Static Linking and Loading (1A)

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

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 multvec.c

ar rcs libvector**.a** addvec.o multvec.o

"Computer Architecture: A Programmer's Perspective", Bryant & O'Hallaron

main2.c

```
/*::::: main2.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]);
}
```

```
/*::::: vector.h ::::::::::::::::::::::::*/
void addvec(int *x, int *y, int *z, int n);
void multvec(int *x, int *y, int *z, int n);
```

gcc -O2 -c main2.c

gcc -static -o p2 main2.o **./libvector.a**

"Computer Architecture: A Programmer's Perspective", Bryant & O'Hallaron

Static Library

gcc -c addvec.c multvec.c

ar rcs libvector**.a** addvec.o multvec.o

gcc -O2 -c main2.c

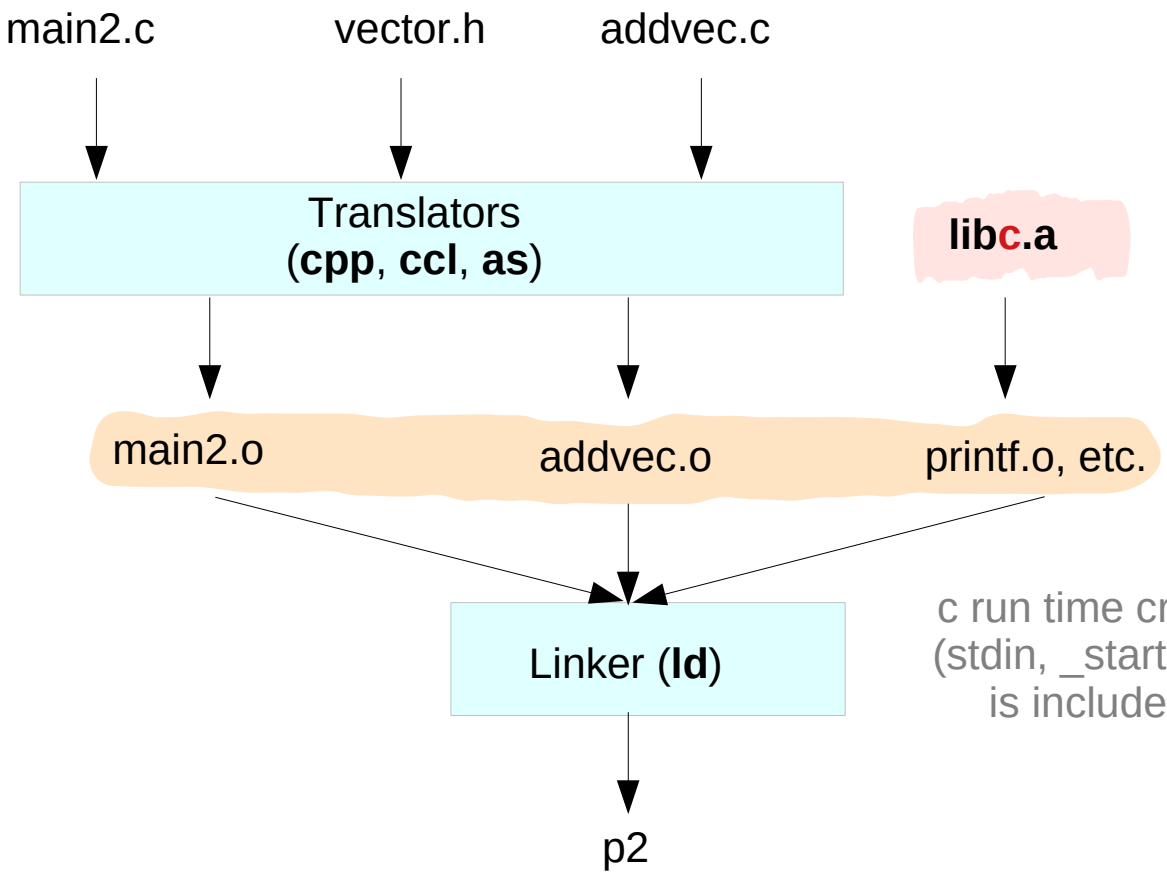
gcc -static -o p2 main2.c **./lib**vector**.a**

-static : On systems that support dynamic linking, this prevents linking withthe shared libraries. On other systems, this option has no effect.

ar c : create an archive
ar r : insert member files to the archive with replacing
ar s : add/update an index to the symbols for speeding up link process

"Computer Architecture: A Programmer's Perspective", Bryant & O'Hallaron

Static Linking



Source Files

Static Libraries

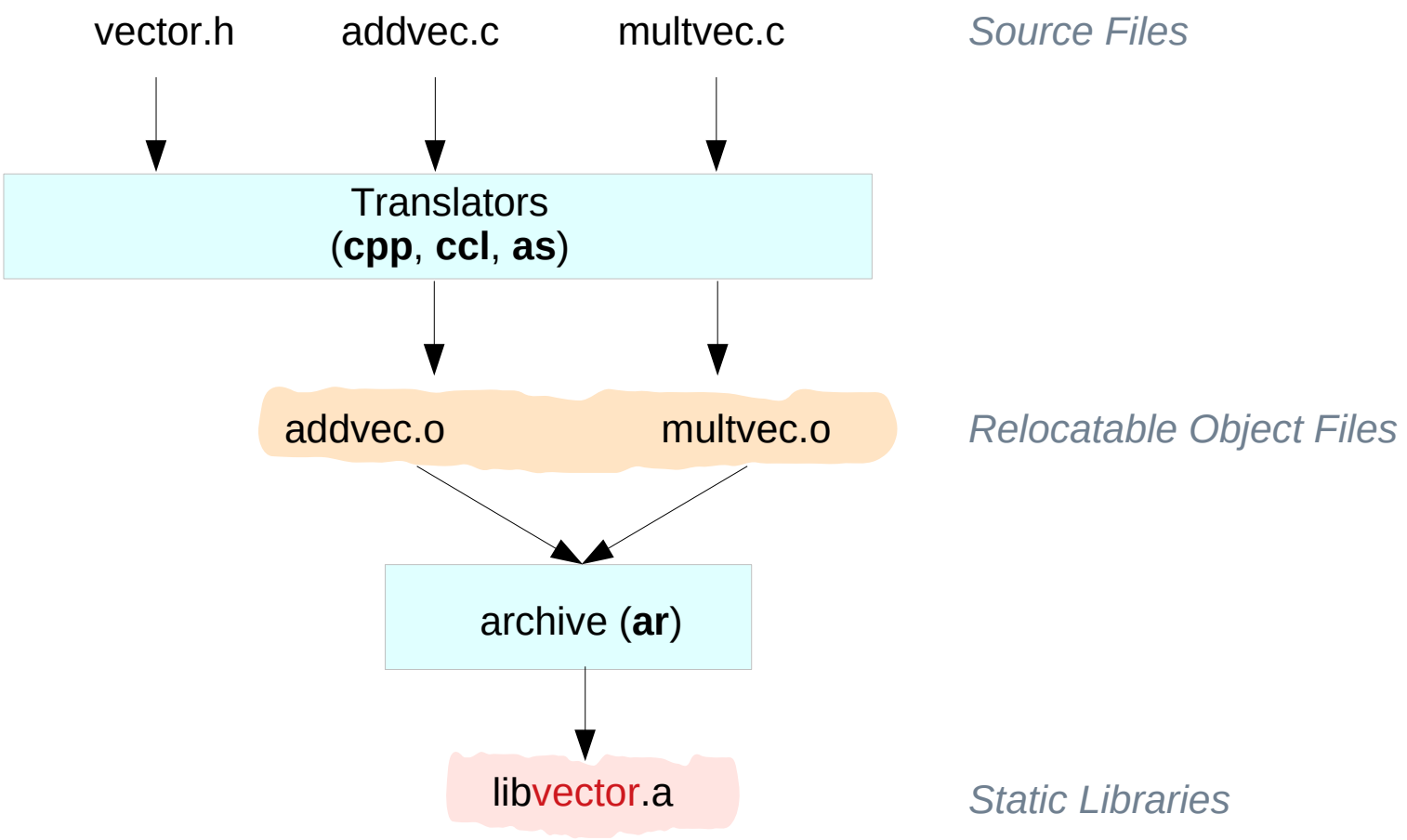
Relocatable Object Files

c run time crt1.o
(stdin, _start, …)
is included

Fully Linked
Executable Object Files

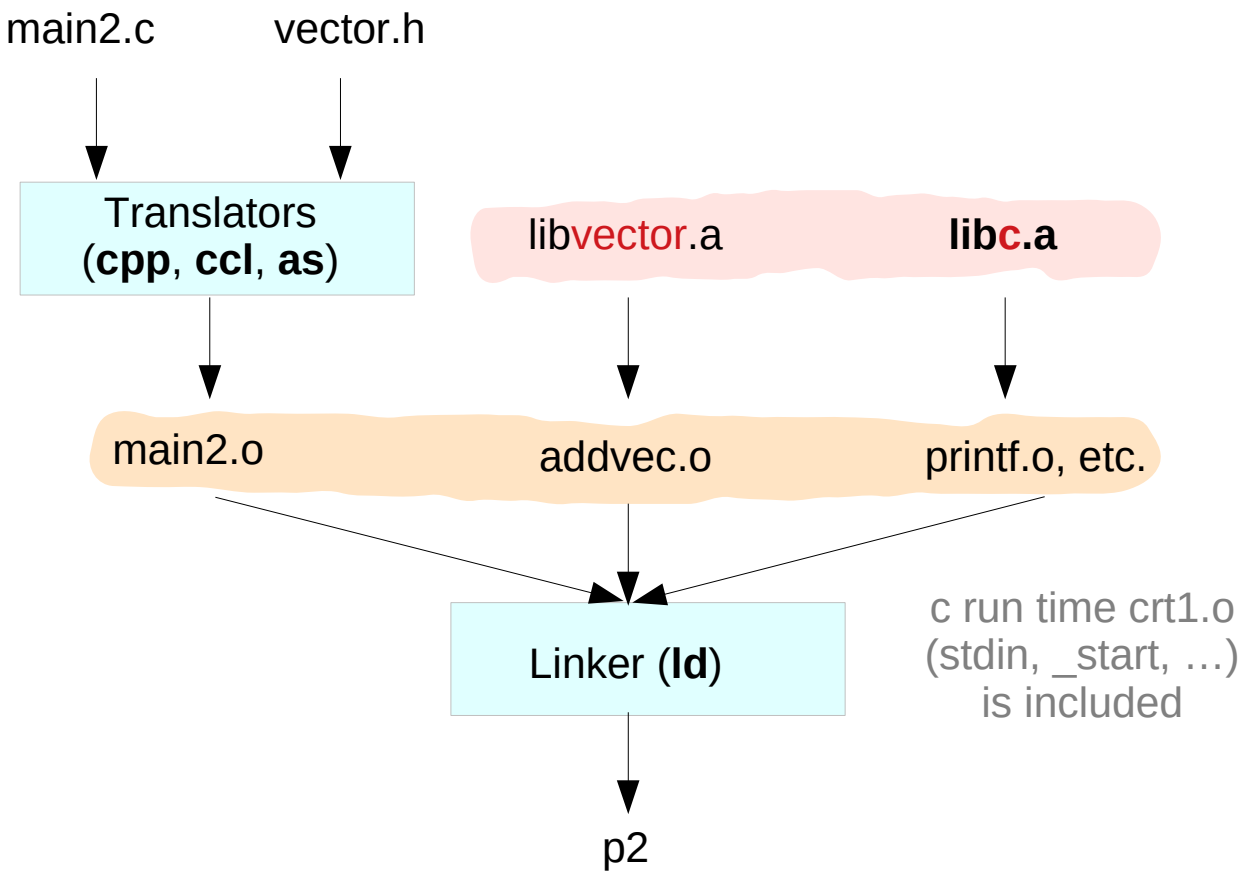
"Computer Architecture: A Programmer's Perspective", Bryant & O'Hallaron

Static Library



"Computer Architecture: A Programmer's Perspective", Bryant & O'Hallaron

Static Linking with Static Libraries



"Computer Architecture: A Programmer's Perspective", Bryant & O'Hallaron

Separate Compilation

sum.c

```
int sum(int a, int b) {
        return a+b;
}
```

mul.c

```
int mul(int a, int b) {
        return a*b;
}
```

sub.c

```
int sub(int a, int b) {
        return a-b;
}
```

div.c

```
int div(int a, int b) {
        return a/b;
}
```

Separate Compilation

main1.c

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

int main(void) {
        int a=10; int b=20;
        printf("%d \n", sum(a,b));
        printf("%d \n", sub(a,b));
        printf("%d \n", mul(a,b));
        printf("%d \n", div(a,b));
        return 0;
}
```

main2.c

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

int main(void) {
        int i;

        for (i=0; i<10; ++i)
                printf("%d \n", sum(i,20));
        return 0;
}
```

ttt.h

```
int sum(int, int);
int sub(int, int);
int mul(int, int);
int div(int, int);
```

Static Library

gcc -c sum.c ➡ sum.o
gcc -c sub.c ➡ sub.o
gcc -c mul.c ➡ mul.o
gcc -c div.c ➡ div.o

ar rcs libttt**.a** sum.o sub.o mul.o div.o

Linking with Static Library

gcc -c main1.c ➡ main1.o
gcc -c main2.c ➡ main2.o

gcc -o p1 main1.o **-L**./ **-I**./ **-lttt** ⬅ **./libttt.a**
gcc -o p2 main2.o **-L**./ **-I**./ **-lttt** ⬅ **./libttt.a**

./p1
./p2

```
LIBRARY_PATH="$LIBRARY_PATH:./:"
export LIBRARY_PATH
```

Lthe Standard C Library (libc.a)

the standard C library **libc.a**
is automatically linked into your programs

bios functions
conio functions
cpu functions
ctype functions
debugging functions
dos functions
dpmi functions
environment functions
file system functions
go32 functions
io functions
locale functions
math functions
memory functions
misc functions
mono functions
posix functions
process functions
profiling functions
random number functions
shell functions
signal functions
sound functions
startup functions crt1.o, _startup
stdio functions
string functions
sys functions
termios functions
time functions
unistd functions
unix functions

http://www.delorie.com/djgpp/doc/libc/libc_toc.html#SEC_Contents

C Run Time crt1.o, crt0.o

Both crt0/crt1 do the same thing
what is needed before calling main()
(like initializing stack, setting irqs, etc.).
You should link with one or the other but not both.
are not really libraries but really inline assembly code.

crt1 is used on systems that support
constructors and **destructors**
(functions called **before** and **after main** and **exit**).
In this case **main** is treated like a normal function call.

crt0 is used on systems that do not support
constructors / destructors

https://stackoverflow.com/questions/2709998/crt0-o-and-crt1-o-whats-the-difference

crt0.S example

```
.text

.globl _start

_start:                 # _start is the entry point known to the linker
    mov %rsp, %rbp      # setup a new stack frame
    mov 0(%rbp), %rdi   # get argc from the stack
    lea 8(%rbp), %rsi   # get argv from the stack
    call main           # %rdi, %rsi are the first two args to main

    mov %rax, %rdi      # mov the return of main to the first argument
    call exit           # terminate the program
```

https://en.wikipedia.org/wiki/Crt0

References

[1] An Introduction to GCC, B. Gough, http://www.network-theory.co.uk/docs/gccintro/
[2] Unix, Linux Programming Indispensable Utilities, CW Paik