

Forum:[Hack Forums](#)

Thread: [Dealing with the wrong data type for input?](#)

God - July 14, 2017, 6:40 am

Hello,

So I've just started C and have been following thenewboston's tutorials.

PHP Code

```
<?php
#include <stdio.h>

int main(void){

    int age;
    char gender;

    printf("What is your age?:\n");
    scanf(" %d", &age);

    if (age >= 18){
        printf("You are allowed to access this website since you are %d years old, enjoy!\n", age);
        printf("What is your gender? (m/f):\n");
        scanf(" %c", &gender);
        if (gender == 'm' || gender == 'M'){
            printf("You're a male.");
        }
    }
    else if (age < 18){
        printf("You are not allowed to access this website since you are %d years old :(", age);
    }
    else{
        printf("Something went wrong!");
    }

}
```

Here, when I enter a letter for the age input, it counts it as a really high number. For example, if I put 'f':

PHP Code

```
<?php
You are allowed to access this website since you are 4194432 years old, enjoy!
What is your gender? (m/f):
```

Press any key to continue . . .

I also noticed it doesn't let me proceed to the next user input which is the gender. Can anyone explain this and suggest a fix?

Thanks.

Masm4ever - July 14, 2017, 6:57 am

scanf returns an error code:

```
Code if (scanf("%d", &age)) {
    printf("Age is %d\n", age);
} else {
    printf("Invalid input!\n");
}
```

God - July 14, 2017, 7:23 am

(Jul 14, 2017 06:57 AM)Masm4ever Wrote: [\[CITE\]](#) scanf returns an error code:

```
Code if (scanf("%d", &age)) {
    printf("Age is %d\n", age);
} else {
    printf("Invalid input!\n");
}
```

Thanks, this fixed the problem.

0xDEAD10CC - July 14, 2017, 4:05 pm

(Jul 14, 2017 06:57 AM)Masm4ever Wrote: [\[CITE\]](#) scanf returns an error code:

```
Code if (scanf("%d", &age)) {
    printf("Age is %d\n", age);
} else {
    printf("Invalid input!\n");
}
```

Btw, this is not truly the best way, and checking the return for any valid error response is worse than checking for the number of successfully parsed items because even if you want to check for EOF, you can check that in other ways using feof for instance. You should be comparing it against the number of format specifiers you have, checking it for boolean non-zero int is not great.

Consider:

```
Code if (scanf("%d %d", ...))
```

If 1/2 was parsed properly, this will still evaluate as being incorrectly "TRUE" for the logic of a successful call, because it wasn't successful.

For posterity, it should have been IMO:

```
Code if (scanf("%d", &age) == 1) {
```

You did not correctly explain the return value of scanf() so I can see this being a problem once he starts evaluating things like I've shown above.

Lastly, this is only the validation portion. If scanf() fails to parse data from the stdin buffer, it won't junk that data. You can't just loop a call to scanf() expecting it to only consider the next user input.

Masm4ever - July 14, 2017, 4:27 pm

(Jul 14, 2017 04:05 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Btw, this is not truly the best way, and checking the return for any valid error response is worse than checking for the number of successfully parsed items

With one format specifier, the return value can be 1 or 0; which is what my code checks. But I agree of course that the retval is not an "error code", that was sloppy. Apologies to OP.

0xDEAD10CC - July 14, 2017, 4:38 pm

The last thing I'd like to mention about "%d" is:

```
Code int main(void)
{
```

```
int x;  
printf("%d\n", scanf("%d", &x)); // enter 9999999999999999  
printf("%d\n", x);  
return 0;  
}
```

Overflows you'll have to deal with yourself.

In that case, `scanf()` will still return 1. The function was designed to extract only what it can use to construct a valid value using the specified format string. You can type "1234SDF" and to us that looks like an invalid number -- but to `scanf()` it see's the 1234 in that as a valid number and parses that, leaving the "SDF" in the STDIN stream buffer.