

Multi-file C Programs

- Large C programs spread across many C files
e.g. Linux operating system has 50,000+ .c files.
- Files provide a *de facto* module system in C.
- C functions can be called from any file, unless **static**
Declaring functions **static**
 - ▶ avoids name clashes in huge programs
 - ▶ makes programs more readable/maintainable
- No checking of function parameters & return types between files
- By convention include .h files used to share information between .c files
ensure types match between files.

Include Files

- Include .h files contain:
 - ▶ function prototypes
 - ▶ type definitions
 - ▶ #define's
- .h files should not contain code (function definitions)
- #include with "" used to incorporate .h file
put #include at top of .c file

Example: Include File

answer.h

```
int answer(double x);
```

answer.c

```
#include "answer.h"  
int answer(double x) {  
    return x * 21;  
}
```

main.c

```
#include "answer.h"  
  
int main(void) {  
    printf("answer(2) = %d\n", answer(1));  
    return 0;  
}
```

Multi-file Compilation

```
$ gcc main.c answer.c -o answer  
$ ./answer  
42
```

Can also compile file separately creating .o files which contain machine code for one file.

```
$ gcc -c main.c  
$ gcc -c answer.c  
$ gcc main.o answer.o -o answer  
$ ./answer  
42
```

Useful with huge programs because faster to re-compile only part changed since last compilation.