**Library Module- Static Library**

1. **Create 3 files as below. Let cal\_utility.c, .h files be part of the library**

* · libapplication.c – will contain main() and will invoke functions in cal\_utility.c
* cal\_utility.c – will contain atleast 2 or more functions [ You may add definitions of the functions in this file ]
* cal\_utility.h – will contain the extern declarations/prototypes of the functions in cal\_utility.c
* cal\_utility.h - This header file will declare the prototype of the utility functions.



A screen shot of a computer code

Description automatically generated

* cal\_utility.c - This source file will implement the functions declared in the header file.



A screen shot of a computer code

Description automatically generated

libapplication.c – This is the main application that will utilize the function from the utility.



A screenshot of a computer program

Description automatically generated

1. **Refer the steps for static library based application and create a static library application using above set of files.**

* To create a static library from the cal\_utility.c file, Compile the utility source file to object code using the command “gcc -c cal\_utility.c. This will produce an object file named cal\_utility.o.



* To create the static library using the “ar” command, “ar rcs libcal.a cal\_utility.o. This command creates a static library named libcal.a that contains cal\_utility.o.



1. **Execute the application created in step #2**

* Compile the application with static library, by compile the libapplication.c source file and link it with the static library command “gcc -o libapplication libapplication.c -L. -lcal.



* Finally, run the compiled application “./libapplication”.

A screenshot of a computer

Description automatically generated