How do you deal with NUL?

Asked 16 years, 2 months ago Modified 15 years, 5 months ago

Viewed 2k times

5

From time to time, I run into communications issue with other programmers, when we talk about NULL. Now NULL could be

a NULL pointer
the NUL character
an empty data element in some sort of database.

NUL seems to be the most confusing. It is the ASCII character 0x00.
I tend to use '\0' in my code to represent it. Some developers in my group
tend to prefer to simply use 0, and let the compiler implicitly cast it to a char.

What do you prefer to use for NUL? and why?

c++ c

Share

Improve this question

Follow

edited Oct 21, 2008 at 12:47

raven
18.1k ● 17 ● 82 ● 114

asked Oct 21, 2008 at 0:41

EvilTeach
28.8k ● 21 ● 88 ● 144

\0 is NUL, not NULL. Thus, I reverted to the OP's original revision. – C. K. Young Oct 21, 2008 at 0:54

1 Is your question about, clarity of conversation or clarity of code? Please clarify. – mxcl Oct 21, 2008 at 0:59

It's both really. Verbally, it doesn't really do much to set the context. In code I prefer to avoid the extra code that is caused by the implicit cast. I am interested in seeing how other developers deal with this issue, or at least get some other viewpoints on how to think about it. – EvilTeach Oct 21, 2008 at 1:49

What is the intent of each of these lines? char x = NULL;<br> char *x = NULL;<br> char *x = 0;<br> char x = 0;<br> char *x = '\0';<br> char x = '\0';<br> – EvilTeach Oct 21, 2008 at 1:51

They all sort of deposit the same nice binary pattern, but the intent isn't clear. NULL has a pointer connotation with it, while 0 has an integer connotation while '\0' is closer to being NUL. – EvilTeach Oct 21, 2008 at 1:55

15 Answers

Sorted by: Highest score (default)

10

I use `'\0'` for the nul-character and `NULL` for pointers because it is clearest in both cases.

BTW, both `0` and `'\0'` are `int`s in C and either one will be converted to `char` when stored in a `char` variable.

Share Improve this answer Follow

answered Oct 21, 2008 at 0:47
Robert Gamble
109k ● 25 ● 147 ● 138

humm. i was under the impression that a character in single quotes is a char in c. – EvilTeach Oct 21, 2008 at 1:43

In C++ a character constant with a single character is type char, in C a character constant is type int. – Robert Gamble Oct 21, 2008 at 1:54

0 is *not* `int` in C. It's a special integer literal which doesn't have type as such, but can be used in either `int` or pointer context (much like C# `null`). – Pavel Minaev Jul 23, 2009 at 9:10

@Pavel: 0 *is* an int in C which is a regular integer value like any other. When the literal 0 is converted to a pointer the result is a null pointer and because of that 0 or (void *)0 is called a null pointer constant but that does not change the fact that before converting it to a pointer 0 is an integer and after converting it to a pointer it is no longer an integer but a null pointer. You might want to review section 5 of the c-faq (c-faq.com/null/index.html). – Robert Gamble Jul 23, 2009 at 12:22

5

I like the pre-defined *NULL* macro, as it preserves the semantic meaning, rather than some other use of the number 0.

Share Improve this answer Follow

answered Oct 21, 2008 at 0:43

Matt J
45.1k ● 7 ● 51 ● 57

4

There are many English words which are spelled or spoken alike, yet which have different meanings. Like in English, use the context in which the discussion is taking place to guide you toward the intended meaning.

Share Improve this answer Follow

answered Oct 21, 2008 at 0:46

yfeldblum
65.4k ● 12 ● 131 ● 169

4

- For dealing with strings, I alwayse represent the null character as '\0'.
- For pointers, I try to use implicit-conversion-to-boolean (if (!myPtr) or if (myPtr)) for pointer nullity.
- If I need a default value for a pointer, it's NULL, e.g. struct list_head = { 0.0, NULL };).

END_OF_STRING is silly, since it's extra indirection that simply confuses new readers (anyone who doesn't immediately recognize '\0' should step away from the keyboard).

One other thing—I think the difference between a null value and an empty value is extremely important when talking about data modeling. This is especially true when discussing C-style strings or nullable database fields. There's a huge difference between someone telling you "I have no name" and "My name is ."

Share Improve this answer Follow

answered Oct 21, 2008 at 2:51

Tom Barta
1,284 ● 7 ● 3

1 And to increase clarity, I typically explicitly check against NULL when using an assignment statement as in an if-statement: if ((myPtr = GetData()) != NULL) { /* do work on myPtr */ } – Isak Savo Oct 22, 2008 at 8:14

3

@BKB:

I see the point in his advice, but "NULL" makes it clearer that the context is pointers. It's like using "0.0" for floating-point values, as '\0' when dealing with characters. (Likewise, I prefer seeing 0 if a char is being used in an arithmetic context.)

Bjarne further states in this FAQ that NULL is #defined as 0 anyway, so standard code shouldn't have a problem with it. I agree that the all-caps notation is ugly, but we'll have to wait until 0x (where nullptr will be available, as a keyword.)

Share Improve this answer Follow

answered Oct 21, 2008 at 0:54

aib
46.8k ● 10 ● 74 ● 79

3

If I remember correctly most C compilers define NULL like this:

```
#define NULL ((void*)0)
```

This is to ensure that NULL is interpreted as being a pointer type (in C). However this can cause issues in the much more type strict world of C++. Eg:

```
// Example taken from wikibooks.org
std::string * str = NULL; // Can't automatically cast
void (C::*pmf) () = &C::func;
if (pmf == NULL) {} // Can't automatically cast from v
member function.
```

Therefore in the current C++ standard null pointers should be initialized with the literal 0. Obviously because people are so used to using the NULL define I think a lot of C++ compilers either silently ignore the issue or redefine NULL to be 0 in C++ code. Eg:

```
#ifdef __cplusplus
#define NULL (0)
#else
```

```
#define NULL ((void*)0)
#endif
```

The C++x0 standard now defines a `nullptr` keyword to represent null pointers. Visual C++ 2005's CLI/C++ compiler also uses this keyword when setting managed pointers to null. In current compilers you can create a template to emulate this new keyword.

There is a much more detailed article on wikibooks.org discussing this issue.

Share Improve this answer Follow

edited Oct 22, 2008 at 8:10

answered Oct 22, 2008 at 8:02

orj
13.5k ● 14 ● 65 ● 73

NULL and 0 are equivalent in C++. You can use NULL to initialize pointers without a problem in both languages. Any C++ library that defines NULL in the old C-style is nonconforming. – Dan Olson Jul 23, 2009 at 9:20

3

```
A one-L NUL, it ends a string.
A two-L NULL points to no thing.
And I will bet a golden bull
That there is no three-L NULLL.



(The name of the original author is, alas, lost to the
```

Share Improve this answer

Follow

answered Jan 5, 2009 at 14:28

EvilTeach

28.8k ● 21 ● 88 ● 144

NULL for databases, NIL for code.

1

Share Improve this answer

Follow

answered Oct 21, 2008 at 0:43

JosephStyons

58.6k ● 64 ● 167 ● 237

While, on the whole, I would advice using named constants, this is one exception. To me, defining:

1

```
#define NULL 0
#define END_OF_STRING '\0'
```

makes as much sense as defining:

```
#define SEVEN 7
```

which is none. And yes, I am aware that NULL is already defined by the compiler, but I never use it. For pointers, 0; for chars, '\0'. Longer does not always mean more expressive.

Share Improve this answer Follow

answered Oct 21, 2008 at 10:59

Gorpik
11k ● 5 ● 38 ● 57

The difference as I see it is that "7" is a symbol which always means "seven" (as a C token, at least). '\0' is of course used as a string terminator, but that isn't always and inherently what a 0 byte means, so it's not synonymous with end-of-string, any more than 200 is synonymous with success.
– Steve Jessop Oct 21, 2008 at 14:39

That said, I don't see anything wrong with using '\0' as a literal, just because it sometimes is used for other things as well. But I also don't think that using a name is pointless, since there is a semantic difference even though the value of end-of-string will of course be 0 in all C programs.
– Steve Jessop Oct 21, 2008 at 14:41

I guess I was thinking more C++ than C here. \0 is end-of-string only for null-terminated strings. So I think it is much clearer than END_OF_STRING, which might not actually be and end-of-string. And END_OF_NULL_TERMINATED_STRING is getting things too far. – Gorpik Oct 22, 2008 at 8:16

Really Gorpik :) wouldn't that be END_OF_NUL_TERMINATED_STRING ? – EvilTeach Aug 3, 2019 at 16:49

1

I quite like

```
#define ASCII_NUL ('\0')
```

I only very occasionally mistype '\0' as '0'. But when I have done it, I've found the error very hard to spot by code inspection, with hilarious consequences. So I don't like '\0' much, and prefer ASCII_NUL or 0 (of course the latter has the wrong type in C++). Obviously I use '\0' where demanded by consistency with existing code, or style guides.

The Google C++ style guide, which contains a few things I like and a few I don't, but seems mostly sound, prefers NULL to 0 for pointers. It points out that NULL might not be defined simply as 0 (or 0L), especially in implementations where sizeof(void*) might not be sizeof(int) (or sizeof(long int)).

0 and NULL are both specified to be of integral type, and when converted to a pointer type they both must yield a null pointer value. But they aren't necessarily of the same integral type. So you might conceivably get some useful warnings or errors in some situations by using NULL.

Share Improve this answer Follow

edited Oct 21, 2008 at 15:49

answered Oct 21, 2008 at 10:34

Steve Jessop

279k ● 40 ● 469 ● 709

0

For communication I use NULL. If I'm working with a developer who cannot grasp the concept of NULL for different data-types then I'd be concerned.

For implementation it's case-specific. Numbers are 0 (post-fixed f for floating-point), pointers are NULL and character strings are 0.

Share Improve this answer Follow

answered Oct 21, 2008 at 1:06

Andrew Grant
58.8k ● 22 ● 131 ● 144

0

Systems that don't use binary 0 for NULL are getting harder to find. They also tend to have various portability issues. Why? Because on these systems neither memset nor calloc can clear out a struct that contains pointers correctly.

Share Improve this answer Follow

answered Oct 21, 2008 at 1:24

Joshua
43.1k ● 9 ● 78 ● 148

i certainly have never seen a NULL that didn't equate to zero. – EvilTeach Aug 19, 2009 at 0:25

0

```
const char END_OF_STRING = '\0';
```

So when you say:

```
str[i] = END_OF_STRING;
```

or

```
if (*ptr == END_OF_STRING)
```

there is absolutely no question what you mean.

Share Improve this answer

Follow

answered Oct 21, 2008 at 2:00

James Curran

103k ● 37 ● 185 ● 262

0

We use NULL for pointers and NULLCHAR for characters, using

```
#define NULLCHAR '\0'
```

Share Improve this answer

Follow

answered Oct 21, 2008 at 10:47

Graeme Perrow

57.2k ● 24 ● 86 ● 125

0

Sort of related: Slashdot recently had a story on the `comp.lang.c` FAQ section on null pointers, which I found quite interesting.

Share Improve this answer

Follow

answered Jul 23, 2009 at 9:00

Martin Geisler

73.7k ● 25 ● 173 ● 229