Subject: EditString: how to convert data with "~" operator to char*?
Posted by Cpu86 on Wed, 01 Mar 2006 22:31:52 GMT
View Forum Message <> Reply to Message

Hello,

how can I convert the data stored in an editstring from the type returned by applying the ~ operator to char *.
I want to manipulate the value of the editstring as same as an array of char.
It is possible?

Thanks

Marco

Subject: Re: Editstring control
Posted by zsolt on Wed, 01 Mar 2006 22:36:28 GMT
View Forum Message <> Reply to Message

String s = edit.GetData();
const char *cs = ~s;

Subject: Re: Editstring control
Posted by mirek on Wed, 01 Mar 2006 22:55:34 GMT
View Forum Message <> Reply to Message

zsolt wrote on Wed, 01 March 2006 17:36String s = edit.GetData();
const char *cs = ~s;

BTW, instead of .GetData you can use ~ here too, and you do not need '~' for converting as there is operator const char*:

String s = ~edit;
const char *cs = s;

String::operator~ is reserved just for ambiguous cases like

printf("%s", ~s);

Mirek

Subject: Re: Editstring control

Posted by Cpu86 on Wed, 01 Mar 2006 22:59:41 GMT
View Forum Message <> Reply to Message

Ok, but I must declare the pointer as a const?

Subject: Re: Editstring control
Posted by mirek on Wed, 01 Mar 2006 23:29:43 GMT
View Forum Message <> Reply to Message

Cpu86 wrote on Wed, 01 March 2006 17:59Ok, but I must declare the pointer as a const?

I see...

Yes. Constness is needed to give enough freedom for COW String implementations.

If you want to write those chars, you could use StringBuffer:

String x...

StringBuffer xb(x); //Clears x, takes the content

char *x = xb;

// do what you want, investigate StringBuffer interface to learn more

x = xb; // put it back, clears xb

Of course, another option is to simply copy String to separate buffer, however StringBuffer guarantees minimal copying.. (if possible, it performs none).

Mirek

Subject: Re: Editstring control
Posted by malaugh on Fri, 22 Jun 2007 15:56:29 GMT
View Forum Message <> Reply to Message

I have been struggling with the same issue. I need to get a hex number from an edit box, so I used an EditField called editHex, and converted to a string using

```
char HexDigits[MAX_HEX_CHARACTERS + 1];
UINT8 DataOut[MAX_HEX_CHARACTERS/2 ];

strcpy(HexDigits, editHex.GetText().ToString());
GetHexValue(DataOut, HexDigits);
```

GetHexValue reads the string and does the conversion.

This works fine if I just execute the program. But if run in the debugger and set a breakpoint on the strcpy line, I get the Microsoft "crash" dialog box with the message.

"gdb.exe has encountered a problem and needs to close. We are sorry for the inconvenience."

If I comment out the strcpy, then gdb does not crash. I am using the windows version with the minGW compiler and the internal debugger. The EditField length is set to MAX_HEX_CHARACTERS.

Am I doing something wrong?

Subject: Re: Editstring control
Posted by mirek on Sat, 23 Jun 2007 06:06:58 GMT
View Forum Message <> Reply to Message

Most likely nothing (means your code is OK).

Unfortunately, there are bugs in gdb...

Well, while the bug you mention is caused by gdb, you code might have serious buffer overflow problem: Imagine what happens when you enter more than "MAX_HEX_CHARACTERS".

Anyway, I recommend you to investigate EditField::SetConvert, e.g.
look here:

http://www.ultimatepp.org/reference$Convert.html

Subject: Re: Editstring control
Posted by malaugh on Sun, 24 Jun 2007 21:16:05 GMT
View Forum Message <> Reply to Message

Fantastic, thanks for your help. I tried your link and it works great. I was able to read out a hex value from the EditFiled after changing the code to do the hex conversion.

There is another issue though. How do I write to the EditField? I thought I could change the FormatIntBase function to base 16, but I ran into problems. Here is part of my code that I was using to debug the problem.

```
void HexEdit::OnPopulate()
{
int x = 1234;
char s[20];
Value y = 1234;
Value z;

 sprintf(s,"%d",x);
 z = FormatIntBase(y, 2, 0, 0, 0);
 editOutput = s;
// editOutput = x;
// editOutput = y;
// editOutput = z;
}
```

editOutput is an EditField control. Here is what happens:

editOutput = s; works, it displays 1234
editOutput = x; crashes the program.
editOutput = y; crashes the program.
editOutput = z; works, it displays the binary conversion of 1234

I am not sure why y should fail and z should pass.

Also if I change the code by putting

editOutput.SetConvert(Single<ConvertBin>());

in the initialization (which should call FormatIntBase on a write to the control, and remove FormatIntBase from the populate function, the program crashes.

What am I doing wrong? Any suggestions?

Subject: Re: Editstring control
Posted by fudadmin on Mon, 25 Jun 2007 01:21:49 GMT
View Forum Message <> Reply to Message

If I remember well this is related to operator overloading.
My tip is always to try something like:

```
 editOutput<< ~x;
```

Subject: Re: Editstring control
Posted by mirek on Mon, 25 Jun 2007 06:52:30 GMT
View Forum Message <> Reply to Message

Use editOutput <<= value...

Mirek