goto statement ➤6.4

Normally, a function returns to the point at which it was called. We can't use a `goto` statement to make it go elsewhere, because a `goto` can jump only to a label within the same function. The `<setjmp.h>` header, however, makes it possible for one function to jump directly to another function without returning.

The most important items in `<setjmp.h>` are the `setjmp` macro and the `longjmp` function. `setjmp` "marks" a place in a program; `longjmp` can then be used to return to that place later. Although this powerful mechanism has a variety of potential applications, it's used primarily for error handling.

setjmp

To mark the target of a future jump, we call `setjmp`, passing it a variable of type `jmp_buf` (declared in `<setjmp.h>`). `setjmp` stores the current "environment" (including a pointer to the location of the `setjmp` itself) in the variable for later use in a call of `longjmp`; it then returns zero.

Q&A

longjmp

Returning to the point of the `setjmp` is done by calling `longjmp`, passing it the same `jmp_buf` variable that we passed to `setjmp`. After restoring the environment represented by the `jmp_buf` variable, `longjmp` will—here's where it gets tricky—*return from the* `setjmp` *call*. `setjmp`'s return value this time is `val`, the second argument to `longjmp`. (If `val` is 0, `setjmp` returns 1.)

Be sure that the argument to `longjmp` was previously initialized by a call of `setjmp`. It's also important that the function containing the original call of `setjmp` must not have returned prior to the call of `longjmp`. If either restriction is violated, calling `longjmp` results in undefined behavior. (The program will probably crash.)

To summarize, `setjmp` returns zero the first time it's called; later, `longjmp` transfers control back to the original call of `setjmp`, which this time returns a nonzero value. Got it? Perhaps we need an example...

PROGRAM

Testing `setjmp`/`longjmp`

The following program uses `setjmp` to mark a place in `main`; the function `f2` later returns to that place by calling `longjmp`.

tsetjmp.c

```
/* Tests setjmp/longjmp */

#include <setjmp.h>
#include <stdio.h>

jmp_buf env;

void f1(void);
void f2(void);

int main(void)
{
  if (setjmp(env) == 0)
    printf("setjmp returned 0\n");
```