

Re: Converting ascii to hex

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2006-01/msg02149.html>

- *From:* Joseph M. Newcomer <newcomer@xxxxxxxxxxxxx>
 - *Date:* Mon, 30 Jan 2006 10:28:46 -0500
-

I find the code to be incredibly convoluted to solve a simple problem. Even concepts such as 0x30 indicate incredibly bad programming style. Why such meaningless and unreadable hex constants?

goto statements? How quaint!

OK, I have a specification here, however vague.

I don't even know what is meant by a sentence that says "ciptmp is an ascii hex file which is a CString". Files are not CStrings. Did you perhaps mean that "ciptmp is a CString which has been read in from a file"?

For that matter, since you don't show how the string is read in, how do you know that the 8 seconds isn't spent reading in the file?

In addition, you are doing some really funky things, such as doing one-character string appends of the converted bytes. Not only is this outrageously expensive computationally, requiring 200000 allocations and copies, guaranteed to give you disastrous performance, but if the string '00' appears in the input, the concatenation `a1+= tmp` is guaranteed to produce erroneous output.

A CString is the worst possible selection of representation for the result.

Better choices of identifier names would help a lot.

```
CArray<BYTE, BYTE>result;
```

```
#define BAD ((BYTE)0xFF)
```

```
static const BYTE hextable[256] = {  
BAD, BAD, BAD, BAD, BAD, BAD, BAD, BAD, // 00..07  
BAD, BAD, BAD, BAD, BAD, BAD, BAD, BAD, // 08..0F  
BAD, BAD, BAD, BAD, BAD, BAD, BAD, BAD, // 10..17  
BAD, BAD, BAD, BAD, BAD, BAD, BAD, BAD, // 18..1F  
BAD, BAD, BAD, BAD, BAD, BAD, BAD, BAD, // 20..27  
BAD, BAD, BAD, BAD, BAD, BAD, BAD, BAD, // 28..2F  
0, 1, 2, 3, 4, 5, 6, 7, // 30..37  
8, 9, BAD, BAD, BAD, BAD, BAD, BAD // 38..3F  
BAD, 10, 11, 12, 13, 14, 15, BAD, // 40..47
```

Re: Converting ascii to hex

```
BAD, BAD, BAD, BAD, BAD, BAD, BAD, BAD, // 48..4F
BAD, BAD, BAD, BAD, BAD, BAD, BAD, BAD, // 50..57
BAD, BAD, BAD, BAD, BAD, BAD, BAD, BAD, // 58..5F
BAD, 10, 11, 12, 13, 14, 15, BAD, // 60..67
BAD, BAD, BAD, BAD, BAD, BAD, BAD, BAD, // 68..6F
BAD, BAD, BAD, BAD, BAD, BAD, BAD, BAD, // 70..77
BAD, BAD, BAD, BAD, BAD, BAD, BAD, BAD, // 78..7F
....repeat lines for 80..FF
```

```
BOOL Convert(const CString & input, CArray<BYTE, BYTE> & result)
{
#ifdef _UNICODE
ASSERT(FALSE); // does not work with Unicode without more effort
return FALSE;
#else
result.SetSize(input.GetLength() / 2); // allocate max length once
int inpos = 0;
int outpos = 0;

while(TRUE)
{ /* loop */
if(inpos >= input.GetLength())
break; // conversion done

BYTE ch1= hextable[(BYTE)input[inpos]];
if(ch1== BAD)
{ /* ignore it */
inpos++;
continue;
} /* ignore it */

inpos++;

if(inpos >= input.GetLength())
return FALSE; // second nybble missing, error

BYTE ch2 = hextable[(BYTE)input[inpos]];
ASSERT(ch2 != BAD); // hex digits must come in pairs!

if(ch2 == BAD)
return FALSE; // erroneous input string

result[outpos] = (ch1 << 4) | ch2;
inpos++;
outpos++;
} /* loop */
result.SetSize(outpos); // trim unused space
return TRUE;
#endif
}
```

Re: Converting ascii to hex

Re: Converting ascii to hex

This is about as fast as I can imagine doing it. The output is a byte array, not a CString, so NUL characters will make sense; there is a minimum of allocation and copy; there is no complicated set of tests based on meaningless hex values (if you want to say "if the value is less than the character 0" you would write `if(val < '0')`, not some pointless and unintelligible value like `0x30`, and should you find that you need to something as silly as a hex constant to represent a character, **AT LEAST PUT A MEANINGFUL COMMENT ON THE LINE**, such as one that explains what the mysterious hex constant means!), there is no complicated set of if-tests, it handles syntax errors on input (although not particularly gracefully, simply rejecting meaningless input strings), it computes results by bitshift, not multiply, **AND THERE IS NOT A SINGLE 'goto' ANYWHERE IN IT!**
joe

On Sun, 29 Jan 2006 18:59:50 GMT, "Me" <none@xxxxxx> wrote:

```
>I have a large ascii file which represents multiple sections of hex data.
>I have the following code which works fine but takes like 8 sseconds to
>process.
>Each block of data contains about 200000 bytes to process.
>I need to have it decode faster if possible.
>
> ciptmp is an ascii hex file which is a CString
> The ascii/hex file is all 8-bit bytes eg:ff ascii = 0x66 0x66
> total range of data is 00-ff
> a = current position in the string
> btmp is position of the new converted file
> bsize is calculated number of bytes to convert
>
> int tp;
> char asc;
> int btmp=0;
> CString a1; // final string of hex data
>
>loop:
> asc = ciptmp[a];
> if (asc < 0x30){ // if the data is less than
>0x30
> a++; // ignore it and go to next
>location
> goto loop;
> }
>
>// first convert the msd
> if ((asc > 0x2f) || (asc < 0x39)) // data is ascii 0-9
>convert to hex
> tp = asc-0x30;
> if (asc > 0x60) // convert ascii a-f to
>0x0a-0x0f
> tp = asc-0x57;
> tp = tp*16; // move into msd
>0x0a=0xa0...
```

Re: Converting ascii to hex

Re: Converting ascii to hex

```
>
>// then convert the lsd
> asc = ciptmp[a+1]; // second char is lsd
> if ((asc > 0x2f) && (asc < 0x39))
> tp = tp + (asc-0x30); // add to msd
> if (asc > 0x60)
> tp = tp + (asc-0x57); // convert a to 0x0a
> a1+=tp; // insert data into new
>data string
> a+=2; // inc position to ned 2
>bytes
> btmp++; // inc final string
>position
> if (btmp == bsize) // check if all data
>processed
> goto don;
> goto loop;
>
>don:
> finished converting and continues to additional code
>
>Is there a faster way of doing this??
>
Joseph M. Newcomer [MVP]
email: newcomer@xxxxxxxxxxxxx
Web: http://www.flounder.com
MVP Tips: http://www.flounder.com/mvp\_tips.htm
.
```

• **References:**

- ◆ **[Converting ascii to hex](#)**
◇ From: Me

- Prev by Date: **[Retrieving dimensions of a ListView](#)**
- Next by Date: **[Re: Terminating`](#)**
- Previous by thread: **[RE: Converting ascii to hex](#)**
- Next by thread: **[Help with user interaction with worker thread](#)**
- Index(es):
 - ◆ **[Date](#)**
 - ◆ **[Thread](#)**