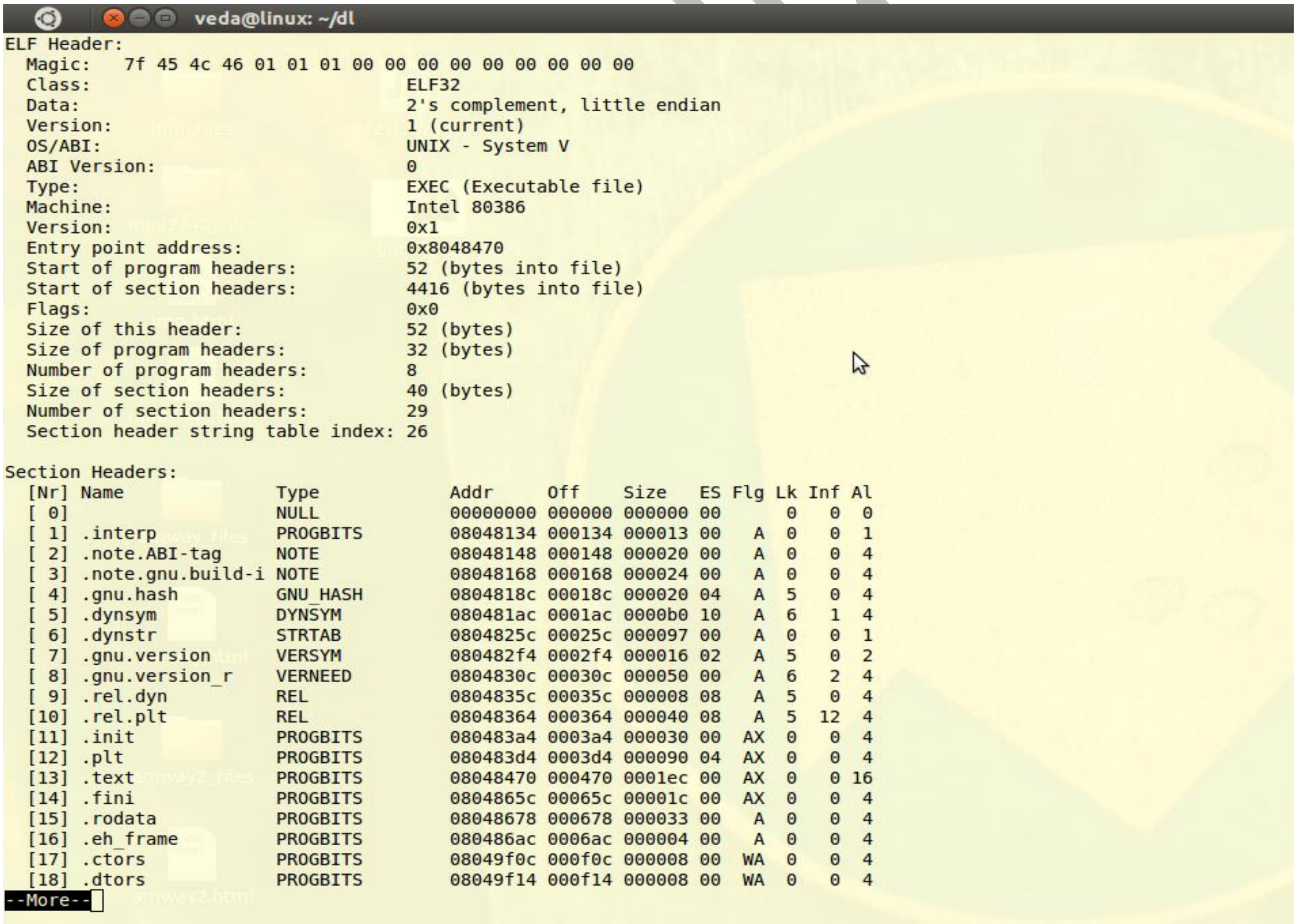
ELF – Class Notes

- Executable binary files are constructed or build using a binary file format standard.
  E.g. ELF
- Most of the binary file formats organize executable image in the binary file as per the platforms memory model.
- Executable file header provides detail about the platform for which the file is built and information about sections and segments that are part of executable image.
- To view the ELF header file of any executable, use a tool called **readelf.** Let's consider one executable image **app.**

$readelf -a app | more



```
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x8048470
  Start of program headers:          52 (bytes into file)
  Start of section headers:          4416 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         8
  Size of section headers:           40 (bytes)
  Number of section headers:         29
  Section header string table index: 26

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        08048134 000134 000013 00   A  0   0  1
  [ 2] .note.ABI-tag     NOTE            08048148 000148 000020 00   A  0   0  4
  [ 3] .note.gnu.build-i NOTE            08048168 000168 000024 00   A  0   0  4
  [ 4] .gnu.hash         GNU_HASH        0804818c 00018c 000020 04   A  5   0  4
  [ 5] .dynsym           DYNSYM          080481ac 0001ac 0000b0 10   A  6   1  4
  [ 6] .dynstr           STRTAB          0804825c 00025c 000097 00   A  0   0  1
  [ 7] .gnu.version      VERSYM          080482f4 0002f4 000016 02   A  5   0  2
  [ 8] .gnu.version_r    VERNEED         0804830c 00030c 000050 00   A  6   2  4
  [ 9] .rel.dyn          REL             0804835c 00035c 000008 08   A  5   0  4
  [10] .rel.plt          REL             08048364 000364 000040 08   A  5  12  4
  [11] .init             PROGBITS        080483a4 0003a4 000030 00  AX  0   0  4
  [12] .plt              PROGBITS        080483d4 0003d4 000090 04  AX  0   0  4
  [13] .text             PROGBITS        08048470 000470 0001ec 00  AX  0   0 16
  [14] .fini             PROGBITS        0804865c 00065c 00001c 00  AX  0   0  4
  [15] .rodata           PROGBITS        08048678 000678 000033 00   A  0   0  4
  [16] .eh_frame         PROGBITS        080486ac 0006ac 000004 00   A  0   0  4
  [17] .ctors            PROGBITS        08049f0c 000f0c 000008 00  WA  0   0  4
  [18] .dtors            PROGBITS        08049f14 000f14 000008 00  WA  0   0  4
--More--
```

Elf header for an executable provides following important details

- Platform's application binary interface standard considered while building the executable. ABI standard specifies the following:
    - Addressing format to be used by assembler and linker while binding machine instructions to a corresponding address
    - Function calling conventions (fast call , standard call , windows calling conventions)
    - Binary file format to be used to build object files
- It specifies object file type, mainly three types relocatable, executable, shared object.
- It specifies base address or entry point address of code segment (usually base address is _start)
- Offset of the program header table in the file

Load time and Run time Libraries

Load time Libraries:

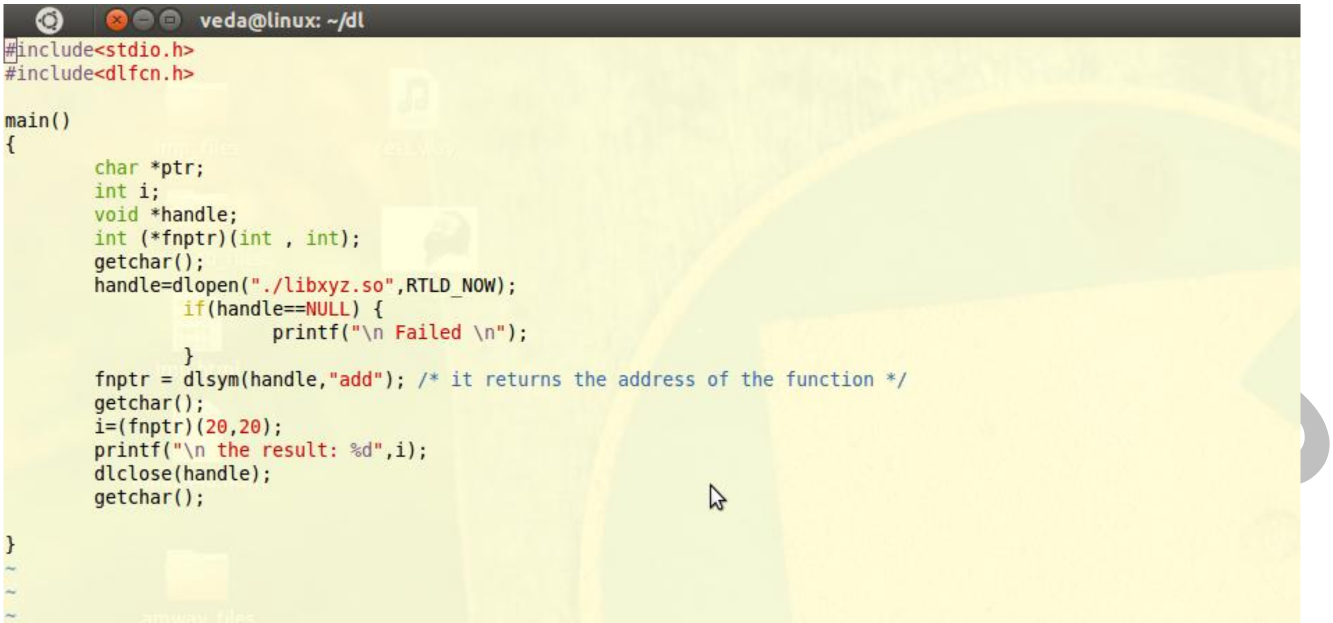
- When a shared object is linked into application address space during application initialization, it is referred as a load time library.
- All symbols linked into executables from a dynamic library are resolved during application initialization.
- Load time libraries remain resident in the process until application terminates.

Run time Libraries:

- Applications at run time could raise a request to load shared objects and bind them into address space. Such shared objects can also be detached again if application makes a request.
- If an application intends to use a shared object as runtime library, all direct references to library symbol must be avoided.

In order to understand **run time libraries** in detail, let's take a simple **C** program **test.c** which calls a shared library **libxyz.so,** where the function '**add'** resides.

$vim test.c



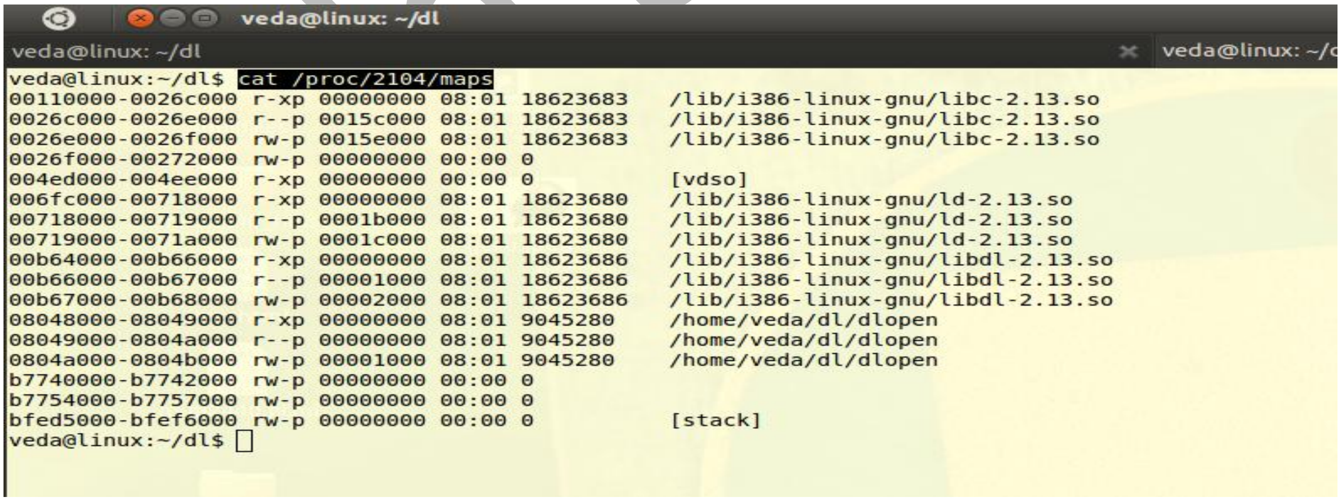
```
#include<stdio.h>
#include<dlfcn.h>

main()
{
        char *ptr;
        int i;
        void *handle;
        int (*fnptr)(int , int);
        getchar();
        handle=dlopen("./libxyz.so",RTLD_NOW);
                if(handle==NULL) {
                        printf("\n Failed \n");
                }
        fnptr = dlsym(handle,"add"); /* it returns the address of the function */
        getchar();
        i=(fnptr)(20,20);
        printf("\n the result: %d",i);
        dlclose(handle);
        getchar();

}
~
~
~
```

In the program **getchar()** is included to put the process under wait stage. Compile it by including **–ldl,** where the **dlopen ,dlsym , dlclose** are defined.

$gcc test.c -o test -ldl

Execute the output, on the first **getchar()** . Get the **PID** of our process using **ps –Af** and check $**cat /proc/2104/maps**.

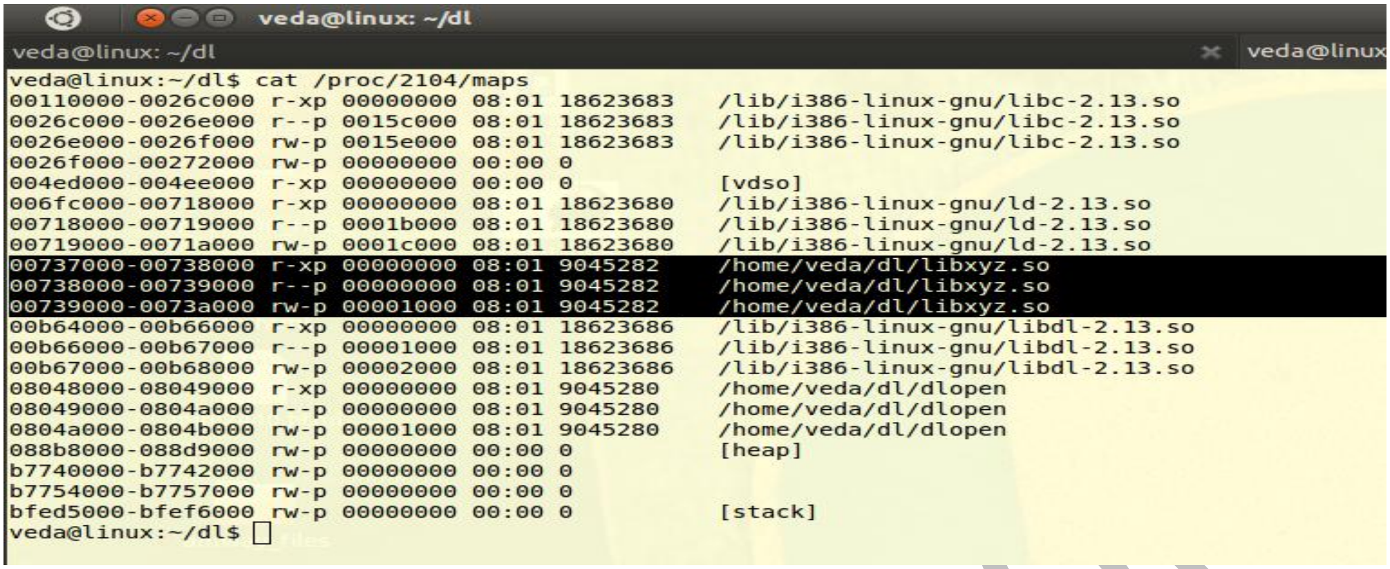
It shows as below



```
veda@linux:~/dl$ cat /proc/2104/maps
00110000-0026c000 r-xp 00000000 08:01 18623683   /lib/i386-linux-gnu/libc-2.13.so
0026c000-0026e000 r--p 0015c000 08:01 18623683   /lib/i386-linux-gnu/libc-2.13.so
0026e000-0026f000 rw-p 0015e000 08:01 18623683   /lib/i386-linux-gnu/libc-2.13.so
0026f000-00272000 rw-p 00000000 00:00 0
004ed000-004ee000 r-xp 00000000 00:00 0          [vdso]
006fc000-00718000 r-xp 00000000 08:01 18623680   /lib/i386-linux-gnu/ld-2.13.so
00718000-00719000 r--p 0001b000 08:01 18623680   /lib/i386-linux-gnu/ld-2.13.so
00719000-0071a000 rw-p 0001c000 08:01 18623680   /lib/i386-linux-gnu/ld-2.13.so
00b64000-00b66000 r-xp 00000000 08:01 18623686   /lib/i386-linux-gnu/libdl-2.13.so
00b66000-00b67000 r--p 00001000 08:01 18623686   /lib/i386-linux-gnu/libdl-2.13.so
00b67000-00b68000 rw-p 00002000 08:01 18623686   /lib/i386-linux-gnu/libdl-2.13.so
08048000-08049000 r-xp 00000000 08:01 9045280    /home/veda/dl/dlopen
08049000-0804a000 r--p 00000000 08:01 9045280    /home/veda/dl/dlopen
0804a000-0804b000 rw-p 00001000 08:01 9045280    /home/veda/dl/dlopen
b7740000-b7742000 rw-p 00000000 00:00 0
b7754000-b7757000 rw-p 00000000 00:00 0
bfed5000-bfef6000 rw-p 00000000 00:00 0          [stack]
veda@linux:~/dl$ 
```

It shows our library **libxyz.so** has not loaded.

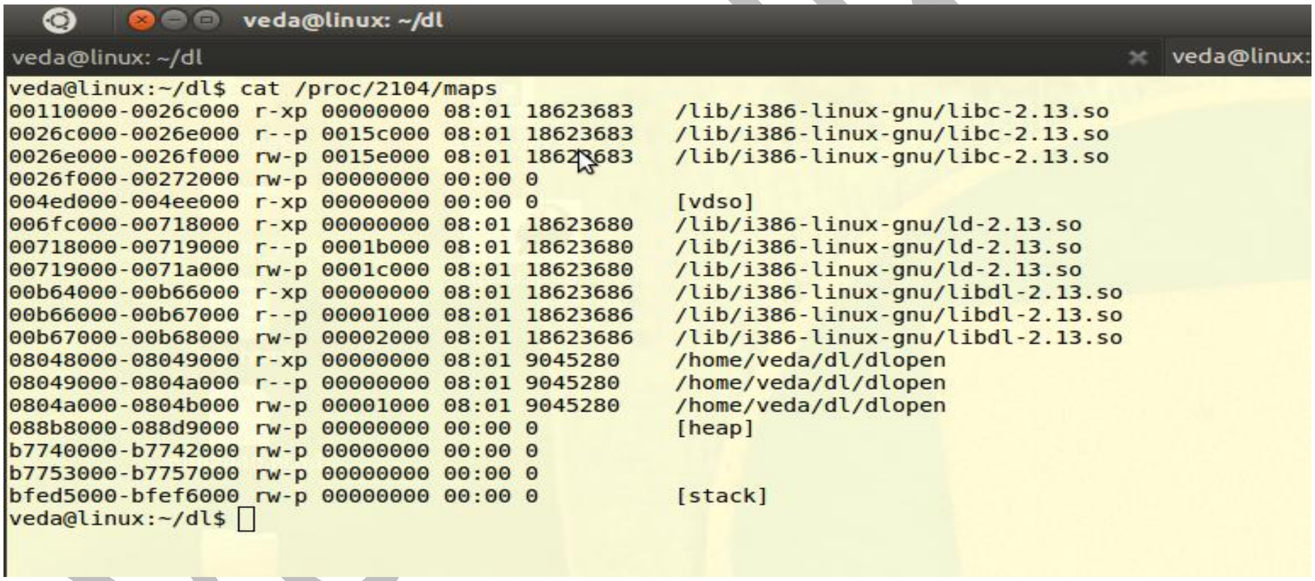
Execute the second **getchar()** and check the maps again.



```
veda@linux:~/dl$ cat /proc/2104/maps
00110000-0026c000 r-xp 00000000 08:01 18623683   /lib/i386-linux-gnu/libc-2.13.so
0026c000-0026e000 r--p 0015c000 08:01 18623683   /lib/i386-linux-gnu/libc-2.13.so
0026e000-0026f000 rw-p 0015e000 08:01 18623683   /lib/i386-linux-gnu/libc-2.13.so
0026f000-00272000 rw-p 00000000 00:00 0
004ed000-004ee000 r-xp 00000000 00:00 0          [vdso]
006fc000-00718000 r-xp 00000000 08:01 18623680   /lib/i386-linux-gnu/ld-2.13.so
00718000-00719000 r--p 0001b000 08:01 18623680   /lib/i386-linux-gnu/ld-2.13.so
00719000-0071a000 rw-p 0001c000 08:01 18623680   /lib/i386-linux-gnu/ld-2.13.so
00737000-00738000 r-xp 00000000 08:01 9045282    /home/veda/dl/libxyz.so
00738000-00739000 r--p 00000000 08:01 9045282    /home/veda/dl/libxyz.so
00739000-0073a000 rw-p 00001000 08:01 9045282    /home/veda/dl/libxyz.so
00b64000-00b66000 r-xp 00000000 08:01 18623686   /lib/i386-linux-gnu/libdl-2.13.so
00b66000-00b67000 r--p 00001000 08:01 18623686   /lib/i386-linux-gnu/libdl-2.13.so
00b67000-00b68000 rw-p 00002000 08:01 18623686   /lib/i386-linux-gnu/libdl-2.13.so
08048000-08049000 r-xp 00000000 08:01 9045280    /home/veda/dl/dlopen
08049000-0804a000 r--p 00000000 08:01 9045280    /home/veda/dl/dlopen
0804a000-0804b000 rw-p 00001000 08:01 9045280    /home/veda/dl/dlopen
088b8000-088d9000 rw-p 00000000 00:00 0          [heap]
b7740000-b7742000 rw-p 00000000 00:00 0
b7754000-b7757000 rw-p 00000000 00:00 0
bfed5000-bfef6000 rw-p 00000000 00:00 0          [stack]
veda@linux:~/dl$ 
```

Here we will find our library **libxyz.so** , which has loaded into the process.

Then execute the third **getchar()** and check maps again



```
veda@linux:~/dl$ cat /proc/2104/maps
00110000-0026c000 r-xp 00000000 08:01 18623683   /lib/i386-linux-gnu/libc-2.13.so
0026c000-0026e000 r--p 0015c000 08:01 18623683   /lib/i386-linux-gnu/libc-2.13.so
0026e000-0026f000 rw-p 0015e000 08:01 18623683   /lib/i386-linux-gnu/libc-2.13.so
0026f000-00272000 rw-p 00000000 00:00 0
004ed000-004ee000 r-xp 00000000 00:00 0          [vdso]
006fc000-00718000 r-xp 00000000 08:01 18623680   /lib/i386-linux-gnu/ld-2.13.so
00718000-00719000 r--p 0001b000 08:01 18623680   /lib/i386-linux-gnu/ld-2.13.so
00719000-0071a000 rw-p 0001c000 08:01 18623680   /lib/i386-linux-gnu/ld-2.13.so
00b64000-00b66000 r-xp 00000000 08:01 18623686   /lib/i386-linux-gnu/libdl-2.13.so
00b66000-00b67000 r--p 00001000 08:01 18623686   /lib/i386-linux-gnu/libdl-2.13.so
00b67000-00b68000 rw-p 00002000 08:01 18623686   /lib/i386-linux-gnu/libdl-2.13.so
08048000-08049000 r-xp 00000000 08:01 9045280    /home/veda/dl/dlopen
08049000-0804a000 r--p 00000000 08:01 9045280    /home/veda/dl/dlopen
0804a000-0804b000 rw-p 00001000 08:01 9045280    /home/veda/dl/dlopen
088b8000-088d9000 rw-p 00000000 00:00 0          [heap]
b7740000-b7742000 rw-p 00000000 00:00 0
b7753000-b7757000 rw-p 00000000 00:00 0
bfed5000-bfef6000 rw-p 00000000 00:00 0          [stack]
veda@linux:~/dl$ 
```

Here it doesn't shows our library which clearly explains the functionality of a run time library. It gets loaded when the program calls it and gets unloaded as soon as the function gets over.