Proposal for C2y
Document Number: N3960

Title:	`#fatal` – immediate translation termination directive
Date:	2026-08-31
Author:	Joshua Crofts
Proposal category:	New features

Abstract: An `#error` directive will cause translation to fail; however, it does not stop translation unit processing immediately, leading to long lists of cascading error messages. This paper proposes a `#fatal` directive that immediately halts preprocessing and translation upon encounter and emits a user-defined diagnostic message.

`#fatal` – immediate translation termination directive

Document Number: N3960
Date: 2026-08-31
Reply-to: Joshua Crofts (joshua.crofts1@gmail.com)

1 Motivation

If an `#error` directive is encountered during the preprocessing stage, an error message is emitted, yet preprocessing still continues. This can lead to build failures, causing long lists of potential cascading error messages, which are not very helpful to the user.

Consider the following two files:

```
// header.h
#ifdef TARGET_ARCH_SUPPORTED
typedef struct { double data[16]; int flags; } Matrix;
#else
#error "Unsupported target architecture!"
#endif

void matrix_init(Matrix *m);

// main.c
#include "header.h"

Matrix g_m;

void setup(Matrix *m) {
        m->flags = 1;
        m->data[0] = 1.0;
}

int main(void) {
        setup(&g_m);
        matrix_init(&g_m);
        return 0;
}
```

Compiling `main.c` will cause the error "Unsupported target architecture!" to be emitted, yet translation will still continue, outputting more errors; in this case unknown type name and implicit declaration errors.

2 Proposal

This paper proposes a new `#fatal` directive; upon encounter, translation terminates, outputting a user-defined message, similar to `#error`.

Our aforementioned header file would look like:

```
// header.h
#ifdef TARGET_ARCH_SUPPORTED
```

```
typedef struct { double data[16]; int flags; } Matrix;
#else
#fatal "Unsupported target architecture!"
#endif

void matrix_init(Matrix *m);
```

Therefore, the compiler would only emit

```
In file included from main.c:1:
./header.h:4:2: fatal error: #fatal "Unsupported target architecture!"
    4 | #fatal "Unsupported target architecture!"
      |  ^~~~~
compilation terminated.
```

ending the translation.

3 Impact on Existing Code & Implementations

This proposal is backwards compatible; it does not introduce breaking changes. Existing valid C code is unaffected.

Implementation in existing preprocessors is trivial; an example implementation in GCC using `libcpp` is available [1].

4 Proposed wording

The proposed wording is a diff of the ISO/IEC 9899:2024 (en) — N3220 working draft [2]. Green text is newly added text.

Modify 6.10.1p1:

```
# line pp-tokens new-line
# error pp-tokens_opt new-line
# warning pp-tokens_opt new-line
# fatal pp-tokens_opt new-line
# pragma pp-tokens_opt new-line
# new-line
```

Modify 6.10.7p1:

A preprocessing directive of either form
error pp-tokens$_{opt}$ new-line
warning pp-tokens$_{opt}$ new-line
causes the implementation to produce a diagnostic message that includes the specified sequence of preprocessing tokens.

A preprocessing directive of form
fatal pp-tokens$_{opt}$ new-line
causes the implementation to terminate translation and produce a diagnostic message that includes the specified sequence of preprocessing tokens.

5 References

[1] cjd8/gcc,
https://github.com/cjd8/gcc/commit/4eabace69de7af982973791152bb2332e41175d3

[2] ISO/IEC 9899:2024 (en) — N3220 working draft,
https://www.open-std.org/jtc1/sc22/wg14/www/docs/n3220.pdf