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

Getting started with gcc

1. Log into the Linux server using your respective ids

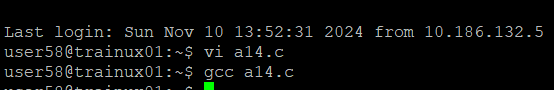
A screenshot of a computer screen

Description automatically generated

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



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

./a.out

A screenshot of a computer

Description automatically generated

4. 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

A screenshot of a computer program

Description automatically generated

Object files

5. 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?

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

gcc sourcefilename.o

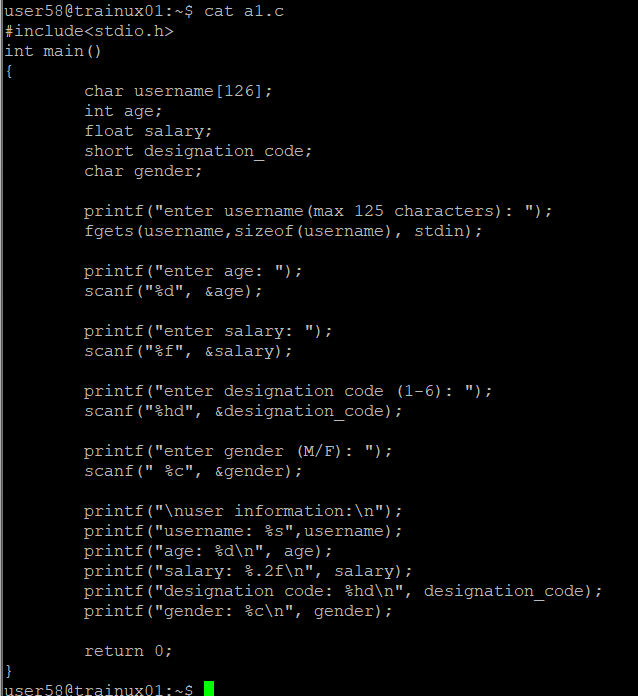
|  |  |
| --- | --- |
| Executable Files | Object Files |
| This is the final output of the compilation,  and it can be executed directly. | This is an intermediate file generated from the source code and is not executable by itself. It contains compiled code but lacks the full linking. |

A black screen with white text

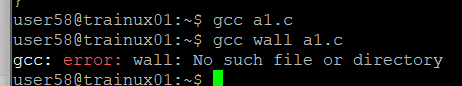
Description automatically generated

Additional gcc switches

6. Copy the program program\_warn.c as directed by the facilitator. Compile the program using gcc without any switch. Observe the result of compilation



7. Now, compile the program with the –Wall switch. Observe the warnings that are now given by the compiler.



Linking

9. 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

A computer screen with white text

Description automatically generated

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

A screen shot of a computer screen

Description automatically generated

Observe the errors received

11. 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\_

A screen shot of a computer

Description automatically generated

12. 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

The –I switch tells gcc where the included header files have to be searched. Thus –I. implies that the current directory needs to be searched by gcc



Compare the result with the earlier compilation using the same command

13. 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

If the included file name is given in double quotes, searching for the file typically starts where the source program was found. If it is not found there, then searching follows the paths given by the –I flag and implementation defined paths.

If the included file name is given in <>, searching follows the paths given by the –I flag and implementation defined paths.

A black screen with white text

Description automatically generated

14. 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

A screen shot of a computer program

Description automatically generated

15. 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

A computer screen with white text

Description automatically generated

16. 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.

A screenshot of a computer program

Description automatically generated