# Working with Larger Programs

**Date:** 2013-06-05

A large program can be split into multiple smaller files containing logically related functions. This helps improve code organization.

Such multiple files belonging to the same program can be compiled together in the following fashion:

`gcc src1.c src2.c main.c -o program_name`

One thing to keep in mind here is that every file is compiled separately and it does not assume that these files are connected. If a file is using a function from a separate module then it must have the function declarations.

## Communication Between Modules

Whenever any global variable or any function is declared, it is actually accessible from within the other files.

If a file wishes to access a variable situated in another module, then it must declare that variable in itself preceded by keyword `extern`. For example, if module `a.c` has a variable name `lola` and `b.c` wishes to access that variable then the following line must be present in `b.c`.

```
extern int lola;
```

Other than declaring `lola` as a global variable, following line can be present in `a.c`:

```
extern int lola = 0; // initialization must be done with a value
```

For functions however, `extern` keyword is not required. Any function is by default accessible by any other module.

## `Static` versus `Extern` Variables and Functions

Now we know that any global variable and any function is by default accessible by other modules. This accessibility can be turned off using `static` keyword. If any (global) variable or function is preceded by `static` keyword, then it is not accessible from any other module.

## Using Header Files

All the commonly used definitions can be defined inside a header files and then that header file can be imported. Benefit of using a header files is that there won't be a need to repeat those definitions again and again in different modules which can be error prone as well. Also, if any definition needs to be updated, then the header file only needs to be edited.