



Let's look at some C code and its binary

```

0804840b <foo>:
804840b: 55                push    ebp
804840c: 89 e5            mov     ebp,esp
804840e: 83 ec 08        sub     esp,0x8
8048411: 83 ec 0c        sub     esp,0xc
8048414: 68 d0 84 04 08   push    0x80484d0
8048419: e8 c2 fe ff ff   call    80482e0 <printf@plt>
804841e: 83 c4 10        add     esp,0x10
8048421: 90              nop
8048422: c9              leave
8048423: c3              ret

08048424 <main>:
8048424: 8d 4c 24 04      lea     ecx,[esp+0x4]
8048428: 83 e4 f0        and     esp,0xffffffff
804842b: ff 71 fc       push    DWORD PTR [ecx-0x4]
804842e: 55              push    ebp
804842f: 89 e5            mov     ebp,esp
8048431: 51              push    ecx
8048432: 83 ec 04        sub     esp,0x4
8048435: e8 d1 ff ff ff   call    804840b <foo>
804843a: b8 00 00 00 00   mov     eax,0x0
804843f: 83 c4 04        add     esp,0x4
8048442: 59              pop     ecx
8048443: 5d              pop     ebp
8048444: 8d 61 fc       lea     esp,[ecx-0x4]
8048447: c3              ret
8048448: 66 90           xchg    ax,ax
804844a: 66 90           xchg    ax,ax
804844c: 66 90           xchg    ax,ax
804844e: 66 90           xchg    ax,ax

```

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

```
int foo(){  
    printf("hello world!");  
}
```

```
int main(int argc, char **argv){  
    foo();  
}
```

Since function addresses and others are hard-encoded in the binary, the program cannot be placed at random locations in memory

# Let's look at some C code and its binary

```
#include <stdio.h>

int foo(){
    printf("hello world!");
}

int main(int argc, char **argv){
    foo();
}
```

Since function addresses and others are hard-encoded in the binary, the program cannot be placed at random locations in memory

```
0804840b <foo>:
804840b: 55                push    ebp
804840c: 89 e5            mov     ebp,esp
804840e: 83 ec 08        sub     esp,0x8
8048411: 83 ec 0c        sub     esp,0xc
8048414: 68 d0 84 04 08   push    0x80484d0
8048419: e8 c2 fe ff ff   call   80482e0 <printf@plt>
804841e: 83 c4 10        add     esp,0x10
8048421: 90              nop
8048422: c9              leave
8048423: c3              ret

08048424 <main>:
8048424: 8d 4c 24 04      lea     ecx,[esp+0x4]
8048428: 83 e4 f0        and     esp,0xffffffff
804842b: ff 71 fc        push    DWORD PTR [ecx-0x4]
804842e: 55              push    ebp
804842f: 89 e5            mov     ebp,esp
8048431: 51              push    ecx
8048432: 83 ec 04        sub     esp,0x4
8048435: e8 d1 ff ff ff   call   804840b <foo>
804843a: b8 00 00 00 00   mov     eax,0x0
804843f: 83 c4 04        add     esp,0x4
8048442: 59              pop     ecx
8048443: 5d              pop     ebp
8048444: 8d 61 fc        lea     esp,[ecx-0x4]
8048447: c3              ret
8048448: 66 90           xchg    ax,ax
804844a: 66 90           xchg    ax,ax
804844c: 66 90           xchg    ax,ax
804844e: 66 90           xchg    ax,ax
```



# Naive Idea : load time linking

How about doing the linking when process executed, not at compile time

- ➡ Determine where process will reside in memory and adjust all references within program
- How to relocate the program in memory during execution? (consider functions but also data pointers now)
- What if no contiguous free region fits program?
- How to avoid programs interfering with each others?