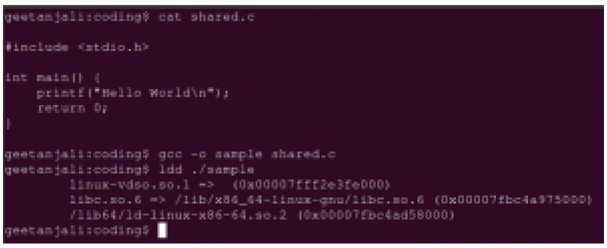
Working with Shared Libraries | Set 2

We have covered basic information about shared libraries in the previous post. In the current article we will learn how to create shared libraries on Linux.

Prior to that we need to understand how a program is loaded into memory, various (basic) steps involved in the process.

Let us see a typical "Hello World" program in C. Simple Hello World program screen image is given below.



We were compiling our code using the command "**gcc -o sample shared.c**" When we compile our code, the compiler won't resolve implementation of the function **printf()**. It only verifies the syntactical checking. The tool chain leaves a stub in our application which will be filled by dynamic linker. Since printf is standard function the compiler implicitly invoking its shared library. More details down.

We are using *ldd* to list dependencies of our program binary image. In the screen image, we can see our sample program depends on three binary files namely, *linux-vdso.so.1*, *libc.so.6* and */lib64/ld-linux-x86-64.so.2*.

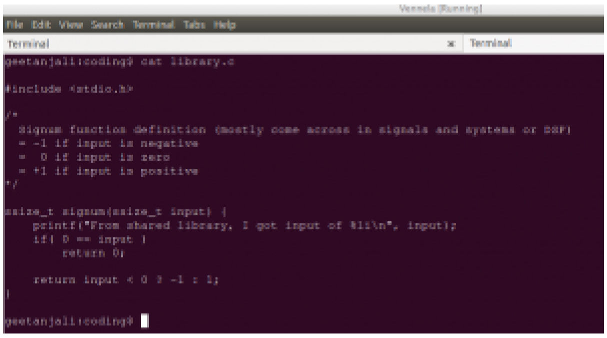
The file VDSO is fast implementation of system call interface and some other stuff, it is not our focus (on some older systems you may see different file name in liue of *.vsdo.*). Ignore this file. We have interest in the other two files.

The file **libc.so.6** is C implementation of various standard functions. It is the file where we see *printf* definition needed for our *Hello World*. It is the shared library needed to be loade▲o memory to run our Hello World program.

The third file /lib64/ld-linux-x86-64.so.2 is infact an executable that runs when an application is invoked. When we invoke the program on bash terminal, typically the bash forks itself and replaces its address space with image of program to run (so called fork-exec pair). The kernel verifies whether the libc.so.6 resides in the memory. If not, it will load the file into memory and does the relocation of libc.so.6 symbols. It then invokes the dynamic linker (/lib64/ld-linux-x86-64.so.2) to resolve unresolved symbols of application code (printf in the present case). Then the control transfers to our program *main*. (I have intensionally omitted many details in the process, our focus is to understand basic details).

Creating our own shared library:

Let us work with simple shared library on Linux. Create a file **library.c** with the following content.

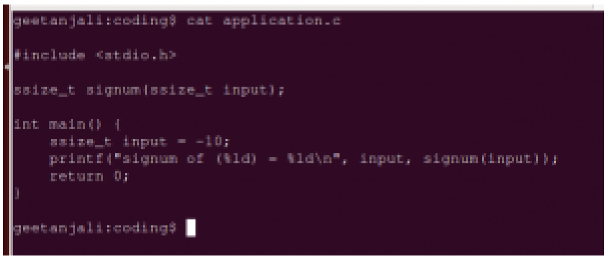


The file library.c defines a function ***signum*** which will be used by our application code. Compile the file library.c file using the following command.

gcc -shared -fPIC -o liblibrary.so library.c

The flag *-shared* instructs the compiler that we are building a shared library. The flag *-fPIC* is to generate position independent code (ignore for now). The command generates a shared library *liblibrary.so* in the current working directory. We have our shared object file (shared library name in Linux) ready to use.

Create another file **application.c** with the following content.



In the file **application.c** we are invoking the function signum which was defined in a shared library. Compile the application.c file using the following command.

gcc application.c -L /home/geetanjali/coding/ -llibrary -o sample

The flag *-llibrary* instructs the compiler to look for symbol definitions that are not available in the current code (signum function in our case). The option *-L* is hint to the compiler to look in the directory followed by the option for any shared libraries (during link time only). The command generates an executable named as “**sample**“.

If you invoke the executable, the dynamic linker will not be able to find the required shared library. By default it won't look into current working directory. You have to explicitly instruct the tool chain to provide proper paths. The dynamic linker searches standard paths available in the LD_LIBRARY_PATH and also searches in system cache (for details explore the command ***ldconfig***). We have to add our working directory to the LD_LIBRARY_PATH environment variable. The following command does the same.

export LD_LIBRARY_PATH=/home/geetanjali/coding/:$LD_LIBRARY_PATH

You can now invoke our executable as shown.

./sample

Sample output on my system is shown below.

Note: *The path **/home/geetanjali/coding/** is working directory path on my machine. You need to use your working directory path where ever it is being used in the above commands.*

Stay tuned, we haven't even explored 1/3rd of shared library concepts. More advanced concepts in the later articles.

Exercise:

It is workbook like article. You won't gain much unless you practice and do some research.

1. Create similar example and write your won function in the shared library. Invoke the function in another application.

2. Is (Are) there any other tool(s) which can list dependent libraries?

3. What is position independent code (PIC)?

4. What is system cache in the current context? How does the directory /etc/ld.so.conf.d/* related in the current context?

— **Venki**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner   Company Wise Coding Practice

Operating Systems | Memory Management | system-programming   Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

About Venki

Software Engineer

View all posts by Venki →

Recommended Posts:

File Systems | Operating System
Working with Shared Libraries | Set 1
Static and Dynamic Libraries | Set 1

Operating System | Memory Management |Partition Allocation Method
Commonly Asked Operating Systems Interview Questions | Set 1
Operating System | Introduction of Operating System – Set 1
Operating System | Process-based and Thread-based Multitasking
Operating System | Difference between multitasking, multithreading and multiprocessing
Operating System | Reader-Writers solution using Monitors
IPC through shared memory

(Login to Rate)

3.4 Average Difficulty : **3.4/5.0**
Based on **7** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List

Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!