Programs with more than one file

So, now you know how to compile a program within one single file, but most of the real world programs are made of several files, tons of them actually and it is necessary to compile every file one by one and then put all of them together through a step called **linking**, here is an easy example with just two files.

File 1 **print.c**:

```
#include <stdio.h>

void printHola( void )
{
    printf("Hola mundo\n");
}
```

File 2 **main.c:**

```
extern void printHola( void );

int main( void )
{
    printHola();
    return 0;
}
```

Let's compile both files using GCC and the option **-c**, we gonna generate a couple of files with the same name and termination **.o**, just like this

```
$ gcc -c print.c -o print.o
$ gcc -c main.c -o main.o
```

and then the linking step to finally have our program ready to run

```
$ gcc print.o main.o -o hola
$ ./hola
Hola mundo
```

It is usual that for every file (*excluding the one with the main function*) there is a header file to place all the interfaces some other file will use

File 1 **print.c**: (remains the same)

```
#include <stdio.h>
#include "print.h"

void printHola( void )
{
```

```
    printf("Hola mundo\n");
}
```

File 2 **print.h** (new file)

```
#ifndef PRINT_H_
#define PRINT_H_

void printHola( void );

#endif
```

File 3 **main.c** (remove reference to printHola function and add include to print.h file)

```
#include "print.h"

int main( void )
{
    printHola();
    return 0;
}
```

The build process can remain the same

```
$ gcc -c print.c -o print.o
$ gcc -c main.c -o main.o
$ gcc print.o main.o -o hola
$ ./hola
Hola mundo
```