

## Re: printing % with printf(), use of \ (escape) character

**Source:** [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.c/2005-02/1623.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2005-02/1623.html)

---

*teachtiro\_at\_yahoo.com*

**Date:** 02/11/05

Date: 11 Feb 2005 02:22:24 -0800

Hi,

i got it through the below explanation

Eric Sosman wrote:

"Here's the fact you seem to be overlooking: printf() doesn't  
> *see the format string until after the compiler has processed it.*  
> *The compiler applies the same rules to the printf() format string*  
> *as it does to any other string literal: it processes \ escapes,*  
> *it splices adjacent literals together, and so on, doing exactly*  
> *the same thing for printf()'s format as for every other string*  
> *literal in the source code. printf() is not a special case."*

thank u

Eric Sosman wrote:

> *teachtiro@yahoo.com wrote:*  
> > *Hi Eric,*  
> > *Thanks for the response,*  
> > *i am sorry for a late reply,*  
> > *i am a newbie in the C world.*  
> > -----  
> > *Note that there's nothing special*  
> >  
> >> *about the tab character; printf() just copies it the same*  
> >> *way it copies H e l l o.) However, you need a way to tell*  
> >> *printf() to stop copying and do something else, like output*  
> >> *the characters that represent an integer value. So printf()*  
> >> *is sensitive to the % character, which means "stop copying*  
> >> *and do something different, as specified by the next few*  
> >> *characters (which also aren't copied)."*  
> >  
> > -----  
> > *can't that be implemented by using a different character with \,*  
for  
> > *e.g., they could have said to printf to consider \ as a special*  
> > *character and said, take \d as a sign of representing corresponding*

comp.lang.c: Re: printing % with printf(), use of \ (escape) character

- > > *variable as a decimal*
- > > *i.e. instead of saying \\d for the purpose, we could use \d or some \k*
- > > *(if d is already in use)*
- > > *the fact that printf() and c compiler work at different levels should*
- > > *make it more feasible*
- >
- > *printf() could have been defined to use \ instead of %.*
- > *Any character other than \0' could have been used: & or #*
- > *or Q or ' ' or : or anything you can imagine.*
- >
- > *However, if \ had been chosen there would have been a*
- > *problem. Let's suppose \ were the chosen character; here*
- > *are some output formats you might try to use:*
- >
- > *printf("\d\n", 1);*
- > *printf("\x\n", 2u);*
- > *printf("\07d\n", 3);*
- > *printf("\fn", 4.0);*
- >
- > *None of these would work as you want them to. The first*
- > *would cause the compiler to complain about an unknown escape*
- > *sequence; \d' is not a defined character escape. The second*
- > *would also provoke a complaint, because when the compiler sees*
- > *\x it expects to find some hexadecimal digits next, as in*
- > *"\x1b" -- the \x without its hex digits is an undefined escape*
- > *sequence.*
- >
- > *The final two would compile, but the results would disappoint*
- > *you. The third attempts to print a decimal number in a seven-*
- > *digit field with leading zeroes, but what it actually does is*
- > *print the two characters \07' and \n' without converting any*
- > *numbers at all -- on an ASCII system, this rings the bell and*
- > *starts a new line. The fourth is trying to convert a floating-*
- > *point number, but what it actually prints is a form feed followed*
- > *by a new line; nothing gets converted.*
- >
- > *All these problems are because \ in a string literal changes*
- > *the meaning of the next few characters. To prevent that from*
- > *happening and to get an actual \ into the string so printf()*
- > *could find it, you would need to write*
- >
- > *printf("\d\n", 1);*
- > *printf("\x\n", 2u);*
- > *printf("\07d\n", 3);*
- > *printf("\fn", 4.0);*
- >
- > *Here's the fact you seem to be overlooking: printf() doesn't*
- > *see the format string until after the compiler has processed it.*
- > *The compiler applies the same rules to the printf() format string*

comp.lang.c: Re: printing % with printf(), use of \ (escape) character

- > *as it does to any other string literal: it processes \ escapes,*
- > *it splices adjacent literals together, and so on, doing exactly*
- > *the same thing for printf()'s format as for every other string*
- > *literal in the source code. printf() is not a special case.*
- >
- > --
- > *Eric.Sosman@sun.com*