

Re: what will be the value of #define

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2007-03/msg00394.html

- *From:* Keith Thompson <kst-u@xxxxxxx>
 - *Date:* Sat, 03 Mar 2007 02:23:07 -0800
-

"raghu" <ragavakumar@xxxxxxxxxx> writes:

On Mar 2, 6:28 pm, mark_blue...@xxxxxxxxxx wrote:

On 2 Mar, 13:22, Eric Sosman <esos...@xxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

raghu wrote:

Hello
I have a doubt plz clarify that
#ifndef SYSTEM_H
#define SYSTEM_H
what will be the value of SYATEM_H after
this #define statement and
before that statement.

[...]

(Assuming it's just a typo): Before the #define, SYSTEM_H is not defined and has no value at all. After it, SYSTEM_H is defined as an object-like macro whose replacement list is empty.

To explain a little more for "raghu" – the approach is a standard convention for ensuring that multiple inclusions of header files (usually due to being nested in other header files) are "harmless".

If my header called "fred.h" has the form:–

```
#ifndef FRED_H
#define FRED_H
...
#endif /*FRED_H*/
```

then the first inclusion will have the macro "FRED_H" undefined, will define the macro (as an empty macro) and do the other declarations, definitions etc which the header defines.

Re: what will be the value of #define

Any subsequent inclusions will have "FRED_H" defined, so will do nothing.

Yes you are correct. I am implementing exactly like this but one thing I didn't understand what will be the value assigned FRED_H here. when the processor looks in to it their must be change in the value. Right. what will be that value.

Thanks for ur reply.

Raghu, you've posted here a number of times. You've probably already been advised not to use silly abbreviations like "plz" and "ur". They just make what you write more difficult to read. If you want us to take the time to read your articles, take the time to spell out the words.

Given

```
#define SYSTEM_H
```

the identifier SYSTEM_H is a macro that expands to nothing. Any uses of it **outside a preprocessing directive** (such as #if or #ifdef) will simply vanish. For example, this:

```
SYSTEM_H int x SYSTEM_H = 42 SYSTEM_H ;
```

is exactly equivalent to this:

```
int x = 42 ;
```

But there's no reason to care what it expands to, because that's not what it's used for. In the common idiom where a macro is used as a header guard, the macro name is **only** used for the purpose of determining whether it's defined or not, using "#ifndef"; what it expands to is irrelevant.

—

Keith Thompson (The_Other_Keith) kst-u@xxxxxxx <<http://www.ghoti.net/~kst>>

San Diego Supercomputer Center <*> <<http://users.sdsc.edu/~kst>>

"We must do something. This is something. Therefore, we must do this."

— Antony Jay and Jonathan Lynn, "Yes Minister"

.