Sharing Macro Definitions and Type Definitions

Most large programs contain macro definitions and type definitions that need to be shared by several source files (or, in the most extreme case, by *all* source files). These definitions should go into header files.

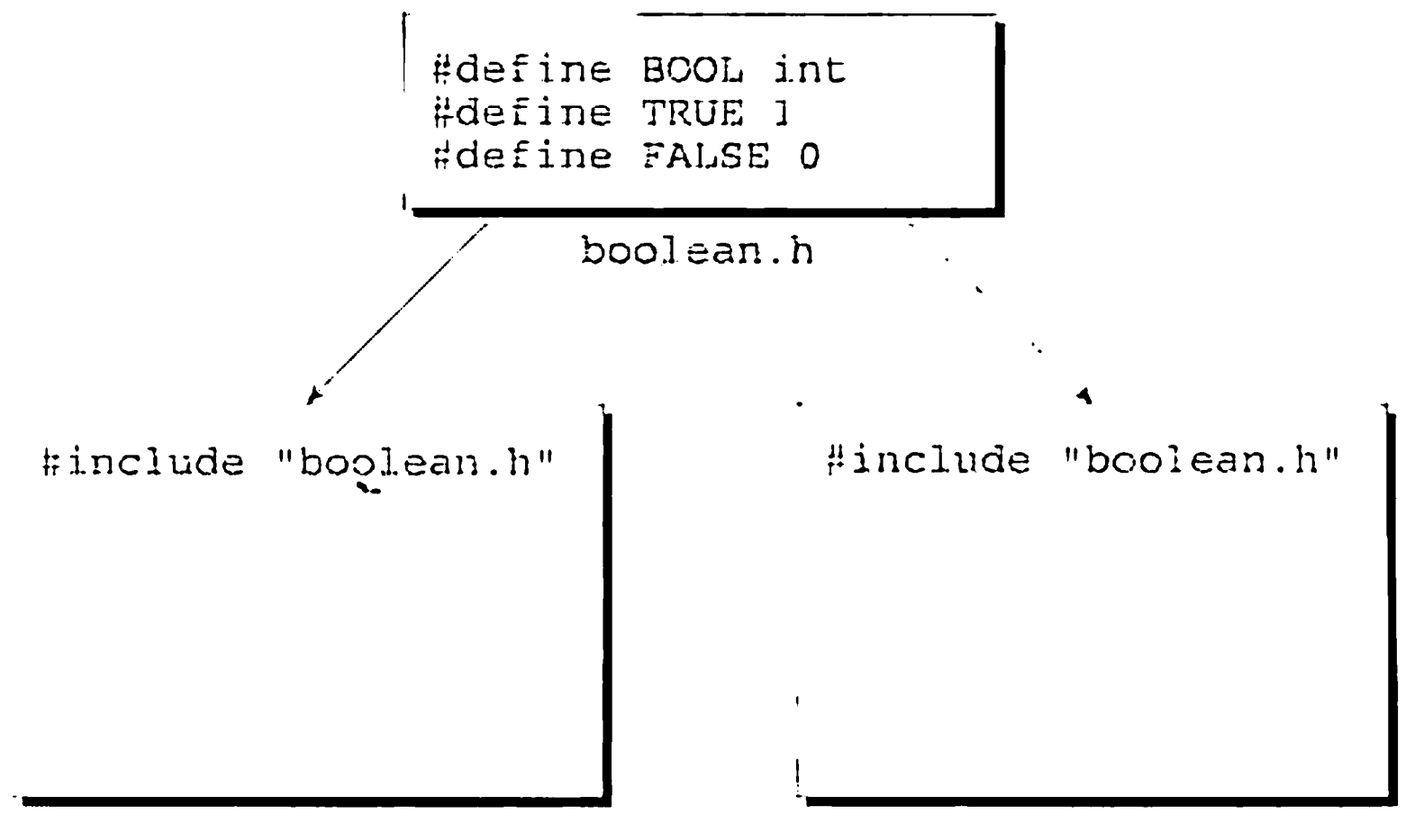
For example, suppose that we're writing a program that uses macros named `BOOL`, `TRUE`, and `FALSE`. (There's no need for these in C99, of course, because the `<stdbool.h>` header defines similar macros.) Instead of repeating the definitions of these macros in each source file that needs them, it makes more sense to put the definitions in a header file with a name like `boolean.h`:

```
#define BOOL int
#define TRUE 1
#define FALSE 0
```

Any source file that requires these macros will simply contain the line

```
#include "boolean.h"
```

In the following figure, two files include `boolean.h`:



Type definitions are also common in header files. For example, instead of defining a `BOOL` macro, we might use `typedef` to create a `Bool` type. If we do, the `boolean.h` file will have the following appearance:

```
#define TRUE 1
#define FALSE 0
typedef int Bool;
```

Putting definitions of macros and types in header files has some clear advantages. First, we save time by not having to copy the definitions into the source files where they're needed. Second, the program becomes easier to modify. Changing the definition of a macro or type requires only that we edit a single header file: we don't have to modify the many source files in which the macro or type is used. Third, we don't have to worry about inconsistencies caused by source files containing different definitions of the same macro or type.