

Link 6.B Loading

Young W. Lim

2019-01-26 Sat

Outline

- 1 Based on
- 2 linker script example
- 3 some links
- 4 linker script example II
- 5 linker script example III
- 6 linker script example IV

"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.

Compiling 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`

example code (1) mymain.c mymain2.c

```
//----- mymain.c -----
```

```
void mymain(void)
```

```
{
```

```
    int a= 77;
```

```
    a++;
```

```
}
```

```
//----- mymain2.c -----
```

```
void mymain(void)
```

```
{
```

```
    int a= 77;
```

```
    a++;
```

```
    asm volatile("mov $1,%%eax; mov %0,%%ebx; int $0x80" : : "r"(a) : "%eax" );
```

```
}
```

<https://stackoverflow.com/questions/7182409/how-to-correctly-use-a-simple-linker->

example code (2) script

```
//----- script -----  
OUTPUT_FORMAT("elf32-i386", "elf32-i386",  
              "elf32-i386")  
OUTPUT_ARCH(i386)  
  
ENTRY(mymain)  
  
SECTIONS  
{  
    . = 0x10000;  
    .text : { *(.text) }  
    . = 0x8000000;  
    .data : { *(.data) }  
    .bss : { *(.bss) }  
}
```

<https://stackoverflow.com/questions/7182409/how-to-correctly-use-a-simple-linker->

results (1)

```
$ gcc -m32 -g -c mymain.c
$ ld -m elf_i386 -T script -o mymain.out mymain.o
$ ./mymain.out
Violación de segmento ('core' generado)
```

<https://stackoverflow.com/questions/7182409/how-to-correctly-use-a-simple-linker->

results (2)

```
$ gdb ./mymain.out
...
(gdb) break mymain
Punto de interrupción 1 at 0x10010: file mymain.c, line 3.
(gdb) run
Starting program: /home/young/lds/mymain.out

Breakpoint 1, mymain () at mymain.c:3
3      int a= 77;
(gdb) si
4      a++;
(gdb) si
6      }
(gdb) si
0x0001001c      6      }
(gdb) si
0x0001001d      6      }
(gdb) si
0x00000001 in ?? ()
(gdb) si
```

<https://stackoverflow.com/questions/7182409/how-to-correctly-use-a-simple-linker->

results (3)

```
(gdb) disas mymain
```

```
Dump of assembler code for function mymain:
```

```
0x00010000 <+0>:    push   %ebp
0x00010001 <+1>:    mov    %esp,%ebp
0x00010003 <+3>:    sub   $0x10,%esp
0x00010006 <+6>:    call  0x1001e <__x86.get_pc_thunk.ax>
0x0001000b <+11>:   add   $0x7fefff5,%eax
0x00010010 <+16>:   movl  $0x4d,-0x4(%ebp)
0x00010017 <+23>:   addl  $0x1,-0x4(%ebp)
0x0001001b <+27>:   nop
0x0001001c <+28>:   leave
0x0001001d <+29>:   ret
```

```
End of assembler dump.
```

<https://stackoverflow.com/questions/7182409/how-to-correctly-use-a-simple-linker->

results (4)

```
$ gcc -m32 -g -o mymain.out mymain.c  
/usr/lib/gcc/x86_64-linux-gnu/7/../../../../lib32/Scrt1.o: En la función ‘_start’:  
(.text+0x28): referencia a ‘main’ sin definir  
collect2: error: ld returned 1 exit status
```

<https://stackoverflow.com/questions/7182409/how-to-correctly-use-a-simple-linker->

results (5)

```
$ gcc -m32 -g -c mymain2.c
$ !ld:p
ld -m elf_i386 -T script -o mymain.out mymain.o
$ ld -m elf_i386 -T script -o mymain2.out mymain2.o
$ ./mymain2.out
$ echo $?
78
```

<https://stackoverflow.com/questions/7182409/how-to-correctly-use-a-simple-linker->

results (6)

```
(gdb) break mymain
```

```
Punto de interrupción 1 at 0x10010: file mymain2.c, line 3.
```

```
(gdb) run
```

```
Starting program: /home/young/lds/mymain2.out
```

```
Breakpoint 1, mymain () at mymain2.c:3
```

```
3         int a= 77;
```

```
(gdb) si
```

```
4         a++;
```

```
(gdb) si
```

```
6         asm volatile("mov $1,%%eax; mov %0,%%ebx; int $0x80" : : "r"(a) : "%eax")
```

```
(gdb) si
```

```
0x0001001e      6         asm volatile("mov $1,%%eax; mov %0,%%ebx; int $0x80" : :
```

```
(gdb) si
```

```
0x00010023      6         asm volatile("mov $1,%%eax; mov %0,%%ebx; int $0x80" : :
```

```
(gdb) si
```

```
0x00010025      6         asm volatile("mov $1,%%eax; mov %0,%%ebx; int $0x80" : :
```

```
(gdb) si
```

```
[Inferior 1 (process 8359) exited with code 0116]
```

```
(gdb) si
```

```
Este programa no está corriendo.
```

```
(gdb) si
```

```
Este programa no está corriendo.
```

```
(gdb)
```

<https://stackoverflow.com/questions/7182409/how-to-correctly-use-a-simple-linker->

<https://www.computerhope.com/unix/uld.htm>

<http://www.hertaville.com/a-sample-linker-script.html>

https://docs.rtems.org/releases/rtemsdocs-4.6.2/share/rtems/html/bsp_howto/bsp_howto.html

<http://www.bravegnu.org/gnu-eprog/lds.html>

https://wiki.osdev.org/Linker_Scripts

<https://sourceware.org/binutils/docs/ld/Simple-Example.html>

example code (1)

SECTIONS

```
{
    .text :
    {
        *(.text)
    } > REGION_TEXT

    .rodata :
    {
        *(.rodata)
        rodata_end = .;
    } > REGION_RODATA <===== PLACE 1

    .data : AT (rodata_end) <===== PLACE 2
    {
        data_start = .;
        *(.data)
    } > REGION_DATA <===== PLACE 3

    data_size = SIZEOF(.data);
    data_load_start = LOADADDR(.data);

    .bss :
    {
```

```
MEMORY
{
    ROM : ORIGIN = 0, LENGTH = 2M           /*0M ~ 2M*/
    ROM2 : ORIGIN = 0x10000000, LENGTH = 1M /*256M ~ 257M*/
    RAM : ORIGIN = 0x20000000, LENGTH = 1M  /*512M ~ 513M*/
}

REGION_ALIAS("REGION_TEXT", ROM);          /*0M ~ 2M*/
REGION_ALIAS("REGION_RODATA", ROM2);      /*256M ~ 257M*/
REGION_ALIAS("REGION_DATA", RAM);         /*512M ~ 513M*/
REGION_ALIAS("REGION_BSS", RAM);          /*512M ~ 513M*/
```

<https://stackoverflow.com/questions/38298184/how-to-understand-such-sample-in-gnu>

example code (1)

```
ld -r -o t.o t1.o t2.o --verbose > /tmp/script
```

```
.text      0 :  
{  
  *(.text .stub)  
  /* .gnu.warning sections are handled specially by elf32.em. */  
  *(.gnu.warning)  
}
```

```
.text      0 :  
{  
  *(.text .stub .text.*)  
  /* .gnu.warning sections are handled specially by elf32.em. */  
  *(.gnu.warning)  
}
```

```
ld -r -o t.o t1.o t2.o -T /tmp/script
```

<https://stackoverflow.com/questions/29701441/merge-sections-with-ld>

example code (1)

SECTIONS

```
{
    .text :
    {
        *(.text)
    } > REGION_TEXT

    .rodata :
    {
        *(.rodata)
        rodata_end = .;
    } > REGION_RODATA <===== PLACE 1

    .data : AT (rodata_end) <===== PLACE 2
    {
        data_start = .;
        *(.data)
    } > REGION_DATA <===== PLACE 3

    data_size = SIZEOF(.data);
    data_load_start = LOADADDR(.data);

    .bss :
    {
```