UNIX ar Examples: How To Create, View, Extract, Modify C Archive Files (\*.a)

*by* BALAKRISHNAN MARIYAPPAN *on* AUGUST 11, 2010

ar is an archive tool used to combine objects to create an archive file with .a extension, also known as library.

In this article, let us discuss about how to create an user defined static library in C programming using the “ar” utility. The examples shows how to create, extract, and modify the archives using Linux ar command.

To demonstrate the static library creation, let us create two C programs — addition.c and multiplication.c

Using gcc, the object code for these programs are obtained, and the static library libarith.a is created from these two objects.

1. Create Two Sample C Programs

Create addition.c program as shown below.

int addition(int a,int b)

{

int result;

result = a + b;

return result;

}

Create multiplication.c program as shown below.

int multiplication(int a, int b)

{

int result;

result = a \* b;

return result;

}

A while back we discussed about fundamental of writing C program using [C hello world example](https://www.thegeekstuff.com/2009/09/how-to-write-compile-and-execute-c-program-on-unix-os-with-hello-world-example/).

2. Compile the Programs and Get Object Codes

Use -c option to compile both the c program. Using option -c will create the corresponding .o files.

$ gcc -c addition.c

$ gcc -c multiplication.c

Now, the current working directory contains both the .c and .o files as shown below.

$ ls

addition.c multiplication.c addition.o multiplication.o

3. Create the C Program Static Library using ar utility

Now create the static library “libarith.a” with the addition object file and multiplication object file as follows,

$ ar cr libarith.a addition.o multiplication.o

4. Write C program to Use the Library libarith.a

The library file libarith.a is now ready to usage. Following example indicates how to write a sample C program with the header file to use the libarith.a static library.

Create header.h :

#include <stdio.h>

int addition(int a,int b);

int multiplication(int a,int b);

Create example.c :

#include "header.h"

int main()

{

int result;

result = addition(1,2);

printf("addition result is : %d\n",result);

result = multiplication(3,2);

printf("multiplication result is : %d\n",result);

}

Note: [How to Debug C Program using gdb in 5 Simple Steps](https://www.thegeekstuff.com/2010/03/debug-c-program-using-gdb/) provides step-by-step instruction on debugging your C code.

Compile example.c :

$ gcc -Wall example.c -L/home/guest/ -larith -o example

The option -L instructs the compiler to look in the /home/guest directory for library files. From this directory, the compiler takes the libarith library file, compiles it with example.c program.

Another method to Compile example.c :

$ gcc -Wall example.c libarith.a -o example

Execute example executable :

$ ./example

addition result is : 3

multiplication result is : 6

5. View Object Files in an Archive Using ar Command, option t

To list the object files available in the libarith.a:

$ ar t libarith.a

addition.o

multiplication.o

The options in ar command are similar to the [tar command](https://www.thegeekstuff.com/2010/04/unix-tar-command-examples/).

6. Extract Object Files from an Archive Using ar Command, option x

You can extract the object files available in an archive as follows.

$ mkdir object

$ cp libarith.a object/

$ cd object

$ ar x libarith.a

$ ls \*.o

addition.o

multiplication.o

7. Add an Object File into the Existing Archive Using ar, option r

Let assume that you have create another object file called subtraction.o

The following command extends the libarith.a library file, by inserting subtraction.o object as shown below.

$ ar r libarith.a subtraction.o

$ ar t libarith.a

addition.o

multiplication.o

subtraction.o

While inserting a .o file, it it already exists in the archive, it would be replaced. Without checking for replacements the objects can be added to end of the archive by using -q option.

8. Delete a Specific Archive Member Using ar, option d

In order to delete a specific archive member from the library file, do the following.

$ ar d libarith.a addition.o

$ ar t libarith.a

multiplication.o

subtraction.o

|  |
| --- |
| ***option*** |
| c | create a new library |
| q | add the named file to the end of the archive  *Quick append*; Historically, add the files *member*... to the end of *archive*, without checking for replacement. |
| r | replace a named archive/library member  Insert the files *member*... into *archive* (with *replacement*). This operation differs from **q** in that any previously existing members are deleted if their names match those being added |
| t | print a table of archive contents |
| d | *Delete* modules from the archive. Specify the names of modules to be deleted as *member*...; the archive is untouched if you specify no files to delete. |
| x | *Extract* members (named *member*) from the archive. You can use the **v** modifier with this operation, to request that **ar** list each name as it extracts it.  If you do not specify a *member*, all files in the archive are extracted. |
| a | Add new files *after* an existing member of the archive. If you use the modifier **a**, the name of an existing archive member must be present as the *relpos* argument, before the *archive* specification. |
| b | Add new files *before* an existing member of the archive. If you use the modifier **b**, the name of an existing archive member must be present as the *relpos* argument, before the *archive* specification. (same as **i**). |
| i | Insert new files *before* an existing member of the archive. If you use the modifier **i**, the name of an existing archive member must be present as the *relpos* argument, before the *archive* specification. (same as **b**). |
| v | This modifier requests the *verbose* version of an operation. Many operations display additional information, such as filenames processed, when the modifier **v** is appended. |