Libraries Task

By : **Ziad Ahmed Ibrahim**
Requirements: Shared library and static libraries implementation

---

Library made:

.c file

```c
#include<stdio.h>
#include<stdint.h>

int64_t add(int32_t num1,int32_t num2)
{
return num1+num2;
}
int32_t sub(int32_t num1,int32_t num2)
{
return num2-num1;
}
int64_t mul(int32_t num1,int32_t num2)
{
return num1*num2;
}
int32_t sqrt1(int64_t num2)
{

int32_t i;
for(i=0;i<=num2/2;i++)
{

if(i*i==num2)
{
  return i;
}

}

return 1;
}
```

.h file

```c
#ifndef mathlib_h__
#define mathlib_h__
int64_t add(int32_t num1,int32_t num2);
int32_t sub(int32_t num1,int32_t num2);
int64_t mul(int32_t num1,int32_t num2);
int32_t sqrt1(int64_t num2);
#endif
```

1) Shared library

What is it?

- A shared library is used during run time
- their address is provided during linking and the library is to be accessed when needed in run time
- it preserves space unlike static libraries
- fetching of the library is done by operating system in my case the linux kernel
- we don't need to re compile the application when modifying it

How I implemented it

- Using the library `mathlib.c` I made earlier, I will be using it in my main file and access it in run time

Steps

1. Compiling the library `$gcc -c -Wall -Werror -fpic mathlib.c`
    1. -c : compile the source file into only .o
    2. -Wall: enable all compiler warning messages
    3. -Werror: treats warning as error
    4. -fpic: position independent code, any address can be given to my code
2. Creating a shared library from the .o made `gcc -shared -o libmathlib.so mathlib.o`
    1. -shared: generates shared object file for shared library
    2. -o : allows us to name the generated file
    3. we name it `libmathlib.so` as gcc recognizes all starting with lib as libraries and .so is extension of shared objects
    4. mathlib.o is the name of the used object file
3. Linking the shared library `gcc -L/home/ziad/math -Wall -o test main.c -lmathlib`
    1. we give gcc the address of the library
    2. and compile the main.c
    3. link to mathlib (-l will tell gcc to search for .so files starting with lib and named mathlib)
    4. file made will be named test using -o flag

4. add the library path to an environment variable to be accessed in run time `LD_LIBRARY_PATH=/home/ziad/math:$LD_LIBRARY_PATH`
5. export the path to be inherited by child processes `export LD_LIBRARY_PATH=/home/ziad/math:$LD_LIBRARY_PATH`
6. run the executable of the main.c named "test"

Testing the library

- here I tested the square root function

```
ziad@ziad-VirtualBox: ~/math
ziad@ziad-VirtualBox:~/math$ gcc -L/home/ziad/math -Wall -o test main.c -lmathlib
ziad@ziad-VirtualBox:~/math$ ./test

Square root of 25 is 5
ziad@ziad-VirtualBox:~/math$
```

- here I tested the rest

```
ziad@ziad-VirtualBox: ~/math
ziad@ziad-VirtualBox:~/math$ ./test

enter two numbers
10
20

10 + 20 = 30
10 - 20 = 10
10 x 20 = 200
ziad@ziad-VirtualBox:~/math$
```

- contents of the main used in previous test

```
ziad@ziad-VirtualBox:~/math$ cat main.c
#include <stdio.h>
#include<stdint.h>
#include "mathlib.h"

int main(void)
{
int x,y;
printf("\nenter two numbers\n");
scanf("%d%d",&x,&y);
printf("\n%d + %d =%ld",x,y,add(x,y));
printf("\n%d - %d =%ld",x,y,sub(x,y));
printf("\n%d x %d =%ld\n",x,y,mul(x,y));
return 0;
}
ziad@ziad-VirtualBox:~/math$
```

2)Static library

What is it?

- During building process linker starts linking exports with imports
- For example the main needs a function, the linker is responsible to provide the address of the function to be used
- in our case the address is the address of a library, and when provided it will be taken from their source into the code segment part of the memory
- If we changed the library the whole application should be built again

How I implemented it

- Using the library `mathlib.c` I made earlier, I will be using it in my main file and access it statically
- The goal here is to compile the code then link it with my library before run time
-

Steps

1. I compile my main.c `gcc -c main.c -o main.o`, this will generate main.o
2. Compile mathlib.c to Generate the mathlib.o `gcc -c mathlib.c -o mathlib.o`
3. Use archiver to create the .a file also know as the archive of the library `ar rcs libstaticmathlib.a mathlib.o`
    1. ar: to use the archiver
    2. rcs: c-> create the archive, r-> replace existing with new conents, s-> write object file index into the archive or update existing one
    3. name it `libstaticmathlib.a`

4. Now we continue manually building by linking the library by using `gcc main.o -L/home/ziad/math -lstaticmathlib -o statical_link`
    1. using gcc
    2. link `main.o` with file inside `L/home/ziad/math`
    3. `-lstaticmathlib` file name is -> `libstaticmathlib.a`
    4. name the generated file `statical_link`
5. Now we execute the file generated and test

Test

```
ziad@ziad-VirtualBox:~/math$ gcc main.o -L/home/ziad/math -lstaticmathlib -o statical_link
ziad@ziad-VirtualBox:~/math$ ls
bin  libmathlib.so  libstaticmathlib.a  main.c  main.o  mathlib.c  mathlib.h  mathlib.o  statical_link  test
ziad@ziad-VirtualBox:~/math$ ./statical_link

enter two numbers
5
5

5 + 5 = 10
5 - 5 = 0
5 x 5 = 25
ziad@ziad-VirtualBox:~/math$ gcc main.o -L/home/ziad/math -lstaticmathlib -o statical_link
```

- as we can see here the file `statical_link` is generated
- when we run it goes like we wanted from the beginning
- gives same output as the shared link