

# Relocation revisited

# Example C Program

## main.c

```
int buf[2] = {1, 2};

int main()
{
    swap();
    return 0;
}
```

## swap.c

```
extern int buf[];

int *bufp0 = &buf[0];
static int *bufp1;

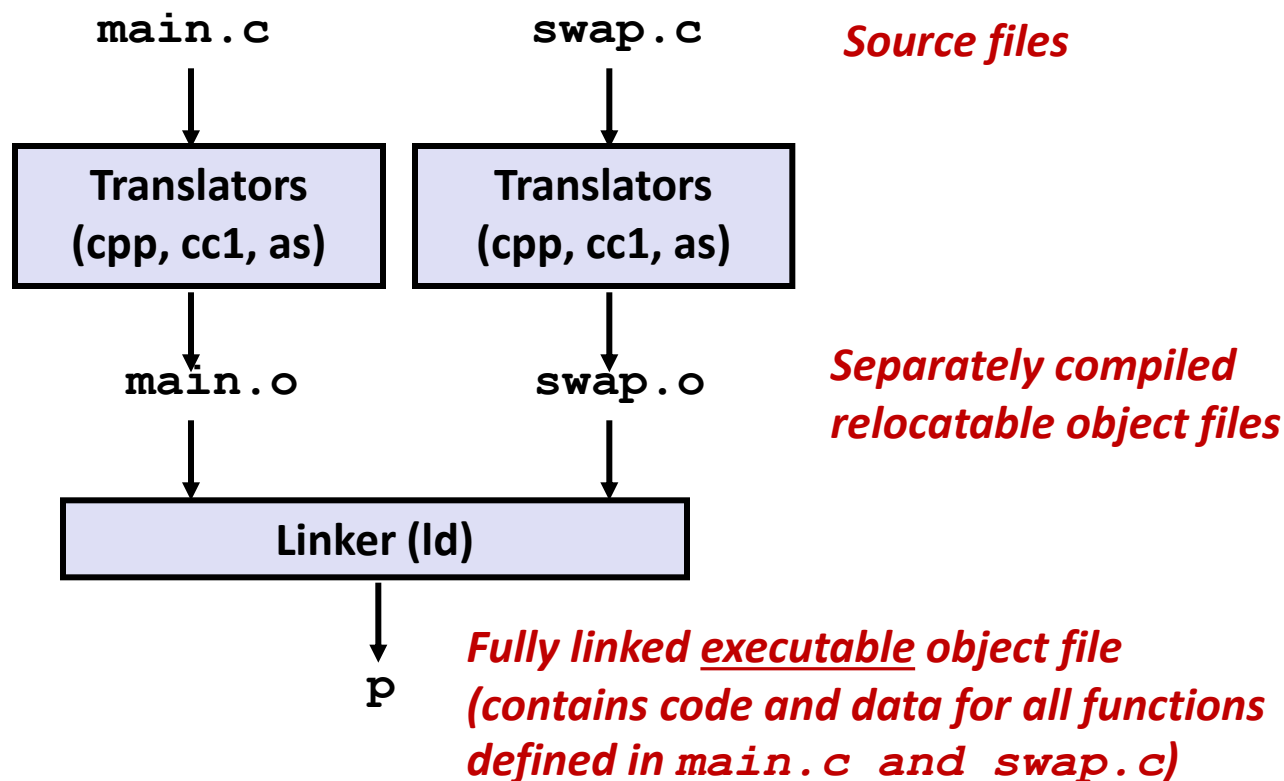
void swap()
{
    int temp;

    bufp1 = &buf[1];
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

# Static Linking

■ Programs are translated and linked using a *compiler driver*:

- `unix> gcc -O2 -g -o p main.c swap.c`
- `unix> ./p`



# What Do Linkers Do?

## ■ Step 1. Symbol resolution

- Programs define and reference *symbols* (variables and functions):
  - `void swap() {...}      /* define symbol swap */`
  - `swap();                /* reference symbol swap */`
  - `int *xp = &x;          /* define symbol xp, reference x */`
- Symbol definitions are stored (by compiler) in *symbol table*.
  - Symbol table is an array of structs
  - Each entry includes name, size, and location of symbol.
- Linker associates each symbol reference with exactly one symbol definition.

# What Do Linkers Do? (cont)

## ■ Step 2. Relocation

- Merges separate code and data sections into single sections
- Relocates symbols from their relative locations in the `.o` files to their final absolute memory locations in the executable.
- Updates all references to these symbols to reflect their new positions.

# Relocation Entries


```
typedef struct {  
    int offset;  
    int symbol:24,  
        type:8;  
} Elf32_rel;
```

- Offset : section offset of the references to relocate
- Symbol : identifies the symbol that the modified reference should point to.
- Type : tells the linker how to modify the new reference
  
- ELF defines 11 relocation types.
- two most widely used :
  - R\_386\_PC32 : 32 bit PC\_relative address
    - Add the 32 bit value to the current PC value (address of the next instruction)
  - R\_386\_32 : 32 bit absolute address


# Relocation Algorithm


```

foreach section s {
    foreach relocation entry r {
        refptr = s + r.offset; /* ptr to reference to be relocated */

        /* Relocate a PC-relative reference */
        if (r.type == R_386_PC32) {
            refaddr = ADDR( + r.offset; /* ref's run-time address */
            *refptr = (unsigned) (ADDR(r.symbol) + *refptr - refaddr);
        }
        /* Relocate an absolute reference */
        if (r.type == R_386_32)
            *refptr = (unsigned) (ADDR(r.symbol) + *refptr);
    }
}

```

  
 PC adjustment

  
 Array offset

Some compiler/linker store the **value** separately in relocation entry

# Relocation Info (main)

main.c

```
int buf[2] =
    {1,2};

int main()
{
    swap();
    return 0;
}
```

main.o

```
00000000 <main>:
    0:  8d 4c 24 04      lea    0x4(%esp), %ecx
    4:  83 e4 f0         and    $0xffffffff0, %esp
    7:  ff 71 fc         pushl  0xfffffffffc(%ecx)
    a:  55              push   %ebp
    b:  89 e5           mov    %esp, %ebp
    d:  51 -4(수행시의 PC 보정값) push   %ecx
    e:  83 ec 04        sub    $0x4, %esp
   11: e8 fc ff ff ff   call   12 <main+0x12>
                        12: R_386_PC32 swap ←
   16:  83 c4 04        add    $0x4, %esp
   19:  31 c0           xor    %eax, %eax
   1b:  59             pop    %ecx
   1c:  5d             pop    %ebp
   1d:  8d 61 fc       lea    0xfffffffffc(%ecx), %esp
   20:  c3             ret
```

```
Relocation entry{
    offset: 12
    symbol: swap
    type: R_386_PC32
    value: -4}
```

Disassembly of section .data:

```
00000000 <buf>:
    0:  01 00 00 00 02 00 00 00
```

Source: objdump -r -d



# Relocation Info (swap, .text)

swap.c

```
extern int buf[];
```

```
int
```

```
    *bufp0 = &buf[0];
```

```
static int *bufp1;
```

```
void swap()
```

```
{
```

```
    int temp;
```

```
    bufp1 = &buf[1];
```

```
    temp = *bufp0;
```

```
    *bufp0 = *bufp1;
```

```
    *bufp1 = temp;
```

```
}
```

```
Relocation entry{
  offset: (7,14,1f)
  symbol: buf
  type:R_386_32
  value: 4}
```

swap.o

Disassembly of section .text.

00000000 <swap>:

0: 8b 15 00 00 00 00

2: R\_386\_32

6: a1 04 00 00 00

7: R\_386\_32

b: 55

c: 89 e5

e: c7 05 00 00 00 00 04

15: 00 00 00

10: R\_386\_32

14: R\_386\_32

18: 8b 08

1a: 89 10

1c: 5d

89 0d 04 00 00 00

1f: R\_386\_32

c3

```
Relocation entry{
  offset: 10
  symbol: bufp1
  type:R_386_32
  value: 0}
```

mov 0x0,%edx

buf

mov 0x4,%eax

buf

push %ebp

mov %esp,%ebp

movl \$0x4,0x0

.bss

buf

mov (%eax),%ecx

mov %edx,(%eax)

```
Relocation entry{
  offset: 2
  symbol: buf
  type:R_386_32
  value: 0}
```

# Relocation Info (swap, .data)

swap.c

```
extern int buf[];

int *bufp0 =
    &buf[0];
static int *bufp1;

void swap()
{
    int temp;

    bufp1 = &buf[1];
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

Disassembly of section .data:

```
00000000 <bufp0>:
    0:  00 00 00 00

    0:  R_386_32 buf
```

# Executable Before/After Relocation (.text)

00000000 <main>:

```

. . .
e: 83 ec 04      sub    $0x4,%esp
11: e8 fc ff ff ff call   12 <main+0x12>
      12: R_386_PC32 swap
16: 83 c4 04      add    $0x4,%esp
. . .

```

-4(수행시의 PC 보정값)

0x8048396 + 0x1a  
= 0x80483b0

PC = <main+0x12>+4  
= <main+0x16>

08048380 <main>:

```

. . .
804838e: 83 ec 04      sub    $0x4,%esp
8048391: e8 1a 00 00 00 call   80483b0 <swap>
8048396: 83 c4 04      add    $0x4,%esp
. . .

```

080483b0 <swap>:

```

80483b0: 8b 15 20 96 04 08 mov    0x8049620,%edx
80483b6: a1 24 96 04 08 mov    0x8049624,%eax
80483bb: 55            push   %ebp

```

실제 계산에서는 PC를 모르기 때문에

080483b0 + (-4) - (08048380(section addr) + 12(offset)) ←

를 계산한다. = 0x1a

```

0:  8b 15 00 00 00 00      mov     0x0,%edx
           2: R_386_32      buf
6:  a1 04 00 00 00      mov     0x4,%eax
           7: R_386_32      buf
...
e:  c7 05 00 00 00 00 04  movl    $0x4,0x0
15:  00 00 00
           10: R_386_32      .bss
           14: R_386_32      buf
. . .
1d:  89 0d 04 00 00 00      mov     %ecx,0x4
           1f: R_386_32      buf
23:  c3                    ret

```

080483b0 <swap>:

```

80483b0:  8b 15 20 96 04 08      mov     0x8049620,%edx
80483b6:  a1 24 96 04 08      mov     0x8049624,%eax
80483bb:  55                    push    %ebp
80483bc:  89 e5                mov     %esp,%ebp
80483be:  c7 05 30 96 04 08 24  movl    $0x8049624,0x8049630
80483c5:  96 04 08
80483c8:  8b 08                mov     (%eax),%ecx
80483ca:  89 10                mov     %edx,(%eax)
80483cc:  5d                    pop     %ebp
80483cd:  89 0d 24 96 04 08      mov     %ecx,0x8049624
80483d3:  c3                    ret

```

# Executable After Relocation (.data)

Disassembly of section .data:

08049620 <buf>:

8049620: 01 00 00 00 02 00 00 00

08049628 <bufp0>:

8049628: 20 96 04 08

.bss

08049630 <bufp1>:

8049630: 24 96 04 08

```
extern int buf[];
```

```
int *bufp0 = &buf[0];
```

```
static int *bufp1;
```

```
void swap()
```

```
{
```

```
    int temp;
```

```
    bufp1 = &buf[1];
```

```
    temp = *bufp0;
```

```
    *bufp0 = *bufp1;
```

```
    *bufp1 = temp;
```

```
}
```

# Before Relocation (.text) main.o

Disassembly of section .text:

0000000000000000 <main>:

```

  0:    55                push    %rbp
  1:    48 89 e5          mov     %rsp,%rbp
  4:    b8 00 00 00 00    mov     $0x0,%eax
  9:    e8 00 00 00 00    callq   e <main+0xe>
                        a: R_386_PC32    <swap>
  e:    b8 00 00 00 00    mov     $0x0,%eax
13:    5d                pop     %rbp
14:    c3                retq

```

# After Relocation (.text) main

00000000004004ed <main>:

```

4004ed:    55                push    %rbp
4004ee:    48 89 e5          mov     %rsp,%rbp
4004f1:    b8 00 00 00 00    mov     $0x0,%eax
4004f6:    e8 07 00 00 00    callq   400502 <swap>
4004fb:    b8 00 00 00 00    mov     $0x0,%eax
400500:    5d                pop     %rbp
400501:    c3                retq

```

# Before Relocation (.text) swap.o

0000000000000000 <swap>:

0:	55	push	%rbp		
1:	48 89 e5	mov	%rsp,%rbp		
4:	48 c7 05 00 00 00 00	movq	\$0x0,0x0(%rip)	(수행시의 PC 보정값)	# f <swap+0xf>
			7: R_386_PC32	bufp1 -0x8	
b:	00 00 00 00				
			b: R_386_32	buf+0x4	
f:	48 8b 05 00 00 00 00	mov	0x0(%rip),%rax		# 16 <swap+0x16>
			12: R_386_PC32	bufp0 -0x4	
16:	8b 00	mov	(%rax),%eax		
18:	89 45 fc	mov	%eax,-0x4(%rbp)		
1b:	48 8b 05 00 00 00 00	mov	0x0(%rip),%rax		# 22 <swap+0x22>
			1e: R_386_PC32	bufp0 -0x4	
22:	48 8b 15 00 00 00 00	mov	0x0(%rip),%rdx		# 29 <swap+0x29>
			25: R_386_PC32	bufp1 -0x4	
29:	8b 12	mov	(%rdx),%edx		
2b:	89 10	mov	%edx,(%rax)		
2d:	48 8b 05 00 00 00 00	mov	0x0(%rip),%rax		# 34 <swap+0x34>
			30: R_386_PC32	bufp1 -0x4	
34:	8b 55 fc	mov	-0x4(%rbp),%edx		
37:	89 10	mov	%edx,(%rax)		
39:	5d	pop	%rbp		
3a:	c3	retq			

# After Relocation (.text) swap

0000000000400502 <swap>:

400502:	55	push	%rbp	
400503:	48 89 e5	mov	%rsp,%rbp	
400506:	48 c7 05 3f 0b 20 00	movq	\$0x60103c,0x200b3f(%rip)	#601050 <bufp1>
40050d:	3c 10 60 00			
400511:	48 8b 05 28 0b 20 00	mov	0x200b28(%rip),%rax	# 601040 <bufp0>
400518:	8b 00	mov	(%rax),%eax	
40051a:	89 45 fc	mov	%eax,-0x4(%rbp)	
40051d:	48 8b 05 1c 0b 20 00	mov	0x200b1c(%rip),%rax	# 601040 <bufp0>
400524:	48 8b 15 25 0b 20 00	mov	0x200b25(%rip),%rdx	# 601050 <bufp1>
40052b:	8b 12	mov	(%rdx),%edx	
40052d:	89 10	mov	%edx,(%rax)	
40052f:	48 8b 05 1a 0b 20 00	mov	0x200b1a(%rip),%rax	# 601050 <bufp1>
400536:	8b 55 fc	mov	-0x4(%rbp),%edx	
400539:	89 10	mov	%edx,(%rax)	
40053b:	5d	pop	%rbp	
40053c:	c3	retq		
40053d:	0f 1f 00	nopl	(%rax)	



# Main– Before Relocation

Disassembly of section .text:

```
0000000000000000 <main>:
   9:  e8 00 00 00 00      callq  e <main+0xe>
                                a: R_386_PC32  <swap>
   e:  b8 00 00 00 00      mov     $0x0,%eax
```

PC-relative Address:

$$\text{ADDR}(\text{swap}) - \text{PC} = 0x400502 - 0x4004fb \\ = 0x000007$$

# Main – After Relocation

```
00000000004004ed <main>:
4004f6: e8 07 00 00 00      callq  400502 <swap>
4004fb: b8 00 00 00 00      mov     $0x0,%eax
```

```
0000000000400502 <swap>:
400502: 55                  push   %rbp
400503: 48 89 e5            mov     %rsp,%rbp
...
```

# Swap – Before Relocation

```

0000000000000000 <swap>:
  0:   55                push    %rbp
  1:   48 89 e5          mov     %rsp,%rbp
  4:   48 c7 05 00 00 00 00  movq   $0x0,0x0(%rip)    # f <swap+0xf>
                                7: R_386_PC32          bufp1      -0x8

  b:   00 00 00 00
                                b: R_386_32S buf+0x4
  f:   48 8b 05 00 00 00 00  mov     0x0(%rip),%rax    # 16 <swap+0x16>

```

# Swap – After Relocation

```

0000000000400502 <swap>:
400502:   55                push    %rbp
400503:   48 89 e5          mov     %rsp,%rbp
400506:   48 c7 05 3f 0b 20 00  movq   $0x60103c,0x200b3f(%rip) #601050 <bufp1>
40050d:   3c 10 60 00
400511:   ...

```

Disassembly of section .bss:

```

0000000000601050 <bufp1>:

```

Disassembly of section .data:

```

0000000000601038 <buf>:

```

```

601038:   01 00
60103a:   00 00
. 60103c:   02 00

```

## PC-relative Address:

$\text{ADDR}(\text{bufp1}) - \text{PC} = 0x601050 - 0x400511$   
 $= 0x200b3f$

## Absolute Address:

$\text{ADDR}(\text{buf}) + 0x4 = 0x601038 + 0x4$   
 $= 0x60103c$

# Swap – Before Relocation

```

000000000000000000 <swap>:
  f:   48 8b 05 00 00 00 00    mov     0x0(%rip),%rax          # 16 <swap+0x16>
                                12: R_386_PC32      bufp0      -0x4
 16:   8b 00                    mov     (%rax),%eax

```

# Swap – After Relocation

```

0000000000400502 <swap>:
 400511:  48 8b 05 28 0b 20 00    mov     0x200b28(%rip),%rax      # 601040 <bufp0>
 400518:  ...                    (next instruction)

```

Disassembly of section .data:

```
0000000000601040 <bufp0>:
```

## PC-relative Address:

$$\begin{aligned}
 \text{ADDR(bufp0)} - \text{PC} &= 0x601040 - 0x400518 \\
 &= 0x200b28
 \end{aligned}$$

# Swap – Before Relocation

```

0000000000000000 <swap>:
 16:      8b 00                mov     (%rax),%eax
 18:      89 45 fc            mov     %eax,-0x4(%rbp)
1b:      48 8b 05 00 00 00 00  mov     0x0(%rip),%rax      # 22 <swap+0x22>
                                1e: R_386_PC32      bufp0      -0x4
 22:      48 8b 15 00 00 00 00  mov     0x0(%rip),%rdx

```

# Swap – After Relocation

```

0000000000400502 <swap>:
 400518:      8b 00                mov     (%rax),%eax
 40051a:      89 45 fc            mov     %eax,-0x4(%rbp)
 40051d:      48 8b 05 1c 0b 20 00  mov     0x200b1c(%rip),%rax  # 601040 <bufp0>
 400524:      ...                  (next instruction)

```

Disassembly of section .data:

```
0000000000601040 <bufp0>:
```

## PC-relative Address:

$$\begin{aligned}
 \text{ADDR(bufp0)} - \text{PC} &= 0x601040 - 0x400524 \\
 &= 0x200b1c
 \end{aligned}$$

## Swap – Before Relocation

```

0000000000000000 <swap>:
 22:  48 8b 15 00 00 00 00    mov     0x0(%rip),%rdx        # 29 <swap+0x29>
                               25: R_386_PC32      bufp1      -0x4
 29:  8b 12                  mov     (%rdx),%edx

```

## Swap – After Relocation

```

0000000000400502 <swap>:
 400524: 48 8b 15 25 0b 20 00    mov     0x200b25(%rip),%rdx    # 601050 <bufp1>
 40052b: ...                    (next instruction)

```

Disassembly of section .bss:

```
0000000000601050 <bufp1>:
```

## PC-relative Address:

$$\begin{aligned}
 \text{ADDR(bufp1)} - \text{PC} &= 0x601050 - 0x40052b \\
 &= 0x200b25
 \end{aligned}$$

## Swap – Before Relocation

```

0000000000000000 <swap>:
 2d:  48 8b 05 00 00 00 00    mov     0x0(%rip),%rax      # 34 <swap+0x34>
                                30: R_386_PC32      bufp1      -0x4
 34:  8b 55 fc                mov     -0x4(%rbp),%edx

```

## Swap – After Relocation

```

0000000000400502 <swap>:
 40052f: 48 8b 05 1a 0b 20 00    mov     0x200b1a(%rip),%rax      # 601050 <bufp1>
 400536:  ...                    (next instruction)

```

Disassembly of section .bss:

```

0000000000601050 <bufp1>:

```

## PC-relative Address:

$$\begin{aligned}
 \text{ADDR(bufp1)} - \text{PC} &= 0x601050 - 0x400536 \\
 &= 0x200b1a
 \end{aligned}$$