

Re: #include behavior

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2007-02/msg03763.html

- *From:* Yevgen Muntyan <muntyan.removethis@xxxxxxxx>
 - *Date:* Sat, 24 Feb 2007 03:28:03 GMT
-

Keith Thompson wrote:

Yevgen Muntyan <muntyan.removethis@xxxxxxxx> writes:

Keith Thompson wrote:

Beej Jorgensen <beej@xxxxxxxx> writes:

In article
<ltzxcabmx.fsf@xxxxxxxxxxxxxxxxxxxx>,
Keith Thompson <kst-u@xxxxxxxx> wrote:

7.1.2
paragraph
3:
If a file with
the same
name as one
of the above
< and >
delimited
sequences,
not
provided as
part of the
implementation,
is placed in
any of the
standard
places that
are searched
for included
source
files, the
behavior is
undefined.

Interesting. That says that if
I place a "stdlib.h" file in

Re: #include behavior

one of
the places searched for a
#include "stdlib.h"
directive but *not* for a
#include <stdlib.h>

Hmmm. That wasn't my read... Elaborate.

Why not for <stdlib.h>, especially since "" is
a "superset" (so to
speak) of <>?

[...]

Sorry, the scope of the "*not*" was unclear.

Suppose the compiler searches for <foo.h> in /usr/include,
and for

"foo.h" in \$HOME/include and then /usr/include (these are
just

arbitrary examples). 7.1.2p3, as I read it, says that if I place a
"stdlib.h" file in \$HOME/include (which, as I wrote above, is
one of

the places searched for "stdlib.h", but not one of the places
searched

for <stdlib.h>), then the behavior is undefined.

If I place a "stdlib.h" file in /usr/include, of course, it also
causes UB.

Given that the whole include business is implementation-defined,
7.1.2.3 is rather just an additional "don't mess with implementation"
hint. Especially given that the standard doesn't require the very
possibility to place a source file into some place. From the other
hand, programs having "string.h" header do exist, even the alive
supported ones.

But I think it goes beyond "don't mess with the implementation".

I missed the fact that UB is quite different for an implementation
from what it is for a user. I was thinking rather about writing
portable code, which means not abusing implementations and not using
standard header names (the "don't mess" part). But indeed UB is UB,
and it's not "just" something, it's undefined.

And I just noticed that 7.1.2p3 talks about "a file with the
same
name", which ignores the mapping specified by 6.10.2. In
6.10.2,
it says that <foo.h> refers to
a header identified uniquely by the specified sequence

Re: #include behavior

Re: #include behavior

between
the
< and > delimiters
whereas "foo.h" refers to
the source file identified by the specified sequence between
the
"
delimiter
What does "identified by" mean here? Is "foo.h" actually the
name
of the file (as would be accepted by fopen(), or is it subject to
the
same mapping as <foo.h>?

It means the latter (not *the same* though), since the standard simply doesn't talk about files at all, i.e. it doesn't say source files should be "real" files on disk (you know, those bizarre implementations where there is no disk and files or which use database or streams or anything). fopen() is irrelevant since it's what happens on the target system in compiled program, not what compiler does.

The mapping may or may not be the same as for <foo.h> (and it isn't in practice with at least one popular compiler, and it must not be the same according to the common practice of "yours.h" and <system.h>).

You're right, fopen() is irrelevant; I was trying to use it to define what a file name is.

As for the "mapping", the way I tend to think of it is something like this: The compiler searches for include files and/or headers in one or more "places". You can (maybe) have two files with the same name in two different "places". The mapping mentioned in 6.10.2 maps the thing between <> or "" to a file name; the compiler searches for a file with that name in one or more "places". The mapping and the search path (set of "places") are two different things. (I'm not entirely sure my mental model is supported by the standard.)

I believe it's not like that. There are two mappings, one for "..."
and another one for <...>. If implementation uses files (i.e. usual case, for simplicity), then those two mappings map characters sequences to *absolute* file paths. In other words, the part about mappings in 6.10.2.5 is saying that all '#include "foo.h"' will use the same file in current translation unit (or in the whole program?), as well as all '#include <foo.h>' will do the same thing, or the preprocessing directive (every '#include "foo.h"', not just the second one, for instance) will cause failure to process the file.

It seems to me that 6.10.2.5 also could imply that you are allowed to have two different headers, foo.h and bar.h (they start with letter,

Re: #include behavior

end with .h, not more than eight characters, etc.) But the standard doesn't require the very possibility to have a custom header file, so it's way too vague.

But 7.10.2.3 doesn't ignore 6.10.2. It just uses human language, it says exactly what it says: "you put `stdlib.h` into one of those places and you get UB", even though it doesn't define what it means to put a source file into some place. It's not the only place where it talks humanish, isn't it?

The problem, I think, is that 7.10.2.3 is too expansive about which "places" need to be protected. To invent some terminology, let's say the compiler searches for `<foo.h>` in the "angle-bracket places", and for `"foo.h"` in the "quotation-mark places" followed by the "angle-bracket places". Certainly installing a file called `stdlib.h` in one of the angle-bracket places constitutes messing with the implementation, but installing such a file in one of the quotation-mark places should be ok. Logically, that shouldn't affect any program that has a `#include <stdlib.h>`. IMHO, it would make more sense for 7.10.2.3 to say only that putting `stdlib.h` in one of the angle-bracket places invokes UB.

I actually read it in exactly this way. I understood "the standard places that are searched for included source files" as "angle-bracket places". If it means something different, then I have no idea what this paragraph says.

Regards,
Yevgen

.

Re: #include behavior