Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

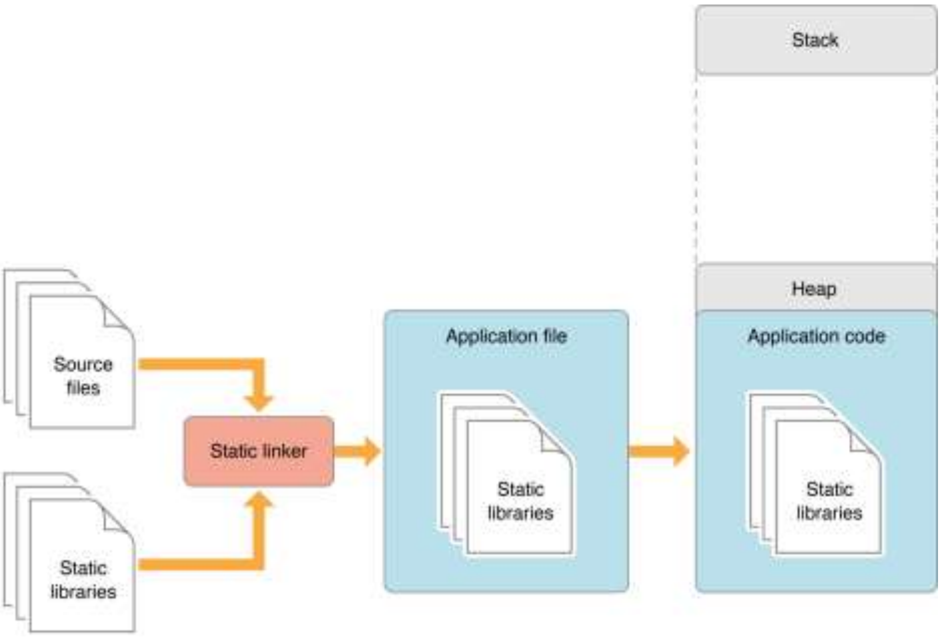
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department & Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

Name	**Pratik Pujari**		
UID no.	**2020300054**	**Class:**	**Comps C Batch**
Experiment No.	2		

AIM:	To explain the concepts of static and dynamic library
THEORY:	**Static Library:** Static libraries, while reusable in multiple programs, are locked into a program at compile time. The downside of using a static library is that its code is locked into the final executable file and cannot be modified without a re-compile. In contrast, a dynamic library can be modified without a need to re-compile.  Stack Heap Application code Static libraries Source files Static linker Application file Static libraries Static libraries Static libraries permit users to link to programs without having to recompile its code, saving recompilation time. Note that recompilation time is less important given today's

faster compilers, so this reason is not as strong as it once was. Static libraries are often useful for developers if they wish to permit programmers to link to their library, but don't want to give the library source code (which is an advantage to the library vendor, but obviously not an advantage to the programmer trying to use the library). In theory, code in static ELF libraries that is linked into an executable should run slightly faster (by 1-5%) than a shared library or a dynamically loaded library, but in practice this rarely seems to be the case due to other confounding factors.

Another benefit of using static libraries is execution speed at run-time. Because the it’s object code (binary) is already included in the executable file, multiple calls to functions can be handled much more quickly than a dynamic library’s code, which needs to be called from files outside of the executable.

Static Library Creation (Linux only)

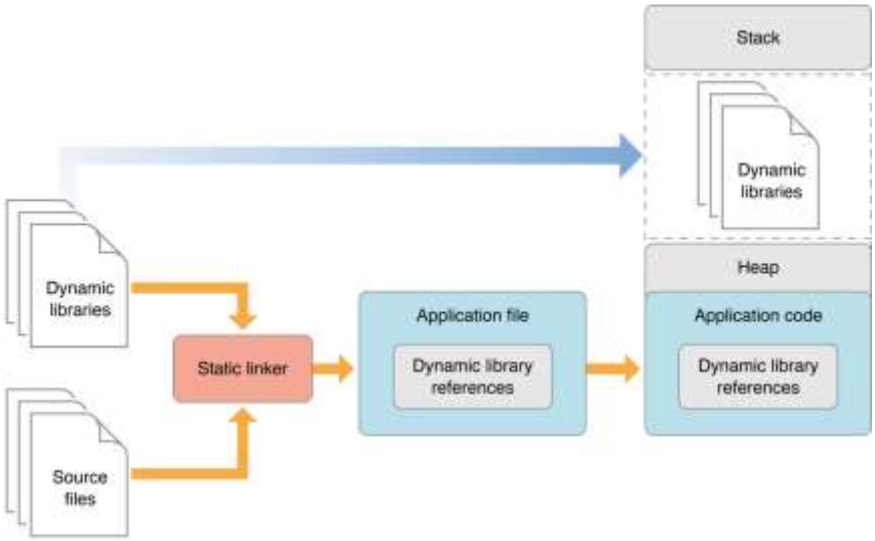
1. Creating a static library is much simpler. First, create the object files the same way as step 1 above.
2. Then archive the library using ‘ar rcs liball.a *.0’
3. Your program should include a prototype for each of the functions that exist in your library.
4. If you’re using a header file for these prototypes, make sure to include the name of that file in your other files by using #include “<header file>”

Dynamic Library

Dynamic, or shared libraries on the other hand, exist as separate files outside of the executable file. Because dynamic libraries live outside of the executable file, the program need only make one copy of the library’s files at compile-time. Whereas using a static library means every file in your program must have it's own copy of the library's files at compile-time.

The downside of using a dynamic library is that a program is much more susceptible to breaking. If a dynamic library for example becomes corrupt, the executable file may no longer work. A static library, however, is untouchable because it lives inside the executable file.

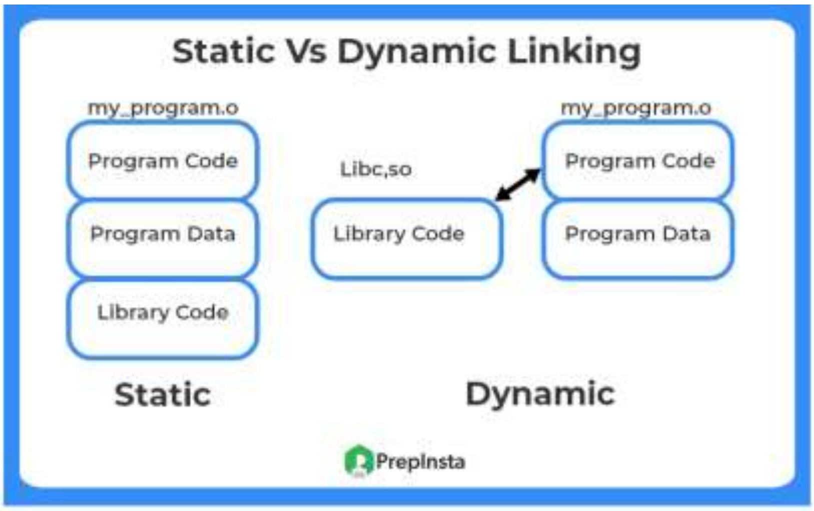
.



Dynamic Library Creation (Linux only)

1. gcc *.c -c -fpic
   The .c source files need to be prepared for use in a dynamic library. Since multiple programs can all use one instance of a dynamic library, the library can't store data at fixed addresses. This is because the location of the library in memory will vary between programs. This is done by using the compiler flag -fpic. Since we need to apply this step after the compile process has generated the object code, the compiler must be told to halt and return one object file (.o) for each source file. This is done by using the -c flag.

2. gcc *0 -shared -o liball.so
   The object files are now ready to be compiled into a dynamic library. This is done by compiling all of the .o files using by using the -shared flag. Later when compiling program files, the compiler identifies a library by looking for files beginning with 'lib' and ending with a library extension (.so for dynamic, .a for static).
3. export LD_LIBRARY_PATH=$PWD:$LD_LIBRARY_PATH
   Because a program needs to know where to look for library files, we must add that location to the environmental variable LD_LIBRARY_PATH.



Program compilation
gcc -L. -lall -o my_program main.c
When compiling program files, we have to tell the compiler to use the library files and where to find them. '-l' tells it we want to include library files. And 'all' tells it to look for the library liball.so. It's important to leave the 'lib' and '.so' out of the flag because the compiler already identifies library files that way. '-L.' tells the compiler it can find the library file in the current directory.

Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department & Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

STATIC LIBRARY

CODE:

Addition of two complex numbers:

```
#include <stdio.h>
void add(float real1,float img1,float real2,float img2){
	printf("The addition of complex numbers is %.2f
+%.2fi\n",(real1+real2),(img1+img2));
}
```

Subtraction of two complex number:

```
#include <stdio.h>
void sub(float real1,float img1,float real2,float img2){
	printf("\nThe subtraction of complex numbers is %.2f
+%.2fi\n",(real1-real2),(img1-img2));
}
```

Header file:

```
void add(float,float,float,float);
void sub(float,float,float,float);
```

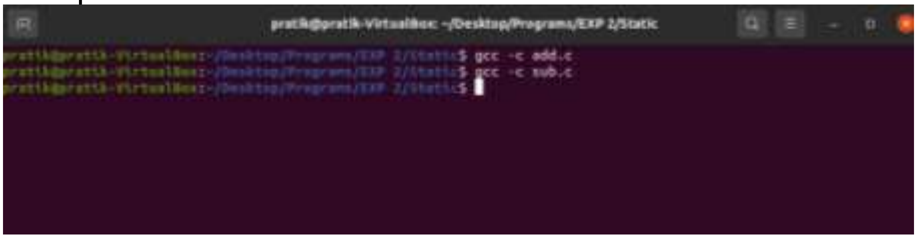
Program Code:

```
#include<stdio.h>
#include"complex.h"

int main(){
	float real1,img1,real2,img2;
	printf("\nEnter the real and imaginary part of first
number: ");
	scanf("%f %f",&real1,&img1);
	printf("\nComplex Number 1:
%.2f+%.2fi\n",real1,img1);
	printf("\nEnter the real and imaginary part of second
number: ");
	scanf("%f %f",&real2,&img2);
	printf("\nComplex Number 1:
%.2f+%.2fi\n",real2,img2);
	add(real1,img1,real2,img2);
```

```
        sub(real1,img1,real2,img2);
        return 0;
}
```

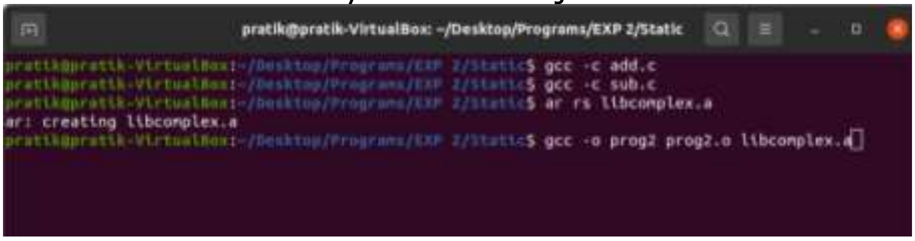
OUTPUT:

Compile the add and sub files



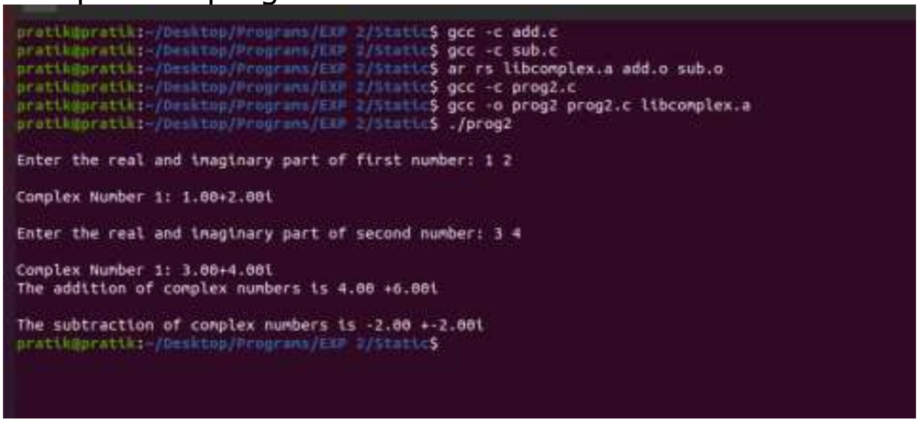
```
pratik@pratik-VirtualBox: ~/Desktop/Programs/EXP 2/Static
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Static$ gcc -c add.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Static$ gcc -c sub.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Static$ 
```

Make the static library with the object files in it.



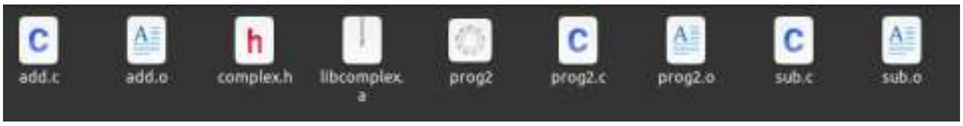
```
pratik@pratik-VirtualBox: ~/Desktop/Programs/EXP 2/Static
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Static$ gcc -c add.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Static$ gcc -c sub.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Static$ ar rs libcomplex.a
ar: creating libcomplex.a
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Static$ gcc -o prog2 prog2.o libcomplex.a
```

Compile the program:



```
pratik@pratik:~/Desktop/Programs/EXP 2/Static$ gcc -c add.c
pratik@pratik:~/Desktop/Programs/EXP 2/Static$ gcc -c sub.c
pratik@pratik:~/Desktop/Programs/EXP 2/Static$ ar rs libcomplex.a add.o sub.o
pratik@pratik:~/Desktop/Programs/EXP 2/Static$ gcc -c prog2.c
pratik@pratik:~/Desktop/Programs/EXP 2/Static$ gcc -o prog2 prog2.c libcomplex.a
pratik@pratik:~/Desktop/Programs/EXP 2/Static$ ./prog2

Enter the real and imaginary part of first number: 1 2

Complex Number 1: 1.00+2.00i

Enter the real and imaginary part of second number: 3 4

Complex Number 1: 3.00+4.00i
The addition of complex numbers is 4.00 +6.00i

The subtraction of complex numbers is -2.00 +-2.00i
pratik@pratik:~/Desktop/Programs/EXP 2/Static$
```

Files



Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

DYNAMIC LIBRARY

CODE:

Program to multiply two complex number

```
#include <stdio.h>
void mult(float real1,float img1,float real2,float img2){
	float real,img;
	real=real1*real2-img1*img2;
	img=real1*img2+real2*img1;
	printf("\nThe multiplication of complex numbers is %.2f +%.2fi\n",real,img);
}
```

Program to find the modulus of two complex number

```
#include<math.h>

float mod(float real,float img){
	float mod=sqrt(real*real+img*img);
	return mod;
}
```

Program to conjugate two complex number

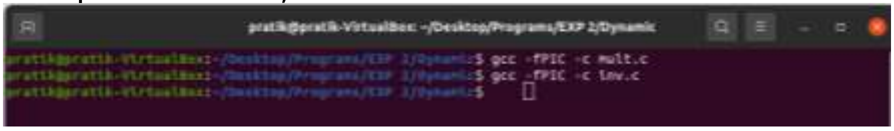
```
#include<stdio.h>
void conjugate(float real,float img){
	printf("\nThe conjugate of the %.2f+%.2fi is %.2f-%.2fi\n",real,img,real,img);
}
```

Program Code:

```
#include<stdio.h>
#include "complex.h"

int main(){
	float real1,img1,real2,img2;
	printf("\nEnter the real and imaginary part of first number: ");
	scanf("%f %f",&real1,&img1);
	printf("\nComplex Number 1:
%.2f+%.2fi\n",real1,img1);
	printf("\nModulus of %.2f+%.2fi is
%.3f\n",real1,img1,mod(real1,img1));
	printf("\nEnter the real and imaginary part of second
number: ");
	scanf("%f %f",&real2,&img2);
	printf("\nComplex Number 1:
%.2f+%.2fi\n",real2,img2);
	printf("\nModulus of %.2f+%.2fi is
%.3f\n",real1,img1,mod(real2,img2));
	conjugate(real1,img1);
	conjugate(real2,img2);
	return 0;
}
```

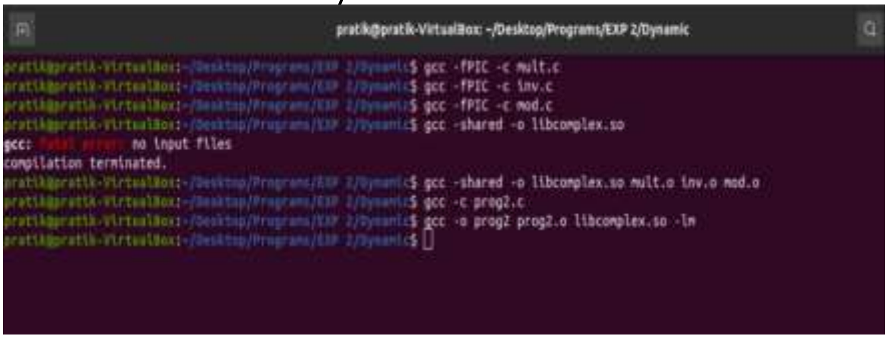
OUTPUT:

Compile the mult,inv and mod files



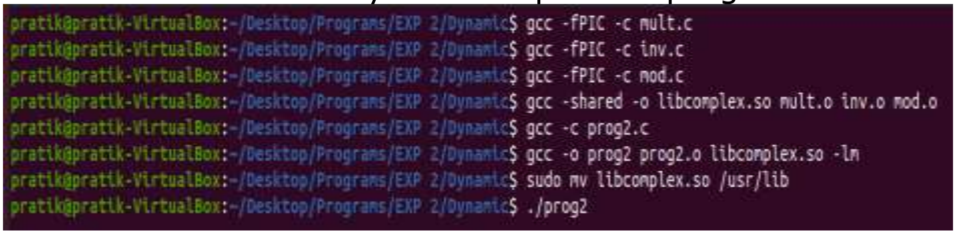
```
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -fPIC -c mult.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -fPIC -c inv.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$
```

Make a shared library and link it with the code file:



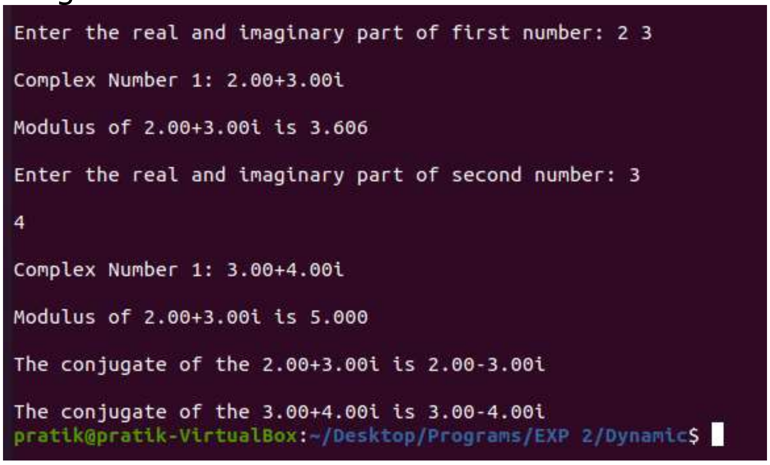
```
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -fPIC -c mult.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -fPIC -c inv.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -fPIC -c mod.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -shared -o libcomplex.so
gcc: fatal error: no input files
compilation terminated.
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -shared -o libcomplex.so mult.o inv.o mod.o
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -c prog2.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -o prog2 prog2.o libcomplex.so -lm
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$
```

Move the shared library and compile the program:



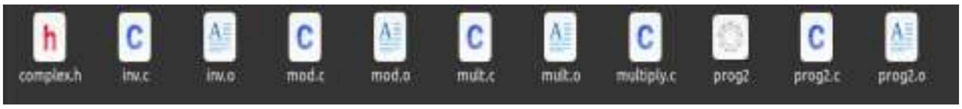
```
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -fPIC -c mult.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -fPIC -c inv.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -fPIC -c mod.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -shared -o libcomplex.so mult.o inv.o mod.o
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -c prog2.c
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ gcc -o prog2 prog2.o libcomplex.so -lm
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ sudo mv libcomplex.so /usr/lib
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$ ./prog2
```

Program Code:



```
Enter the real and imaginary part of first number: 2 3

Complex Number 1: 2.00+3.00i

Modulus of 2.00+3.00i is 3.606

Enter the real and imaginary part of second number: 3

4

Complex Number 1: 3.00+4.00i

Modulus of 2.00+3.00i is 5.000

The conjugate of the 2.00+3.00i is 2.00-3.00i

The conjugate of the 3.00+4.00i is 3.00-4.00i
pratik@pratik-VirtualBox:~/Desktop/Programs/EXP 2/Dynamic$
```

Files:



RESULT: I learnt about the shared library and dynamic library. Also learnt how to include math library in shared lib to use the sqrt function. -lm is used to while linked