Use of goto to continue from several nested loops C

Asked 9 months ago Modified 5 months ago Viewed 200 times

4

I have a code in C that runs in an infinite while loop. Inside it, it has several nested loops performing some tasks. But at a certain point inside the inner for loop, for error handling purposes I'm forced to continue the execution on the next iteration of the outer while.

```
main_loop:
    while (1) {
        //Some functionalities...
        for (j = 0; j < num; j++) {
            for (k = 0; k < num; k++) {
                goto main_loop; // Continue the while loop
            }
        }
    }
```

If I write `continue;`, it continues the for iteration. I know I can add some flags to control the execution flow of all loops and modify those flags to implement the `continue` operation. But I finally decided to use a `goto` sentence to break the inner loops and continue to the desired iteration of while.

As `goto` can lead to spaghetti code situations, I'm wondering if this use-case of `goto` is sufficiently justified for being correct or is there any other specific way to continue the outer loop in similar codebases in C?

c loops goto

Share

Improve this question

Follow

edited Feb 24 at 9:18

Peter - Reinstate Monica

15.9k 4 40 67

asked Feb 24 at 9:01

Cardstdani

5,223 3 13 35

1 As an alternative to using `goto` you could extract the inner two loops into their own function and use `return`. – Konrad Rudolph Feb 24 at 9:11

Interestingly, because `while(1)` is essentially nothing but a jump destination itself, the `main_loop` label could as well be inside the loop, after the `while`!
– Peter - Reinstate Monica Feb 24 at 9:21

1 "As goto can lead to spaghetti code situations," And in situations like yours, it does the opposite! Even if it is matter of taste and goto is blamed always, I would vote to use it here!
– Klaus Feb 24 at 9:25

4 Answers

Sorted by: Highest score (default)

7

Your question actually contains it's *own* answer:

> As `goto` *can* lead to spaghetti code situations, ...

The main problem of things like `goto` is that they *may* lead to unreadable code, not that they always *do*. People who blindly refuse to use language features, because they don't understand this, are doing themselves a disservice.

Another example is is the fail-fast strategy where you check required conditions at the start of a function and simply return an error if they're not all met. The people who despise multiple return points in a function often get apoplectic when they see that, often a good reason to slip it into pull request in my opinion :-)

And yet, in that case and yours, the code they arrive at in order to follow the "rules" is often *less* readable.

Bottom line, there are certainly guidelines for a reason. But, unless you know why those reasons exist and apply sense to them, you're not really a artisan (I use that word because the same reasoning actually applies to a great many fields of endeavour).

Share
Improve this answer
Follow

edited Feb 24 at 10:19

answered Feb 24 at 9:10
paxdiablo
880k • 241 • 1.6k • 2k

5

This is the one situation where `goto` is completely warranted. Go for it.

Share Improve this answer Follow

answered Feb 24 at 9:03

wilx
18.2k • 7 • 64 • 120

2

For the reasons stated in the other answers, using `goto` is completely warranted for breaking out of a nested loop, or continuing an outer loop.

However, some programmers say that `goto` should generally only be used for jumping forwards, not backwards. I would say that this is a good guideline (but not a strict rule). If you want to comply with this guideline, then you way want to use the following code instead:

```
while (1) {
    // some additional code here
    for (j = 0; j < num; j++) {
        for (k = 0; k < num; k++) {
            goto continue_outer_loop;
        }
    }

continue_outer_loop:
    continue;
}
```

Share

Improve this answer

Follow

edited Jun 29 at 0:02

answered Feb 24 at 10:11

Andreas Wenzel
24.4k • 4 • 28 • 44

2

Spaghetti code can be written with-or-without invoking C's `goto`.

Spaghetti code can be come from not recognising the *flow* of control through the logic.

Think about this:

```
main_loop: // loop forever from here
    /* some stuff */
    for (j = 0; j < num; j++) {
        for (k = 0; k < num; k++) {
            /* more stuff */
            if( condition )
                goto main_loop; // bail. restart outer loop
        }
    }
    goto main_loop; // infinite loop
```

In the code fragment you've presented, the `while()` loop is superfluous and should be removed.

As this is *atypical*, use comments that assure the reader of the coder's intent.

Share Improve this answer Follow

answered Jun 29 at 5:39

Fe2O3

8,344 2 8 24