## **Library Module – Static Library**

1. **Create 3 files as below. Let cal\_utility.c, .h files be part of the library**
   1. **libapplication.c – will contain main() and will invoke functions in cal\_utility.c:**
2. Created a libapplication.c file that contains main() and invoking add(), sub() functions in cal\_utility.c**.**

Output: The code in libapplication.c

A computer screen with red and white text

Description automatically generated

* 1. **cal\_utility.c – will contain atleast 2 or more functions [ You may add definitions of the functions in this file]:**

1. Created a cal\_utility.c file containing definitions of mul() and div() functions.

Output: The code in cal\_utility.c

A screen shot of a computer code

Description automatically generated

* 1. **cal\_utility.h – will contain the extern declarations/prototypes of the functions in cal\_utility.c:**

1. Created a cal\_utility.h file containing prototypes of the functions in cal\_utility.

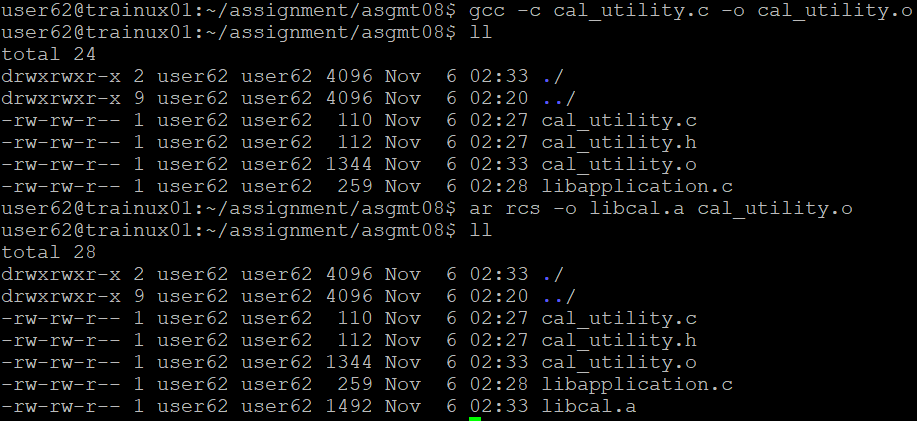
Output: The code in cal\_utility.h

A black screen with colorful text

Description automatically generated

1. **Refer the steps for static library-based application and create a static library application using above set of files?**
2. To create a static library, we need to first compile the cal\_utility.c file to create a .o file. This .o file will form the static library, to do that use ar command. (Ensure that library name is prefixed with “lib” and extension as “.a”) as $ ar rcs -o libcal.a cal\_utility.o.

Output:



1. **Execute the application created in step #2?**
2. To create an executable file, link the library with the .o file containing the main() using command below. [Exclude the “lib” prefix and extension in library name].

$ gcc libapplication.c -L./ -lcal -o stcal

Output:

