# setjmp

Defined in header `<csetjmp>`

```
#define setjmp(env) /* implementation-defined */
```

Saves the current execution context into a variable `env` of type `std::jmp_buf`. This variable can later be used to restore the current execution context by `std::longjmp` function. That is, when a call to `std::longjmp` function is made, the execution continues at the particular call site that constructed the `std::jmp_buf` variable passed to `std::longjmp`. In that case **setjmp** returns the value passed to `std::longjmp`.

The invocation of `setjmp` must appear only in one of the following contexts:

- the entire controlling expression of `if`, `switch`, `while`, `do`-`while`, `for`.

```
switch(setjmp(env)) { ..
```

- one operand of a relational or equality operator with the other operand an integer constant expression, with the resulting expression being the entire controlling expression of `if`, `switch`, `while`, `do`-`while`, `for`.

```
if(setjmp(env) > 0) { ...
```

- the operand of a unary ! operator with the resulting expression being the entire controlling expression of `if`, `switch`, `while`, `do`-`while`, `for`.

```
while(!setjmp(env)) { ...
```

- the entire expression of an expression statement (possibly cast to `void`).

```
setjmp(env);
```

If `setjmp` appears in any other context, the behavior is undefined.

Upon return to the scope of `setjmp`, all accessible objects, floating-point status flags, and other components of the abstract machine have the same values as they had when `std::longjmp` was executed, except for the non-volatile local variables in the function containing the invocation of `setjmp`, whose values are indeterminate if they have been changed since the `setjmp` invocation.

## Parameters

**env** - variable to save the execution state of the program to.

## Return value

`0` if the macro was called by the original code and the execution context was saved to `env`.

Non-zero value if a non-local jump was just performed. The return value is the same as passed to `std::longjmp`.

## Notes

Above requirements forbid using return value of `setjmp` in data flow (e.g. to initialize or assign an object with it). The return value can only be either used in control flow or discarded.

## Example

Run this code

```
#include <iostream>
#include <csetjmp>

std::jmp_buf my_jump_buffer;

[[noreturn]] void foo(int count)
{
    std::cout << "foo(" << count << ") called\n";
    std::longjmp(my_jump_buffer, count+1);  // setjmp() will return count+1
}

int main()
{
    volatile int count = 0; // modified locals in setjmp scope must be volatile
    if (setjmp(my_jump_buffer) != 5) { // equality against constant expression in an if
        count = count + 1; // ++count, count += 1, etc on 'volatile'-qualified
                           // left operand are deprecated since C++20 (P1152)
        foo(count); // This will cause setjmp() to exit
    }
}
```

Output:

```
foo(1) called
foo(2) called
foo(3) called
foo(4) called
```

## See also

| | |
|---|---|
| **longjmp** | jumps to specified location (function) |

**C documentation** for **setjmp**

Retrieved from "https://en.cppreference.com/mwiki/index.php?title=cpp/utility/program/setjmp&oldid=136560"