCopy(
  pItem->pszText,
  pItem->cchTextMax,
  (LPCTSTR)s
  );
}
else
{
  StringCchCopy(
  pItem->pszText,
  pItem->cchTextMax,
  (LPCTSTR)(*Value)
  );
}
```

And this is how I initialize my ListCtrl named m\_List

```
// ListCtrl Font
CFont *ListFont = m_List.GetFont();
if(ListFont)
{
```

\*\*\*

I would not use the if, just an ASSERT().  
ASSERT( ListFont != NULL );

because I think that GetFont() is supposed to return a valid font...

```
LOGFONT LogFont;
ListFont->GetLogFont(&LogFont);
lstrcpy(LogFont.lfFaceName, _T("Arial Unicode MS"));
```

\*\*\*

Again, better avoiding lstrcpy...

```
StringCbCopy( LogFont.lfFaceName, sizeof(LogFont.lfFaceName),
_T("Arial Unicode MS")
```

Re: CListCtrl unicode doesn't display korean characters correctly

);

However, I don't understand very well your code...

You're trying to load the font "Arial Unicode MS" (I don't have this on my system... is this a custom font?? Do you have this font on your system or the system where the program will be deployed?).

If this fails, you're loading Arial.

Are the Korean characters available for Arial?

Or do exist Korean-specific fonts?

Maybe "MS Mincho" font contains Korean characters?

Or you can use this fonts from here:

<http://babel.uoregon.edu/yamada/fonts/korean.html>

MrAsm