****
Guide To Gcc[Part 1]

By, Asis Biswas
****

Introduction:

This is a simple tutorial[1] about how to teach yourself Gcc, the master compiler.Just read this tutorial step by step and i think you will find a new angle of c programming.At first i'll give you the idea how you can compile small programs with it.Then i will give you the idea of making libraries, both static and dynamic.Next parts of this tutorial will contain more advenced features like inline assembly, module programming etc.

Things U need:

I hope that you know simple C programming.And softwares you need are:
a)Gcc compiler,b)As(Gnu assembler),c)Ld(Gnu Linker).If you are a linux user then probably you have all these things already installed ,if not download them from www.gnu.org and if you are a windows user then download Djgpp compiler(windows port of Gcc) from www.delorie.com/Djgpp and binutils package for windows from Gnu site for As and Ld. If you download full Djgpp then you will get As and Ld with it.

History of Gcc:

Richard Stallman, the founder of Free Software Foundation was developing Gnu operating system.But for coding of it he had not a good optimized,customizable C compiler which can output object files in various formats.Then he made the Gcc compiler.

At the time of developing,Gcc was an acronym of GNU C Compiler.But now it has a name GNU Compiler Collection that means today Gcc can compile many source programs like C,C++,Java, Fortran and many more. And it has been ported into 30(around!!) different architectures.

Installation:

Install Gcc or Djgpp properly according to the instruction accompanying them.Here i will use GNU/Linux environment and save all my programs in my home directory, And i recommend you to do the same.But they are applicable to windows gcc too.Just change the pathname and other convensions.

So let's start:

Here i will give you the idea of compiling a small program in Gcc.Just follow the steps

a)Go to command prompt,

b)Editing with vi editor:
---
i)In command prompt type-

```
$vi test.c
```

this will open vi editor for editing test.c file

ii)Press insert or i key once,

iii)write the following code:

```
#include<stdio.h>

int main()
{
   printf("Hello World.\n");
   return 0;
}
```

iv)Press Esc key once,

v)Press :

vi)Now you should see a : in lower left corner of the editor, if not goto step iv and continue.

vii)Press wq and enter, that means save and quit.
---

c)Now you are again in command prompt,

d)Compiling the program:

```
$gcc test.c
```

Basically this command compiles test.c and links all the library.So this is not only
compiling but compiling+linking.

e)If compile goes successfully then you will get an executable a.out (stands for assembly out) in
your current directory.

f)Running or executing the program:

```
$./a.out
```

here . means current directory so by ./a.out we are giving a command to execute a.out
which is under current directory.

g)You will get the output "Hello World".

And this is the simplest form of using gcc.

Gcc Options:

You can specify a lot of command line options at the time of compiling your program depending on what you want.I am not going to list all the options but few of them are really important and necessary----

a)If you don't like the name a.out or you want to give another name to the exeutable then specify it with -o option.

For example: if you want you can give the executable the name hw by following commnd

```
$gcc -o hw test.c
```

then execute it by ./hw instead of ./a.out

b)If you want to output the object file only then you can do it by -c command line option. Technically it means you want to compile the program but not link it.This is useful when you are creating libary or mixing assembly program with c.

For example:

```
$gcc -c test.c
```

This will output a file named test.o(o means object file).You can make executable from test.o too.Just type

```
$gcc -o hw test.o
```

which will give you executable named hw

c)You can specify include directory by -I option. Suppose you want to use a header file that is not in a standard place.Then you can do two things---copy the header file at standard place and use or

```
$gcc -o hw -I/usr/mine/asis test.c
```

[here we assume that test.c needs a header file existed in the /usr/mine/asis directory which is not a standard place]

in this command gcc will find header files necessary in specified as well as standard directory.

**standard include directory is /usr/include

d)In the same way if you have a libary file in a nonstandard location then you can specify it with -L option.

Suppose you want to compile a mysql client program which needs a library named libmysqlclient.a which exists in /usr/lib/mysql directory what is different from standard directory /usr/lib.It also uses a nonstandrd header directory /usr/include/mysql.Then you

should use the following command line--

```
$gcc -o exe_name -I/usr/include/mysql -L/usr/lib/mysql source.c -
lmysqlclient
          |               |                   |                       |
          |               |                   |                       |--
link with libmysqlclient
          |               |                   |
          |               |                   |-search for library in
/usr/lib/mysql as well as
          |               |                     standard location
          |               |-search for header files in
/usr/include/mysql as well as standard
          |                 location
          |-executable file name of your choice.
```

e)If you want that gcc will output all warnings at the time of compiling then use -Wall option.
By default gcc doesn't output all warnings at the time of compiling.This is useful when you are
intended in developing a bug free software or debugging.

-----OK that should be enough for beginners, After all this is not a guide for experts.If you
want to know how many options Gcc provides then open man page of gcc.But be careful.....because
I experienced a minor heart attack....

Now you know simple editing with vi editor, compiling your program and some useful command line
options, Now let's create library file----

What?? Library! Header files!

Simply libraries are precompiled functions. And header files hold the prototype
declarations of those functions and other useful definitions.When you use a function like
printf in your program you must have to include the stdio.h header file because it holds the
prototype declaration stdio.h. you can see it by opening stdio.h file.

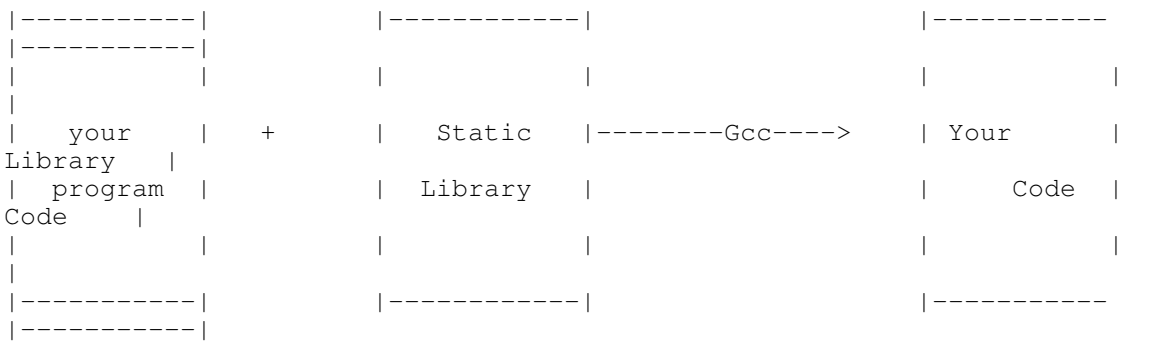
And for each functions like printf there is a low level handler funtion that is already
compiled in a library file.Every operating system has its own mechanism to handle input/output.
So as printf is an output function,it is very much operating system dependent.To provide you
a standard interface of c programming compiler designers hides the system dependencies by
provideing the precompiled system dependent functions in form of library files.Thats why you
can use same printf in your program wheather you are using Linux or Windows or Dos or...
But all functions in libary are not system dependent but they provide those too to follow a
standard like ANSI, and libraries gives the opportunity to reuse the code,which is very crucial

thing in programming.So make a library and use it in any program that needs a function of this
library.

Basically libraries are object files, when you use $gcc test.c then at first gcc makes a
object file from test.c and links this file with the libabry named libc which contains actual
implementation of printf function and makes a.out executable file.

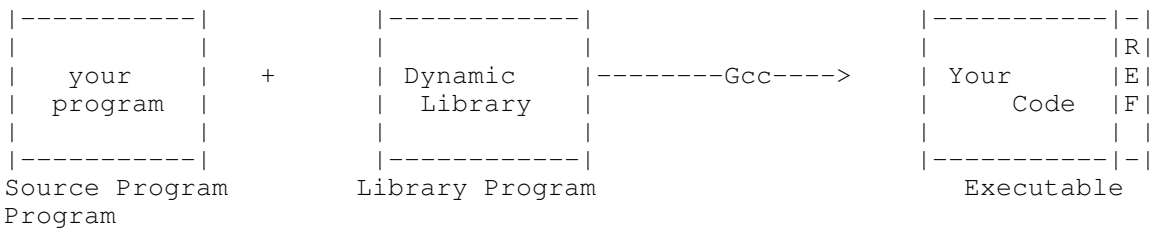
Static and Dynamic Library:

There are two types of library static and dynamic(or shared!).When you are compiling a
program and linking it with a static library then the whole libary code is added in the
executable file like following way:



so in this case both codes loaded in momory as a single exexutable and each program uses this
library has its own copy of library code.

But when you are using Dynamic library then library code in not added in the executable but it
holds a reference to this library.When executable calls a function of that library then system
first looks wheather it is already in memory loaded by another program or not.If loaded then
both programs share the library code saving the memory.If it is not loaded yet then system
loads it in memory and all subsequent calls to this library shares the same library code.
static libraries exists in the form *.o or *.a and dynamic counterparts have the form *.so.



so, shared libraries are much more sophisticated and has a great influence over system
performance bacause it provides a mechanism of saving memory.
      There is another benefit of using dynamic library. Suppose you made a static library
named libasis.a and you used this library in 5 executable programs.After some days you
discover a bug in the library and then by proper coding you make the library bug free. Now you
want to use this bug free libasis.a in all 5 programs.You have only one option that is
recompile or relink all 5 programs with this library.
      But if you had this library dynamic i.e. libasis.so then only thing you have to do is
replace the buggy library with new bug free library.The next time system will load the bug free
library.Thus it saves compile time.

Making Static Library:

      Now we want to make a static library because it is very simple.We will make libtest.a
library with two simple functions.First function has the name add which adds two numbers passed
to it as parameters and returns the sum and Second function(putst) takes a string as a
parameter and just prints it.So follow the steps---

a)At first you should make a header file containing prototypes of these two functions.So create
  a file named test.h containing following code-

```
//file test.h
float add(float,float);
int putst(char *);
```

b) Then Open your favourite editor and write down the following code--

```
//Give this file a name.I am giving first.c

#include "test.h" //for prototypes

float add(float a,float b)
{
  float c;
  c=a+b;
  return c;
}
```

then save it.

b)Create another file

```
//Give a name of your choice, I am giving second.c

#include<stdio.h>   //Because we are going to use printf in this
```

```
function
  #include "test.h"   //For prototypes

  int putst(char *str) //You should make this int instead of void but
you can void it too..
  {
    printf("\n%s\n",str);
    return 0;
  }
```

c)watch that in these two files there is no main().Because we are creating a library.

d)Next in command prompt--

```
  $gcc -c first.c   //This will output a file named first.o
  $gcc -c second.c  //This will output a file named second.o
```

e)After that we will make a archive of these two files.So in command prompt--

```
  $ar crv libtest.a first.o second.o
```

This will make a library named libtest.a containing two functions add () and putst().

f)Now we have to make a index in this library.How? simple.Just type-

```
  $ ranlib iibtest.a
```

That's all.We made a static library named libtest.a

Now we want to make use of this static library in our program.Create a file named testing.c containing following code--

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

int main()                  //Look here is main
{
  float a=3.4,b=5.6,sum;
  char str[]="Test Library";

  sum = add(a,b);           //Wow!We are using our library here

  printf("\nThe sum is: %f",sum); //Print the sum

  putst(str);            //Here too.
  return 0;
}
```

Then in command prompt---

```
  $gcc -o testing -I. -L. testing.c -ltest
```

[here . means current directory we are using this because library and header file is in our current directory]

Execute it at command line $./testing

Thats all about static library.

Making Dynamic Library:

Making Dynamic library is easy but proper installation is necessary.And in installation
there are a lot of issues which I'll not discuss here.I will give you very simple concept.

Every dynamic library has a special name called soname.Actually it is the name of the
library.
The usual convention of naming a shared library is--

lib"soname".so."version" //Ofcource without quote.Soname and version number
can be anything of your choice.

For example libmytime.so.3.0

Now we will make a shared library with previous two functions. So now you have files
test.h,first.c,second.c ready...

a)In command promp:

```
$gcc -fPIC -Wall -c first.c //This will create a file name first.o
```

Then

```
$gcc -fPIC -Wall -c second.c  //This will create a file name second.o
```

Here fPIC flag tells gcc to output "Position Independent Code" in object format.PIC is
necessary in making shared library.

b)Choice a soname(I am using test)

c)Choice a version Number(I am using 1.0.1).An issue to remember::->here 1.0.1 means increment 1
in major version 1 as in 2.0.1 means increment 1 in major version 2.

d)In command prompt type--

```
$gcc -shared -W1,-soname,libtest.so.1 -o libtest.so.1.0.1 first.o second.o -lc
```

e)Copy libtest.so.1.0.1 in directory /usr/lib [Must Do this Else it will fail because Linux
searches this location when loading shared library]

e)Thats all now write a program named testing.c as before and compile it using following
command---

```
$gcc -o testing -I. testing.c -ltest
```

Here shared library is in standard location because we copied it there.So there is
no -L option.

and enjoy executing $./testing .That's Gcc power!

Now let me ask a question to you. Have you checked the size of testing executable in two
cases? I mean after using static library and after using dynamic library?

If not do that.Watch that executable size using dynamic library is much smaller than
executable using static library.Why? Answer yourself.

Note: Dynamic library may not work as expected. Next part of this tutorial will contain this in
detail.