Link 8.B Dynamic Linking

Young W. Lim

2019-02-09 Sat

Outline

1. Based on
2. example codes
   - example 1 : vector addition and multiplication
   - example 2 : swap
   - example 3 : nested functions
3. pic and non-pic
   - pie enabled by defualt in gcc
   - relocatable object file `swap.o`
   - executable object file `swap.out` with static linking
   - executable object file `swap.out` with dynamic linking
4. relocation information
   - relocation information for example 2
   - relocation information for example 3
5. when to use dynamic linking and static linking

Based on

1. https://stac47.github.io/c/relocation/elf/tutorial/2018/03/01/understanding-relocation-elf.html

Compling 32-bit program on 64-bit gcc

- gcc -v
- gcc -m32 t.c
- sudo apt-get install gcc-multilib
- sudo apt-get install g++-multilib
- gcc-multilib
- g++-multilib
- gcc -m32
- objdump -m i386

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];

}
```

main.c

```
/*::::: 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]);

}
```

compiling commands

```
gcc -g -m32 -Wall -fPIC -c addvec.c
gcc -g -m32 -Wall -fPIC -c multvec.c
gcc -g -m32 -shared -o libvector.so addvec.o multvec.o

gcc -g -m32 -Wall -c main.c
gcc -g -m32 -o dynamicp main.o ./libvecotr.so

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./
export LD_LIBRARY_PATH
```

analyzing commands

- ```
  $ readelf --segments nmain_dyn.out
  $ objdump -d -s dynamicp
  $ objdump -d -j .plt.got dynamicp
  $ objdump -d -j .plt.got dynamicp
  $ gdb ... disas, x/a 0x...., c
  $ cat /proc/<pid>/map
  ```
  ```

swap.c

```
/*::::: swap.c ::::::::::::::::::::::::::*/
extern int buf[];

int *p0 = &buf[0];
int *p1;

void swap()
{
  int tmp;

  p1 = &buf[1];

  tmp = *p0;
  *p0 = *p1;
  *p1 = tmp;

}
```

main.c

```
/*::::: main.c ::::::::::::::::::::::::::::::*/
void swap();

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

int main()
{
  swap();

  return 0;
}
```

compiling commands for static linking

- ```
  gcc -m32 -Wall -c swap.c
  ar rcs libswap.a swap.o

  gcc -m32 -Wall -c main.c
  gcc -m32 -static -o swap.out main.o ./libswap.a
  ```
  ```

compiling commands for dynamic linking

```
gcc -m32 -Wall -fPIC -c swap.c -o swap_pic.o
gcc -shared -m32 -o libswap.so swap_pic.o

gcc -m32 -Wall -c main.c
gcc -m32 -o swap_dyn.out main.o ./libswap.so

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./
export LD_LIBRARY_PATH
```

analyzing commands

- ```
  $ readelf --segments swap_dyn.out
  $ objdump -d -s swap_dyn.out
  $ objdump -d -j .plt.got swap_dyn.out
  $ objdump -d -j .plt.got swap_dyn.out
  $ gdb ... disas, x/a 0x...., c
  $ cat /proc/<pid>/map
  ```
  ```

func1.c, func2.c, main.c

```
/*::::: func1.c ::::::::::::::::::::::::::*/
void func2();


void func1() {
  func2();
}

/*::::: func2.c ::::::::::::::::::::::::::*/
void func2() {

}

/*::::: main.c  ::::::::::::::::::::::::::*/
void func1();


int main() {
  func1();
}
```

compiling commands for static linking

- ```
  gcc -m32 -Wall -c func1.c
  gcc -m32 -Wall -c func2.c
  ar rcs libfunc.a func1.o func2.o

  gcc -m32 -Wall -c main.c
  gcc -m32 -static -o nest.out main.o ./libfunc.a
  ```
  ```

compiling commands for dynamic linking

```
gcc -fPIC -m32 -Wall -c func1.c -o func1_pic.o
gcc -fPIC -m32 -Wall -c func2.c -o func2_pic.o
gcc -shared -m32 -o libfunc.so func1_pic.o func2_pic.o

gcc -m32 -Wall -c main.c
gcc -m32 -o nest_dyn.out main.o ./libfunc.so
```

analyzing commands

- ```
  $ readelf --segments swap_dyn.out
  $ objdump -d -s swap_dyn.out
  $ objdump -d -j .plt.got swap_dyn.out
  $ objdump -d -j .plt.got swap_dyn.out
  $ gdb ... disas, x/a 0x...., c
  $ cat /proc/<pid>/map
  ```
  ```

pie and ssp enabled by default in gcc

- Arch now enables PIE and SSP by default in gcc and clang
- SSP and PIE are now enabled in gcc and clang in the stable repos.
- These changes will make it harder to exploit potential security holes in binaries built with these compilers.

https://www.reddit.com/r/archlinux/comments/6n5tkp/arch_now_enables_pie_and_ssp_by

randomization

- The reason for building applications as position-independent is to allow the application to be loaded at a random address;
- normally the kernel loads all executables to the same fixed address. Randomising this address makes it harder for an attacker to exploit the executable, since it is harder to know where the code (and heap) reside.

https://www.reddit.com/r/archlinux/comments/6n5tkp/arch_now_enables_pie_and_ssp_by

pic vs pie

- `-fpic` (position independent code)
  Generate position-independent code (PIC) suitable
  for use in a shared library. . .

- `-fpie` (position independent executables)
  These options are similar to -fpic and -fPIC,
  bt generated position independent code
  can be only linked into executables . . . .

`https://www.reddit.com/r/archlinux/comments/6n5tkp/arch_now_enables_pie_and_ssp_by`

swap.o using -fno-pic (1)

```
00000000 <swap>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 ec 10                sub    $0x10,%esp
   6:   c7 05 00 00 00 00 04    movl   $0x4,0x0
   d:   00 00 00
                        8: R_386_32     p1
                        c: R_386_32     buf
  10:   a1 00 00 00 00          mov    0x0,%eax
                        11: R_386_32    p0
  15:   8b 00                   mov    (%eax),%eax
  17:   89 45 fc                mov    %eax,-0x4(%ebp)
  1a:   8b 15 00 00 00 00       mov    0x0,%edx
                        1c: R_386_32    p1
```

swap.o using -fno-pic (2)

```
20:    a1 00 00 00 00           mov    0x0,%eax
                        21: R_386_32    p0
25:    8b 12                    mov    (%edx),%edx
27:    89 10                    mov    %edx,(%eax)
29:    a1 00 00 00 00           mov    0x0,%eax
                        2a: R_386_32    p1
2e:    8b 55 fc                 mov    -0x4(%ebp),%edx
31:    89 10                    mov    %edx,(%eax)
33:    90                       nop
34:    c9                       leave
35:    c3                       ret
```

swap.o using -fPIC (1)

```
00000000 <swap>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 ec 10                sub    $0x10,%esp
   6:   e8 fc ff ff ff          call   7 <swap+0x7>
                        7: R_386_PC32   __x86.get_pc_thunk.ax
   b:   05 01 00 00 00          add    $0x1,%eax
                        c: R_386_GOTPC  _GLOBAL_OFFSET_TABLE_
  10:   8b 90 00 00 00 00       mov    0x0(%eax),%edx
                        12: R_386_GOT32X        p1
  16:   8b 88 00 00 00 00       mov    0x0(%eax),%ecx
                        18: R_386_GOT32X        buf
  1c:   8d 49 04                lea    0x4(%ecx),%ecx
  1f:   89 0a                   mov    %ecx,(%edx)
  21:   8b 90 00 00 00 00       mov    0x0(%eax),%edx
                        23: R_386_GOT32X        p0
```

swap.o using -fPIC (2)

```
27:   8b 12                   mov    (%edx),%edx
29:   8b 12                   mov    (%edx),%edx
2b:   89 55 fc                mov    %edx,-0x4(%ebp)
2e:   8b 90 00 00 00 00       mov    0x0(%eax),%edx
                      30: R_386_GOT32X        p1
34:   8b 0a                   mov    (%edx),%ecx
36:   8b 90 00 00 00 00       mov    0x0(%eax),%edx
                      38: R_386_GOT32X        p0
3c:   8b 12                   mov    (%edx),%edx
3e:   8b 09                   mov    (%ecx),%ecx
40:   89 0a                   mov    %ecx,(%edx)
42:   8b 80 00 00 00 00       mov    0x0(%eax),%eax
                      44: R_386_GOT32X        p1
48:   8b 00                   mov    (%eax),%eax
4a:   8b 55 fc                mov    -0x4(%ebp),%edx
4d:   89 10                   mov    %edx,(%eax)
4f:   90                      nop
50:   c9                      leave
51:   c3                      ret
```

swap.o using -fPIC (3)

```
Desensamblado de la sección .text.__x86.get_pc_thunk.ax:

00000000 <__x86.get_pc_thunk.ax>:
   0:   8b 04 24                mov    (%esp),%eax
   3:   c3                      ret
```

swap.o without -fno-pic nor -fPIC (1)

```
00000000 <swap>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 ec 10                sub    $0x10,%esp
   6:   e8 fc ff ff ff          call   7 <swap+0x7>
                        7: R_386_PC32   __x86.get_pc_thunk.ax
   b:   05 01 00 00 00          add    $0x1,%eax
                        c: R_386_GOTPC  _GLOBAL_OFFSET_TABLE_
  10:   8b 90 00 00 00 00       mov    0x0(%eax),%edx
                        12: R_386_GOT32X        p1
  16:   8b 88 00 00 00 00       mov    0x0(%eax),%ecx
                        18: R_386_GOT32X        buf
  1c:   8d 49 04                lea    0x4(%ecx),%ecx
  1f:   89 0a                   mov    %ecx,(%edx)
  21:   8b 90 00 00 00 00       mov    0x0(%eax),%edx
                        23: R_386_GOTOFF        p0
```

swap.o without -fno-pic nor -fPIC (2)

```
  27:   8b 12                   mov    (%edx),%edx
  29:   89 55 fc                mov    %edx,-0x4(%ebp)
  2c:   8b 90 00 00 00 00       mov    0x0(%eax),%edx
                        2e: R_386_GOT32X        p1
  32:   8b 0a                   mov    (%edx),%ecx
  34:   8b 90 00 00 00 00       mov    0x0(%eax),%edx
                        36: R_386_GOTOFF        p0
  3a:   8b 09                   mov    (%ecx),%ecx
  3c:   89 0a                   mov    %ecx,(%edx)
  3e:   8b 80 00 00 00 00       mov    0x0(%eax),%eax
                        40: R_386_GOT32X        p1
  44:   8b 00                   mov    (%eax),%eax
  46:   8b 55 fc                mov    -0x4(%ebp),%edx
  49:   89 10                   mov    %edx,(%eax)
  4b:   90                      nop
  4c:   c9                      leave
  4d:   c3                      ret

Desensamblado de la sección .text.__x86.get_pc_thunk.ax:

00000000 <__x86.get_pc_thunk.ax>:
   0:   8b 04 24                mov    (%esp),%eax
   3:   c3                      ret
```

swap.o without -fno-pic nor -fPIC (3)

```
Desensamblado de la sección .text.__x86.get_pc_thunk.ax:

00000000 <__x86.get_pc_thunk.ax>:
   0:   8b 04 24                mov    (%esp),%eax
   3:   c3                      ret
```

swap.out using -fno-pic

```
080488d5 <swap>:
 80488d5:       55                      push   %ebp
 80488d6:       89 e5                   mov    %esp,%ebp
 80488d8:       83 ec 10                sub    $0x10,%esp
 80488db:       c7 05 c4 ac 0d 08 6c    movl   $0x80d906c,0x80dacc4
 80488e2:       90 0d 08
 80488e5:       a1 70 90 0d 08          mov    0x80d9070,%eax
 80488ea:       8b 00                   mov    (%eax),%eax
 80488ec:       89 45 fc                mov    %eax,-0x4(%ebp)
 80488ef:       8b 15 c4 ac 0d 08       mov    0x80dacc4,%edx
 80488f5:       a1 70 90 0d 08          mov    0x80d9070,%eax
 80488fa:       8b 12                   mov    (%edx),%edx
 80488fc:       89 10                   mov    %edx,(%eax)
 80488fe:       a1 c4 ac 0d 08          mov    0x80dacc4,%eax
 8048903:       8b 55 fc                mov    -0x4(%ebp),%edx
 8048906:       89 10                   mov    %edx,(%eax)
 8048908:       90                      nop
 8048909:       c9                      leave
 804890a:       c3                      ret
 804890b:       66 90                   xchg   %ax,%ax
 804890d:       66 90                   xchg   %ax,%ax
 804890f:       90                      nop
```

swap.out using -fPIC (1)

```
080488d5 <swap>:
 80488d5:       55                      push   %ebp
 80488d6:       89 e5                   mov    %esp,%ebp
 80488d8:       83 ec 10                sub    $0x10,%esp
 80488db:       e8 f1 ff ff ff          call   80488d1 <__x86.get_pc_thunk.ax>
 80488e0:       05 20 07 09 00          add    $0x90720,%eax
 80488e5:       c7 c2 c4 ac 0d 08       mov    $0x80dacc4,%edx
 80488eb:       c7 c1 68 90 0d 08       mov    $0x80d9068,%ecx
 80488f1:       8d 49 04                lea    0x4(%ecx),%ecx
 80488f4:       89 0a                   mov    %ecx,(%edx)
 80488f6:       c7 c2 70 90 0d 08       mov    $0x80d9070,%edx
 80488fc:       8b 12                   mov    (%edx),%edx
 80488fe:       8b 12                   mov    (%edx),%edx
 8048900:       89 55 fc                mov    %edx,-0x4(%ebp)
 8048903:       c7 c2 c4 ac 0d 08       mov    $0x80dacc4,%edx
 8048909:       8b 0a                   mov    (%edx),%ecx
 804890b:       c7 c2 70 90 0d 08       mov    $0x80d9070,%edx
```

swap.out using -fPIC (2)

```
8048911:       8b 12                   mov    (%edx),%edx
8048913:       8b 09                   mov    (%ecx),%ecx
8048915:       89 0a                   mov    %ecx,(%edx)
8048917:       c7 c0 c4 ac 0d 08       mov    $0x80dacc4,%eax
804891d:       8b 00                   mov    (%eax),%eax
804891f:       8b 55 fc                mov    -0x4(%ebp),%edx
8048922:       89 10                   mov    %edx,(%eax)
8048924:       90                      nop
8048925:       c9                      leave
8048926:       c3                      ret
8048927:       66 90                   xchg   %ax,%ax
8048929:       66 90                   xchg   %ax,%ax
804892b:       66 90                   xchg   %ax,%ax
804892d:       66 90                   xchg   %ax,%ax
804892f:       90                      nop
```

swap.out without -fno-pic nor -fPIC (1)

```
080488d5 <swap>:
 80488d5:       55                      push   %ebp
 80488d6:       89 e5                   mov    %esp,%ebp
 80488d8:       83 ec 10                sub    $0x10,%esp
 80488db:       e8 f1 ff ff ff          call   80488d1 <__x86.get_pc_thunk.ax>
 80488e0:       05 20 07 09 00          add    $0x90720,%eax
 80488e5:       c7 c2 c4 ac 0d 08       mov    $0x80dacc4,%edx
 80488eb:       c7 c1 68 90 0d 08       mov    $0x80d9068,%ecx
 80488f1:       8d 49 04                lea    0x4(%ecx),%ecx
 80488f4:       89 0a                   mov    %ecx,(%edx)
 80488f6:       8b 90 70 00 00 00       mov    0x70(%eax),%edx
 80488fc:       8b 12                   mov    (%edx),%edx
 80488fe:       89 55 fc                mov    %edx,-0x4(%ebp)
 8048901:       c7 c2 c4 ac 0d 08       mov    $0x80dacc4,%edx
 8048907:       8b 0a                   mov    (%edx),%ecx
 8048909:       8b 90 70 00 00 00       mov    0x70(%eax),%edx
 804890f:       8b 09                   mov    (%ecx),%ecx
```

swap.out without -fno-pic nor -fPIC (2)

```
8048911:       89 0a                    mov    %ecx,(%edx)
8048913:       c7 c0 c4 ac 0d 08        mov    $0x80dacc4,%eax
8048919:       8b 00                    mov    (%eax),%eax
804891b:       8b 55 fc                 mov    -0x4(%ebp),%edx
804891e:       89 10                    mov    %edx,(%eax)
8048920:       90                       nop
8048921:       c9                       leave
8048922:       c3                       ret
8048923:       66 90                    xchg   %ax,%ax
8048925:       66 90                    xchg   %ax,%ax
8048927:       66 90                    xchg   %ax,%ax
8048929:       66 90                    xchg   %ax,%ax
804892b:       66 90                    xchg   %ax,%ax
804892d:       66 90                    xchg   %ax,%ax
804892f:       90                       nop
```

swap.out using -fno-pic

```
00000480 <swap@plt>:
 480:   ff a3 10 00 00 00       jmp    *0x10(%ebx)
 486:   68 08 00 00 00          push   $0x8
 48b:   e9 d0 ff ff ff          jmp    460 <.plt>
```

swap.out using -fPIC

```
00000480 <swap@plt>:
 480:   ff a3 10 00 00 00       jmp    *0x10(%ebx)
 486:   68 08 00 00 00          push   $0x8
 48b:   e9 d0 ff ff ff          jmp    460 <.plt>
```

swap.out without -fno-pic nor -fPIC

```
00000480 <swap@plt>:
 480:   ff a3 10 00 00 00       jmp    *0x10(%ebx)
 486:   68 08 00 00 00          push   $0x8
 48b:   e9 d0 ff ff ff          jmp    460 <.plt>
```

relocation information in addvec.o

objdump -dr addvec.o

```
7:   e8 fc ff ff ff          call   8 <addvec+0x8>
                     8: R_386_PC32   __x86.get_pc_thunk.ax
c:   05 01 00 00 00          add    $0x1,%eax
                     d: R_386_GOTPC  _GLOBAL_OFFSET_TABLE_
```

readelf -r addvec.o

```
 Offset     Info    Type            Sym.Value  Sym. Name
00000008  00001002 R_386_PC32        00000000   __x86.get_pc_thunk.ax
0000000d  0000110a R_386_GOTPC       00000000   _GLOBAL_OFFSET_TABLE_
```

relocation information in multvec.o

objdump -dr multvec.o

```
7:   e8 fc ff ff ff          call   8 <multvec+0x8>
                     8: R_386_PC32   __x86.get_pc_thunk.ax
c:   05 01 00 00 00          add    $0x1,%eax
                     d: R_386_GOTPC  _GLOBAL_OFFSET_TABLE_
```

readelf -r multvec.o

```
 Offset     Info    Type            Sym.Value  Sym. Name
00000008  00001002 R_386_PC32        00000000   __x86.get_pc_thunk.ax
0000000d  0000110a R_386_GOTPC       00000000   _GLOBAL_OFFSET_TABLE_
```

relocation information in main.o (1)

objdump -dr main.o

```
 f:   e8 fc ff ff ff          call   10 <main+0x10>
                      10: R_386_PC32  __x86.get_pc_thunk.bx
14:   81 c3 02 00 00 00       add    $0x2,%ebx
                      16: R_386_GOTPC _GLOBAL_OFFSET_TABLE_
1c:   8b 83 00 00 00 00       mov    0x0(%ebx),%eax
                      1e: R_386_GOT32X        z
23:   8d 83 00 00 00 00       lea    0x0(%ebx),%eax
                      25: R_386_GOTOFF        y
2a:   8d 83 00 00 00 00       lea    0x0(%ebx),%eax
                      2c: R_386_GOTOFF        x
```

readelf -r main.o

```
 Offset     Info    Type            Sym.Value  Sym. Name
00000010  00001402 R_386_PC32        00000000   __x86.get_pc_thunk.bx
00000016  0000150a R_386_GOTPC       00000000   _GLOBAL_OFFSET_TABLE_
0000001e  0000122b R_386_GOT32X      00000004   z
00000025  00001109 R_386_GOTOFF      00000008   y
0000002c  00001009 R_386_GOTOFF      00000000   x
```

relocation information in main.o (2)

objdump -dr main.o

```
31:   e8 fc ff ff ff          call   32 <main+0x32>
                      32: R_386_PLT32 addvec
39:   8b 83 00 00 00 00       mov    0x0(%ebx),%eax
                      3b: R_386_GOT32X        z
42:   8b 83 00 00 00 00       mov    0x0(%ebx),%eax
                      44: R_386_GOT32X        z
4f:   8d 83 00 00 00 00       lea    0x0(%ebx),%eax
                      51: R_386_GOTOFF        .rodata
56:   e8 fc ff ff ff          call   57 <main+0x57>
                      57: R_386_PLT32 printf
```

readelf -r main.o

```
 Offset     Info    Type            Sym.Value  Sym. Name
00000010  00001402 R_386_PC32        00000000   __x86.get_pc_thunk.bx
00000016  0000150a R_386_GOTPC       00000000   _GLOBAL_OFFSET_TABLE_
0000001e  0000122b R_386_GOT32X      00000004   z
00000025  00001109 R_386_GOTOFF      00000008   y
0000002c  00001009 R_386_GOTOFF      00000000   x
00000032  00001604 R_386_PLT32       00000000   addvec
0000003b  0000122b R_386_GOT32X      00000004   z
00000044  0000122b R_386_GOT32X      00000004   z
```

which liking method

https://www.ibm.com/support/knowledgecenter/en/ssw_aix_71
/com.ibm.aix.performance/when_dyn_linking_static_linking.htm

https://www.ibm.com/support/knowledgecenter/en/ssw_aix_71/com.ibm.aix.performance/