**Programming Tools - Compilation and linking Guided Assignments**

***Getting* started *with gcc*:**

1. **Log into the Linux server using your respective ids**
2. **Copy the C source file simple\_program.c. Compile the file with the following command:**

**gcc *sourcefilename.c***

**Where *sourcefilename.c* is the name of the C source code**

1. Output:

A screen shot of a computer code

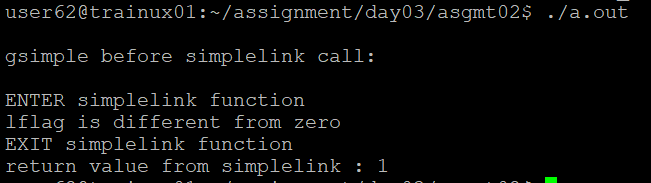
Description automatically generated

1. **Observe the executable formed is called a.out. Execute the file using the command:**

**./a.out**

**(Here ./ implies that the path of a.out is the current directory)**

1. Output:

****

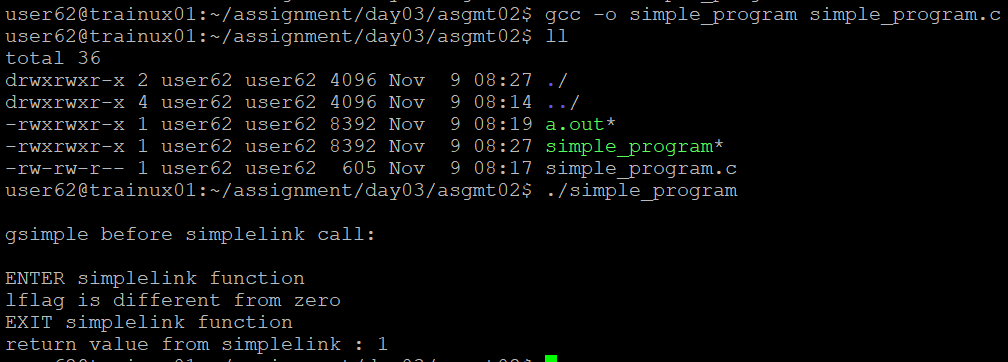
1. **Once again compile the source code with the following command:**

**gcc –o outputfilename sourcefilename.c**

**Where *outputfilename* is the name of the executable file. After this command gets executed successfully, the name of the executable is not a.out but whatever is given as *outputfilename***

**Execute the output file using the command: ./outputfilename**

1. Output:

****

## **Object files:**

1. **Once again compile the source code with the following command:**

**gcc –c *sourcefilename.c***

**Observe using ls that an object file called *sourcefilename.o* is created in the directory**

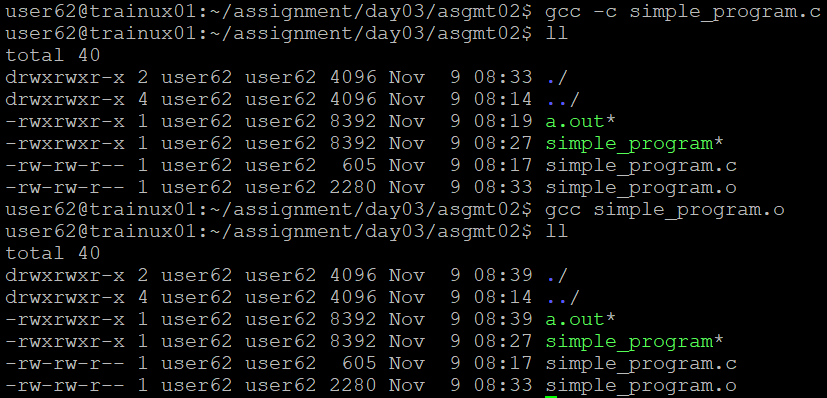
**What is the difference between an executable and an object file?**

An object file is compiled, intermediate file containing machine code, while an executable file is a fully linked, runnable program ready for execution.

**Now, create the executable from the object file using the following command:**

**gcc *sourcefilename.o***

1. Output:



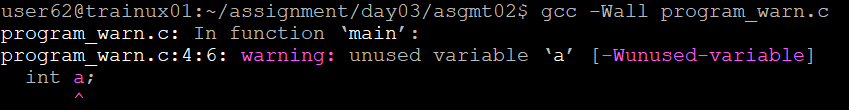
## **Additional gcc switches:**

1. **Copy the program program\_warn.c as directed by the facilitator. Compile the program using gcc without any switch. Observe the result of compilation**
2. Output:

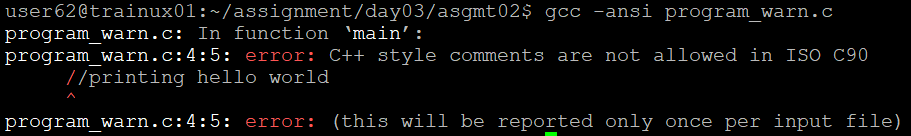
A screenshot of a computer screen

Description automatically generated

1. **Now, compile the program with the –Wall switch. Observe the warnings that are now given by the compiler.**
2. Output:



1. **Now, compile the program with the –ansi switch. Observe the warnings that are now given by the compiler.**
2. Output:



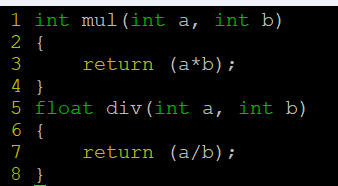
**Linking**

1. **Write a program linking1.c, which has main function. Inside main there should be a call to another function which is defined in another file linking2.c**
2. Output: linking1.c

A computer screen with red and white text

Description automatically generated

linking2.c



**10. Compile the file linking1.c using the following command:**

**gcc –c –Wall linking1.c**

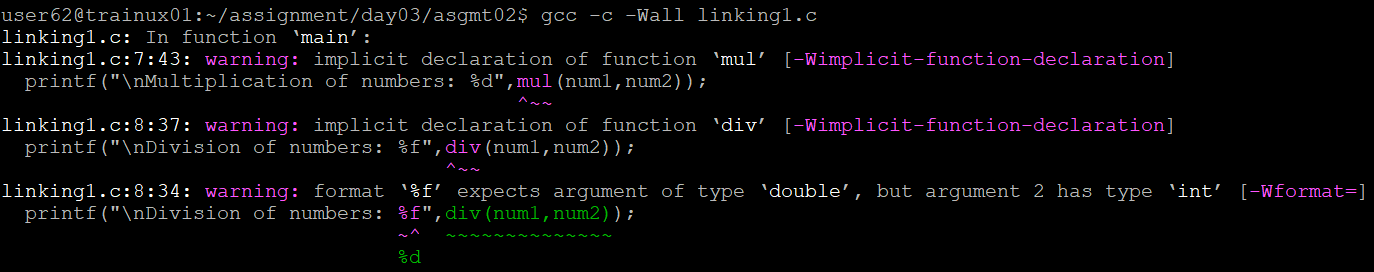
**Observe the warnings received**

**Again compile the file linking1.c using the following command:**

**gcc –Wall linking1.c**

**Observe the errors received**

1. Output:

****

**A computer screen with text on it

Description automatically generated**

1. **Now create a header file prototype.h, which contains the prototype of the function defined in linking2.c. Let the file content begin and end with lines as below, to avoid multiple inclusion of .h file.**

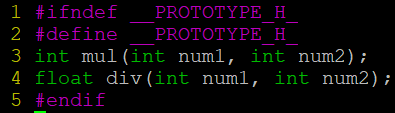
**#ifndef \_\_PROTOTYPE\_H\_**

**#define \_\_PROTOTYPE\_H\_**

**void display(); // an example declaration of function defined in prototype.h**

**#endif //end of \_\_PROTOTYPE\_H\_**

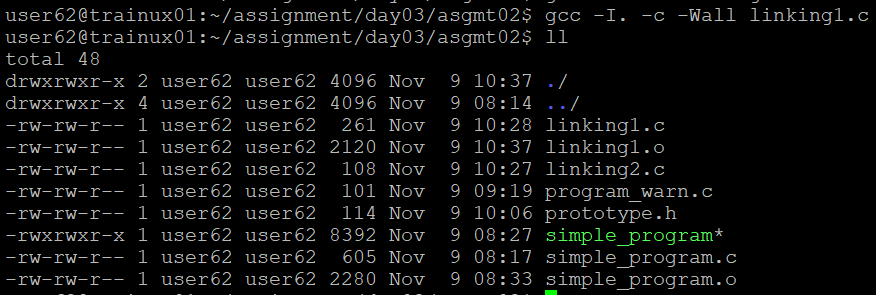
1. Output: prototype.h



1. **Include this file in linking1.c using the following statement:**

**#include <prototype.h> and again compile linking1.c using the following command: gcc –I. –c –Wall linking1.c**

1. Ouput:



1. **Similar effect can be achieved by including the file prototype.h using the following command**

**#include “prototype.h” and then compiling using the following command gcc –c –Wall linking1.c**

1. Output:

A screen shot of a computer

Description automatically generated

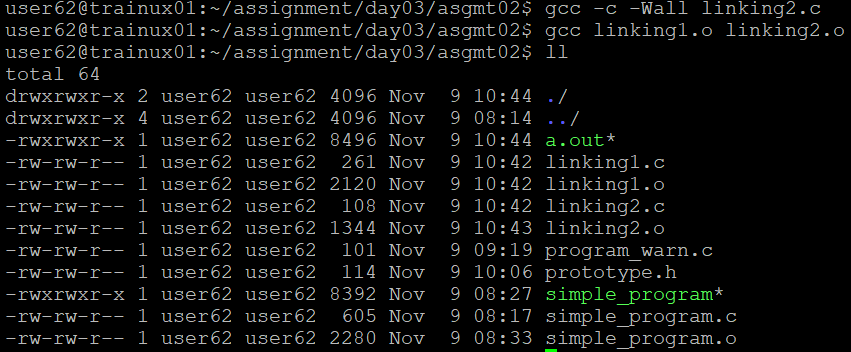
1. **Now after all the compilation warnings have been removed, link the 2 files together:**

**gcc –c –Wall linking1.c**

**gcc –c –Wall linking2.c**

**gcc linking1.o linking2.o**

1. Output:



1. **Do not include the header file prototype.h containing the prototype of the function in the file linking1.c.**

**Instead of that include the file linking2.c in the file linking1.c using the following statement:**

**#include “linking2.c”**

**Now repeat the following three commands:**

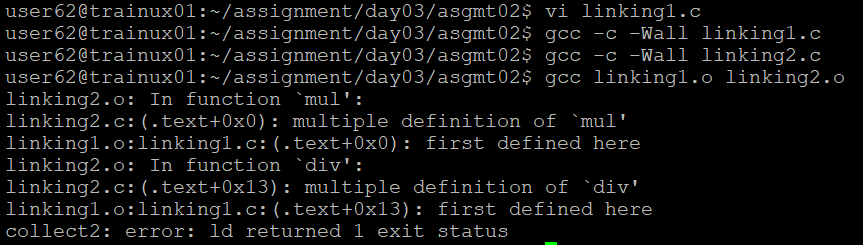
**gcc –c –Wall linking1.c**

**gcc –c –Wall linking2.c**

**gcc linking1.o linking2.o**

**Observe the error received.**

1. Output:



1. **Do not include the file linking2.c in linking1.c. Instead of that include the file prototype.h and proceed as mentioned in the points 15 and 16.**
2. Output:

A screen shot of a computer

Description automatically generated