Introduction to Static & Dynamic Library

K. Prasanna Kumar

CONTENTS

***Abstract*—This module explains how to create user defined static and dynamic libraries using C or C++ Programming in Linux Operating System.**

I. INTRODUCTION

- **Function :** Group of pre-compiled codes.
- **Library :** Package of functions, to avoid repetition of codes.
- There are two type of libraries
  1) Static Library
  2) Dynamic Library
- Library is not an executable. It is used either at compile time or run time.

II. EXECUTABLE FILE

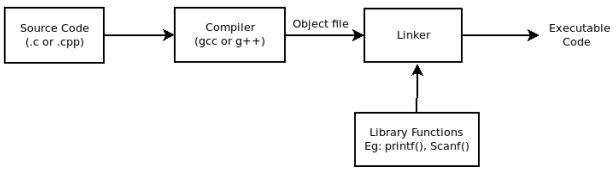

Fig. 1. Steps to Create Executable File

Program 1: Create an executable for the following C program **"Finding the largest number in the array"**

```
#include<stdio.h>
int main()
{
int arr[] = {20, 10, 20, 4, 100};
int n = sizeof(arr)/sizeof(int);
int lrg_elmt = arr[0];

for(int i = 0; i< n; i++)
{
if (arr[i] > lrg_elmt)
{ lrg_elmt = arr[i];}

}

printf("The largest element of the array is : %d\n
return 0;
}
```

Commands to build Executable

- Compile the code using following command.

  ```
  gcc -c lrg_array.c -o larg_arry.o
  ```

- Generating an executable by linking to libraries

  ```
  gcc larg_arry.o -o larg_arry
  ```

- Run the executable

  ```
  ./larg_arry
  ```

Program 2: Generating samples of sine signal with sample time T_s = 0.1 in the range $[0, 2\pi]$

```
#include<stdio.h>
#include<math.h>
int main()
{
float Ts = 0.1;
float To = 2*22/7;
// To Fundamental Period
int n = To/Ts;
// n is the size of the array

float t[n];
t[0] = 0;
printf("t = \n%f \t", t[0]);
for(int i =1; i<n; i++)
{
t[i] = t[i-1] + Ts;
printf("%f \t", t[i]);
}
printf("\n\n");


printf("Sine_wave  = \n");
for(int i =0; i<n; i++)
{
 printf("%f \t", sin(t[i]));
}
printf("\n");
```

The author is with 5G Lab, IISc Bangalore. Email: kprasannakumar.iith@gmail.com, prasannakk@iisc.ac.in

```
return 0;
}
```

Commands to build Executable

- Compile the code using following command.

  ```
  gcc -c sine_sig.c -o sine_sig.o
  ```

- Generating an executable by linking to libraries

  ```
  gcc sine_sig.o -lm -o sine_sig
  ```

  "-lm" is for linking math library.
- Run the executable

  ```
  ./sine_sig
  ```

III. Static Library

Program 3: Program file with function definitions of arithmetic operations

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

int addition(int A, int B)
{
 int C = A + B;
 return C;
}


int substraction(int A, int B)
{
 int C = A - B;
 return C;
}
```

Program 4: Program file with function definitions of Bitwise operations

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

int bit_and(int a, int b)
{
int c = a&b;
return c;
}

int bit_or(int a, int b)
{
int c = a|b;
return c;
}

int bit_not(int a)
{
int b = ~a;
return b;
}
```

Program 5: Header file

```
#include<stdio.h>

// Arthematic Operations
int addition(int, int);

int substraction(int, int);


// Bit Wise Logical Operations
int bit_and(int, int);

int bit_or(int , int);

int bit_not(int);
```

Program 6: main file

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

int main()
{
int a, b, c;
printf("Enter the value of a : ");
scanf("%d", &a);
printf("Enter the value of b : ");
scanf("%d", &b);

c = addition(a, b);
printf("Addition Operation of a and b : %d\n", c);

c = substraction(a, b);
printf("Difference of a and b : %d\n", c);

c = bit_and(a, b);
printf("Bitwise and operation between a, b : %d\n"

c = bit_or(a, b);
printf("Bitwise or operation between a, b : %d\n",

c = bit_not(a);
printf("Bitwise not operation of a : %d\n", c);

return 0;
}
```

Commands to build static lib

- Compilation of Source codes except the main to generate the object files

  ```
  gcc -c arthematic.c -o arthematic.o
  gcc -c logic.c -o logic.o
  ```

- Create a static library using above created objects files by using the following command

```
ar rcs libalu.a arthematic.o logic.o
```

- Compilation of main program generate object file

```
gcc -c alu.c -o alu.o
```

- Linking static library main program object file to create executable

```
gcc alu.o -L. libalu.a -o alu
```

Extension to static library is ".a"

- Running executable

```
./alu
```

IV. Dynamic Library

Programs in used in building static library can be used for building dynamic library. Dynamic library is also know as shared library

Commands to build dynamic lib

- Compilation of Source codes except the main to generate the object files

```
gcc -c arthematic.c logic.c -fpic
```

"-fpic " will create object files for all the sources codes mentioned in the command.

- Combining all the object files to create shared library

```
gcc *.o -shared -o libpraalu.so
```

Extension to shared library is ".so"

- Copy the shared library generated to "/usr/lib"

```
sudo cp /usr/lib/libpraalu.so
```

- Compilation of main program generate object file

```
gcc -c alu.c -o alu.o
```

- Linking dynamic library main program object file to create executable

```
gcc alu.o -L. -libpraalu.so -o alu
```

or

```
gcc alu.o -L. -lpraalu.so -o alu
```

- Running executable

```
./alu
```

References

[1] iFocus Institute, YOUTUBE CHANEL
[2] Dia, Software used to build flowcharts