

Link 7. Dynamic Linking

Young W. Lim

2018-09-19 Wed

- 1 Linking - 7. Dynamic Linking
 - Based on
 - Dynamic Shared Library Examples

"Self-service Linux: Mastering the Art of Problem Determination",

Mark Wilding

"Computer Architecture: A Programmer's Perspective",

Bryant & O'Hallaron

I, the copyright holder of this work, hereby publish it under the following licenses: GNU head Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

CC BY SA This file is licensed under the Creative Commons Attribution ShareAlike 3.0 Unported License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license compatible with this one.

addvec.c and multvec.c

```
/*:::::: addvec.c ::::::::::::::::::::*/
void addvec(int *x, int *y, int *z, int n)
{
    int i;

    for (i=0; i<n; i++)
        z[i] = x[i] + y[i];
}

/*:::::: multvec.c ::::::::::::::::::::*/
void multvec(int *x, int *y, int *z, int n)
{
    int i;

    for (i=0; i<n; i++)
        z[i] = x[i] * y[i];
}
```

- `gcc -c addvec.c`
 - `addvec.o`
- `gcc -c multvec.c`
 - `multvec.o`
- `gcc -shared -fPIC -o libvector.so addvec.o multvec.o`
 - `libvector.so`

```
/*::::: vector.h ::::::::::::::::::::*/  
void addvec(int *x, int *y, int *z, int n);  
void multvec(int *x, int *y, int *z, int n);  
  
/*::::: main.c ::::::::::::::::::::*/  
#include <stdio.h>  
#include "vector.h"  
  
int x[2] = { 1, 2};  
int y[2] = { 3, 4};  
int z[2];  
  
int main() {  
  
    addvec(x, y, z, 2);  
    printf("z= [%d %d]\n", z[0], z[1]);  
  
}
```

Shared libraries (1)

- an object module that can be loaded **at run time**
- at an arbitrary memory address
- linked with a program in memory
- also referred as **shared object** with **.so** suffix
- corresponds to DLLs (Dynamic Link Libraries) on MS Window
- dynamic linking is performed by a programm called a **dynamic linker**

Shared libraries (2)

- shared in two different ways
- in any given file system, for a particular library
 - there is exactly one .so file
 - the code and data in this .so file are shared by all the executable files that reference the library
 - whereas the contents are copied from static libraries
-

Shared libraries (3)

- shared in two different ways
- a single copy of the `.text` section of a shared library in memory can be shared by different running processes
-

Dynamic linking example (1)

- `main2.c, vector.h` \Rightarrow `main2.o`
 - translators (`cpp, cc1, as`)
- `main2.o, libc.so, libvector.so` \rightarrow `p2`
 - linker (`ld`)
- `p2` \Rightarrow partially linked `p2` in memory
 - loader (`execve`)
- `p2` partially linked in memory, `libc.so, libvector.so`
 \Rightarrow `p2` fully linked executable in memory
 - dynamic linker (`ld-linux.so`)

Dynamic linking example (2)

- relocatable object file : `main2.o`
- relocation and symbol table information : `libc.so`, `libvector.so`
- partially linked executable object file : `p2`
- code and data : `libc.so`, `libvector.so`
- fully linked executable in memory