Seguridad en Sistemas Operativos (2017-2018)
Grado en Ingeniería Informática
Universidad de Granada

Práctica 2 Sesión 1

UNIVERSIDAD DE GRANADA

Iván Rodríguez Millán
ivanrodmil@gmail.com

Índice

Índice de figuras

Índice de tablas

1. Ejercicio 1: Construye y compila un programa simple, por ejemplo similar al Hola mundo, en dos versiones, C y C++. (A) Consulta los manuales, o en Internet que contienen las secciones .interp, .got, got.ptl. (B) Compara los ELFs de las dos versiones listando las secciones ¿Hay alguna diferencia?¿Qué contienen las secciones .ctors y .dtors?. (C) Con la opción readelf -r podemos ver las secciones de reubicación. Indicar que contienen.

Apartado (A):

La sección .interp contiene el nombre de la ruta del programa intérprete. [1]

La sección .got contiene la Global offset table. [2] Que no es más que una tabla de datos usada por los programas en tiempo de ejecución para encontrar las direcciones de las variables globales desconocidas en tiempo de compilación. [**?**]

La sección goy.ptl contiene el GOT (Global offset table) para el .plt (Procedure Linkage Table), clásicamente estos datos formaban parte del .got. [3]

Apartado (B): Código C: Lo compilamos con la opción gcc holamundo.c -o holamundoC

```
#include<stdio.h>

main()
{
printf("Hola mundo\n");

}
```

Código C++: Lo compilamos con la opción g++ holamundo.cpp -o holamundoCPP

```
#include <iostream>

using namespace std;

int main(){

cout << "Hola mundo" << endl;

}
```

Para listar las secciones usamos la orden: readelf –sections holamundo +Ç"ó +ÇPP".

Para el fichero C nos sale lo siguente:

```
[root@ivancito P2S1]# readelf --sections holamundoC
Hay 29 encabezados de seccion, comenzando en el desplazamiento: 0x18a8:

Encabezados de Seccion:
[Nr] Nombre          Tipo            Direccion       Despl
Tama           TamEnt          Opts  Enl   Info  Alin
[ 0]                 NULL            0000000000000000 00000000
0000000000000000 0000000000000000        0     0     0
[ 1] .interp         PROGBITS        0000000000400238 00000238
000000000000001c 0000000000000000 A      0     0     1
[ 2] .note.ABI-tag   NOTE            0000000000400254 00000254
0000000000000020 0000000000000000 A      0     0     4
[ 3] .note.gnu.build-i NOTE          0000000000400274 00000274
0000000000000024 0000000000000000 A      0     0     4
[ 4] .gnu.hash       GNU_HASH        0000000000400298 00000298
000000000000001c 0000000000000000 A      5     0     8
[ 5] .dynsym         DYNSYM          00000000004002b8 000002b8
0000000000000060 0000000000000018 A      6     1     8
[ 6] .dynstr         STRTAB          0000000000400318 00000318
000000000000003d 0000000000000000 A      0     0     1
[ 7] .gnu.version    VERSYM          0000000000400356 00000356
```

```
0000000000000008 0000000000000002 A       5     0     2
[ 8] .gnu.version_r  VERNEED         0000000000400360 00000360
0000000000000020 0000000000000000 A       6     1     8
[ 9] .rela.dyn       RELA            0000000000400380 00000380
0000000000000030 0000000000000018 A       5     0     8
[10] .rela.plt       RELA            00000000004003b0 000003b0
0000000000000018 0000000000000018 AI      5    22     8
[11] .init           PROGBITS        00000000004003c8 000003c8
0000000000000017 0000000000000000 AX      0     0     4
[12] .plt            PROGBITS        00000000004003e0 000003e0
0000000000000020 0000000000000010 AX      0     0     16
[13] .text           PROGBITS        0000000000400400 00000400
0000000000000162 0000000000000000 AX      0     0     16
[14] .fini           PROGBITS        0000000000400564 00000564
0000000000000009 0000000000000000 AX      0     0     4
[15] .rodata         PROGBITS        0000000000400570 00000570
000000000000001b 0000000000000000 A       0     0     8
[16] .eh_frame_hdr   PROGBITS        000000000040058c 0000058c
0000000000000034 0000000000000000 A       0     0     4
[17] .eh_frame       PROGBITS        00000000004005c0 000005c0
00000000000000f4 0000000000000000 A       0     0     8
[18] .init_array     INIT_ARRAY      0000000000600e10 00000e10
0000000000000008 0000000000000008 WA      0     0     8
[19] .fini_array     FINI_ARRAY      0000000000600e18 00000e18
0000000000000008 0000000000000008 WA      0     0     8
[20] .dynamic        DYNAMIC         0000000000600e20 00000e20
00000000000001d0 0000000000000010 WA      6     0     8
[21] .got            PROGBITS        0000000000600ff0 00000ff0
0000000000000010 0000000000000008 WA      0     0     8
[22] .got.plt        PROGBITS        0000000000601000 00001000
0000000000000020 0000000000000008 WA      0     0     8
[23] .data           PROGBITS        0000000000601020 00001020
0000000000000004 0000000000000000 WA      0     0     1
[24] .bss            NOBITS          0000000000601024 00001024
0000000000000004 0000000000000000 WA      0     0     1
[25] .comment        PROGBITS        0000000000000000 00001024
000000000000002c 0000000000000001 MS      0     0     1
[26] .symtab         SYMTAB          0000000000000000 00001050
00000000000005a0 0000000000000018       27    43     8
[27] .strtab         STRTAB          0000000000000000 000015f0
00000000000001b5 0000000000000000        0     0     1
[28] .shstrtab       STRTAB          0000000000000000 000017a5
0000000000000103 0000000000000000        0     0     1
Key to Flags:
W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
L (link order), O (extra OS processing required), G (group), T (TLS),
C (compressed), x (unknown), o (OS specific), E (exclude),
l (large), p (processor specific)
}
```

[1] Para el fichero C++ nos sale lo siguente:

```
[root@ivancito P2S1]# readelf --sections holamundoCPP
Hay 29 encabezados de seccion, comenzando en el desplazamiento: 0x1b40:
Encabezados de Seccion:
[Nr] Nombre          Tipo            Direccion       Despl
Tama          TamEnt          Opts   Enl   Info  Alin
[ 0]                 NULL            0000000000000000 00000000
0000000000000000 0000000000000000        0     0     0
[ 1] .interp         PROGBITS        0000000000400238 00000238
000000000000001c 0000000000000000 A      0     0     1
[ 2] .note.ABI-tag   NOTE            0000000000400254 00000254
0000000000000020 0000000000000000 A      0     0     4
[ 3] .note.gnu.build-i NOTE          0000000000400274 00000274
0000000000000024 0000000000000000 A      0     0     4
[ 4] .gnu.hash       GNU_HASH        0000000000400298 00000298
0000000000000030 0000000000000000 A      5     0     8
[ 5] .dynsym         DYNSYM          00000000004002c8 000002c8
00000000000000f0 0000000000000018 A      6     1     8
[ 6] .dynstr         STRTAB          00000000004003b8 000003b8
0000000000000136 0000000000000000 A      0     0     1
[ 7] .gnu.version    VERSYM          00000000004004ee 000004ee
0000000000000014 0000000000000002 A      5     0     2
[ 8] .gnu.version_r  VERNEED         0000000000400508 00000508
0000000000000040 0000000000000000 A      6     2     8
[ 9] .rela.dyn       RELA            0000000000400548 00000548
0000000000000048 0000000000000018 A      5     0     8
[10] .rela.plt       RELA            0000000000400590 00000590
0000000000000090 0000000000000018 AI     5    22     8
[11] .init           PROGBITS        0000000000400620 00000620
0000000000000017 0000000000000000 AX     0     0     4
[12] .plt            PROGBITS        0000000000400640 00000640
0000000000000070 0000000000000010 AX     0     0     16
[13] .text           PROGBITS        00000000004006b0 000006b0
00000000000001d2 0000000000000000 AX     0     0     16
[14] .fini           PROGBITS        0000000000400884 00000884
0000000000000009 0000000000000000 AX     0     0     4
[15] .rodata         PROGBITS        0000000000400890 00000890
000000000000001c 0000000000000000 A      0     0     8
[16] .eh_frame_hdr   PROGBITS        00000000004008ac 000008ac
0000000000000044 0000000000000000 A      0     0     4
[17] .eh_frame       PROGBITS        00000000004008f0 000008f0
0000000000000134 0000000000000000 A      0     0     8
[18] .init_array     INIT_ARRAY      0000000000600dd8 00000dd8
0000000000000010 0000000000000008 WA     0     0     8
```

[1]La columna Tama en realidad es Tamaño pero se ha tenido que omitir la ñ para que lo aceptara el compilador de Latex, debido a que está dentro de una etiqueta lstisting. Siento las molestias.

```
[19] .fini_array     FINI_ARRAY     0000000000600de8 00000de8
0000000000000008 0000000000000008 WA     0     0     8
[20] .dynamic        DYNAMIC        0000000000600df0 00000df0
0000000000000200 0000000000000010 WA     6     0     8
[21] .got            PROGBITS       0000000000600ff0 00000ff0
0000000000000010 0000000000000008 WA     0     0     8
[22] .got.plt        PROGBITS       0000000000601000 00001000
0000000000000048 0000000000000008 WA     0     0     8
[23] .data           PROGBITS       0000000000601048 00001048
0000000000000004 0000000000000000 WA     0     0     1
[24] .bss            NOBITS         0000000000601060 0000104c
0000000000000118 0000000000000000 WA     0     0     32
[25] .comment        PROGBITS       0000000000000000 0000104c
000000000000002c 0000000000000001 MS     0     0     1
[26] .symtab         SYMTAB         0000000000000000 00001078
0000000000000690 0000000000000018       27    47     8
[27] .strtab         STRTAB         0000000000000000 00001708
0000000000000334 0000000000000000        0     0     1
[28] .shstrtab       STRTAB         0000000000000000 00001a3c
0000000000000103 0000000000000000        0     0     1
Key to Flags:
W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
L (link order), O (extra OS processing required), G (group), T (TLS),
C (compressed), x (unknown), o (OS specific), E (exclude),
l (large), p (processor specific)
```

Como diferencias vemos las siguientes:

- No tienen el mismo tamaño secciones como .gnu.hash, .dynsym, dynstr, .gnu.version, etc.
- En relación a la sección .gnu.version_r podemos se puede visualizar que en el programa escrito en C++ la info indica un 2 y en el programa escrito en C un 2.
- A partir de dynsym en adelante no coinciden las direcciones.

.ctors contiene una lista de punteros de funciones constructores globales. [1]

.dtors contiene una lista de punteros de funciones destructores globales. [1]

Apartado (C):

Para C:

```
[root@ivancito P2S1]# readelf -r holamundoC

La seccion de reubicacion '.rela.dyn' en el desplazamiento 0x380 contiene 2
    entradas:
Desplaz        Info          Tipo          Val. Simbolo Nom. Simbolo + Adend
000000600ff0 000200000006 R_X86_64_GLOB_DAT 0000000000000000
    __libc_start_main@GLIBC_2.2.5 + 0
000000600ff8 000300000006 R_X86_64_GLOB_DAT 0000000000000000 __gmon_start__ + 0

La seccion de reubicacion '.rela.plt' en el desplazamiento 0x3b0 contiene 1
    entradas:
Desplaz        Info          Tipo          Val. Simbolo Nom. Simbolo + Adend
000000601018 000100000007 R_X86_64_JUMP_SLO 0000000000000000 puts@GLIBC_2.2.5
    + 0
```

Para C++:

```
[root@ivancito P2S1]# readelf -r holamundoCPP

La seccion de reubicacion '.rela.dyn' en el desplazamiento 0x548 contiene 3
    entradas:
Desplaz        Info          Tipo          Val. Simbolo Nom. Simbolo + Adend
000000600ff0 000100000006 R_X86_64_GLOB_DAT 0000000000000000 __gmon_start__ + 0
000000600ff8 000300000006 R_X86_64_GLOB_DAT 0000000000000000
    __libc_start_main@GLIBC_2.2.5 + 0
000000601060 000900000005 R_X86_64_COPY  0000000000601060 _ZSt4cout@GLIBCXX_3.4
    + 0

La seccion de reubicacion '.rela.plt' en el desplazamiento 0x590 contiene 6
    entradas:
Desplaz        Info          Tipo          Val. Simbolo Nom. Simbolo + Adend
000000601018 000200000007 R_X86_64_JUMP_SLO 0000000000000000
    _ZNSt8ios_base4InitC1E@GLIBCXX_3.4 + 0
000000601020 000400000007 R_X86_64_JUMP_SLO 0000000000000000
    __cxa_atexit@GLIBC_2.2.5 + 0
000000601028 000800000007 R_X86_64_JUMP_SLO 0000000000400670
    _ZNSt8ios_base4InitD1E@GLIBCXX_3.4 + 0
000000601030 000500000007 R_X86_64_JUMP_SLO 0000000000000000
    _ZStlsISt11char_traits@GLIBCXX_3.4 + 0
000000601038 000600000007 R_X86_64_JUMP_SLO 0000000000000000
    _ZNSolsEPFRSoS_E@GLIBCXX_3.4 + 0
000000601040 000700000007 R_X86_64_JUMP_SLO 00000000004006a0
    _ZSt4endlIcSt11char_tr@GLIBCXX_3.4 + 0
```

Este comando muestra el contenido de la sección de reubicación del archivo correspon-

diente.

Para el fichero C tenemos la sección de reubicación de .rela.dyn con 2 entradas y la sección de reubicación de .rela.plt con 1 entrada.

Para el fichero C++ tenemos la sección de reubicación de .rela.dyn con 3 entradas y la sección de reubicación de .rela.plt con 6 entradas.

Hay algunas que son comunes entre el programa en C y el programa en C++, aunque para C++ se necesitan más reubicaciones como es previsible.

2. Ejercicio 2: Mira en el manual en línea o en internet las opciones de la orden objdump, e indica: (A) qué opciones nos permiten ver la información que nos suministra readelf. (B) qué otras opciones nos permite realizar objdump desde el punto de la ingeniería inversa.

Apartado A:

```
[root@ivancito P2S1]# objdump holamundo.c -P
objdump: option requires an argument -- 'P'
Modo de empleo: objdump <opcion(es)> <fichero(s)>
Muestra la informacion de <fichero(s)> objeto.
Se requiere por lo menos una de los siguientes opciones:
-a, --archive-headers  Display archive header information
-f, --file-headers     Display the contents of the overall file header
-p, --private-headers  Display object format specific file header contents
-P, --private=OPT,OPT... Display object format specific contents
-h, --[section-]headers Display the contents of the section headers
-x, --all-headers      Display the contents of all headers
-d, --disassemble      Display assembler contents of executable sections
-D, --disassemble-all  Display assembler contents of all sections
-S, --source           Intermix source code with disassembly
-s, --full-contents    Display the full contents of all sections requested
-g, --debugging        Display debug information in object file
-e, --debugging-tags   Display debug information using ctags style
-G, --stabs            Display (in raw form) any STABS info in the file
-W[lLiaprmfFsoRt] or
--dwarf[=rawline,=decodedline,=info,=abbrev,=pubnames,=aranges,=macro,=frames,
=frames-interp,=str,=loc,=Ranges,=pubtypes,
=gdb_index,=trace_info,=trace_abbrev,=trace_aranges,
=addr,=cu_index]
Display DWARF info in the file
-t, --syms             Display the contents of the symbol table(s)
-T, --dynamic-syms     Display the contents of the dynamic symbol table
-r, --reloc            Display the relocation entries in the file
-R, --dynamic-reloc    Display the dynamic relocation entries in the file
@<file>                Read options from <file>
-v, --version          Display this programs version number
-i, --info             List object formats and architectures supported
-H, --help             Display this information

Los siguientes interruptores son opcionales:
-b, --target=NOMBREBFD      Especifica el formato objeto objetivo
como NOMBREBFD
-m, --architecture=MaQUINA  Especifica la arquitectura objetivo
```

```
como MaQUINA
-j, --section=NOMBRE        Solo muestra la informacion de
la seccion NOMBRE
-M, --disassembler-options=OPC Pasa el texto OPC al desensamblador
-EB --endian=big            Asume el formato big endian al desensamblar
-EL --endian=little         Asume el formato little endian al desensamblar
--file-start-context    Incluye el contexto del inicio del fichero
(con -S)
-I, --include=DIR           Agrega el DIRectorio a la lista de busqueda de
ficheros fuente
-l, --line-numbers          Incluye los numeros de linea y los nombres
de fichero en la salida
-F, --file-offsets          Incluye desplazamientos de fichero al mostrar
la informacion
-C, --demangle[=ESTILO]     Decodifica los nombres de simbolo
obtenidos/procesados
El ESTILO, si se especifica, puede ser
'auto', 'gnu', 'lucid', 'arm', 'hp', 'edg',
'gnu-v3', 'java' o 'gnat'
-w, --wide                  Da formato a la salida para mas de 80 columnas
-z, --disassemble-zeroes    No salta los bloques de ceros al desensamblar
--start-address=DIR     Solo procesa los datos cuya direccion es
>= DIR
--stop-address=DIR      Solo procesa los datos cuya direccion es
<= DIR
--prefix-addresses      Muestra las direcciones completas a lo largo
del desensamblado
--[no-]show-raw-insn    Muestra en hexadecimal a lo largo
del desensamblado simbolico
--insn-width=ANCHO      Muestra ANCHO bytes en una sola linea con -d
--adjust-vma=DESPL      Agrega el DESPLazamiento a todas las direcciones
mostradas de seccion
--special-syms          Incluye simbolos especiales en los volcados de
simbolos
--prefix=PREFIJO        Agrega el PREFIJO a las rutas absolutas con -S
--prefix-strip=NIVEL    Descarta los nombres de directorio iniciales
con -S
--dwarf-depth=N      Do not display DIEs at depth N or greater
--dwarf-start=N      Display DIEs starting with N, at the same depth
or deeper
--dwarf-check        Make additional dwarf internal consistency checks.
```

Información sacada de [4]. Así tenemos la opción de usar :

-a para mostrar información de headers.
-p para mostrar el contenido de los ficheros headers.
-f para mostrar el contenido de los ficheeros headers total.
-h para mostrar el contenido de las secciones headers.

-t para mostrar el contenido de la tabla de símbolos.
-R para mostrar las entradas de reubicación dinámica en el fichero.
-x para mostrar el contenido de todos los headers.
Con todo esto anterior mostraríamos lo mismo que usando readelf -a.

Apartado B:

Podemos usar la opción -d que nos muestra el contenido del ensamblador.[?]

```
[root@ivancito P2S1]# objdump -d holamundoCPP

holamundoCPP:   formato del fichero elf64-x86-64


Desensamblado de la seccion .init:

0000000000400620 <_init>:
400620:   48 83 ec 08           sub    $0x8,%rsp
400624:   48 8b 05 c5 09 20 00  mov    0x2009c5(%rip),%rax     # 600ff0
    <__gmon_start__>
40062b:   48 85 c0              test   %rax,%rax
40062e:   74 02                 je     400632 <_init+0x12>
400630:   ff d0                 callq  *%rax
400632:   48 83 c4 08           add    $0x8,%rsp
400636:   c3                    retq

Desensamblado de la seccion .plt:

0000000000400640 <.plt>:
400640:   ff 35 c2 09 20 00     pushq  0x2009c2(%rip)      # 601008
    <_GLOBAL_OFFSET_TABLE_+0x8>
400646:   ff 25 c4 09 20 00     jmpq   *0x2009c4(%rip)      # 601010
    <_GLOBAL_OFFSET_TABLE_+0x10>
40064c:   0f 1f 40 00           nopl   0x0(%rax)

0000000000400650 <_ZNSt8ios_base4InitC1Ev@plt>:
400650:   ff 25 c2 09 20 00     jmpq   *0x2009c2(%rip)      # 601018
    <_ZNSt8ios_base4InitC1Ev@GLIBCXX_3.4>
400656:   68 00 00 00 00        pushq  $0x0
40065b:   e9 e0 ff ff ff        jmpq   400640 <.plt>

0000000000400660 <__cxa_atexit@plt>:
400660:   ff 25 ba 09 20 00     jmpq   *0x2009ba(%rip)      # 601020
    <__cxa_atexit@GLIBC_2.2.5>
400666:   68 01 00 00 00        pushq  $0x1
40066b:   e9 d0 ff ff ff        jmpq   400640 <.plt>

0000000000400670 <_ZNSt8ios_base4InitD1Ev@plt>:
400670:   ff 25 b2 09 20 00     jmpq   *0x2009b2(%rip)      # 601028
    <_ZNSt8ios_base4InitD1Ev@GLIBCXX_3.4>
400676:   68 02 00 00 00        pushq  $0x2
40067b:   e9 c0 ff ff ff        jmpq   400640 <.plt>

0000000000400680 <_ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@plt>:
```

```
400680:   ff 25 aa 09 20 00      jmpq  *0x2009aa(%rip)      # 601030
    <_ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@GLIBCXX_3.4>
400686:   68 03 00 00 00         pushq $0x3
40068b:   e9 b0 ff ff ff         jmpq  400640 <.plt>

0000000000400690 <_ZNSolsEPFRSoS_E@plt>:
400690:   ff 25 a2 09 20 00      jmpq  *0x2009a2(%rip)      # 601038
    <_ZNSolsEPFRSoS_E@GLIBCXX_3.4>
400696:   68 04 00 00 00         pushq $0x4
40069b:   e9 a0 ff ff ff         jmpq  400640 <.plt>

00000000004006a0
    <_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_@plt>:
4006a0:   ff 25 9a 09 20 00      jmpq  *0x20099a(%rip)      # 601040
    <_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_@GLIBCXX_3.4>
4006a6:   68 05 00 00 00         pushq $0x5
4006ab:   e9 90 ff ff ff         jmpq  400640 <.plt>

Desensamblado de la seccion .text:

00000000004006b0 <_start>:
4006b0:   31 ed                  xor   %ebp,%ebp
4006b2:   49 89 d1               mov   %rdx,%r9
4006b5:   5e                     pop   %rsi
4006b6:   48 89 e2               mov   %rsp,%rdx
4006b9:   48 83 e4 f0            and   $0xfffffffffffffff0,%rsp
4006bd:   50                     push  %rax
4006be:   54                     push  %rsp
4006bf:   49 c7 c0 80 08 40 00   mov   $0x400880,%r8
4006c6:   48 c7 c1 10 08 40 00   mov   $0x400810,%rcx
4006cd:   48 c7 c7 87 07 40 00   mov   $0x400787,%rdi
4006d4:   ff 15 1e 09 20 00      callq *0x20091e(%rip)      # 600ff8
    <__libc_start_main@GLIBC_2.2.5>
4006da:   f4                     hlt
4006db:   0f 1f 44 00 00         nopl  0x0(%rax,%rax,1)

00000000004006e0 <deregister_tm_clones>:
4006e0:   55                     push  %rbp
4006e1:   b8 50 10 60 00         mov   $0x601050,%eax
4006e6:   48 3d 50 10 60 00      cmp   $0x601050,%rax
4006ec:   48 89 e5               mov   %rsp,%rbp
4006ef:   74 17                  je    400708 <deregister_tm_clones+0x28>
4006f1:   b8 00 00 00 00         mov   $0x0,%eax
4006f6:   48 85 c0               test  %rax,%rax
4006f9:   74 0d                  je    400708 <deregister_tm_clones+0x28>
4006fb:   5d                     pop   %rbp
4006fc:   bf 50 10 60 00         mov   $0x601050,%edi
400701:   ff e0                  jmpq  *%rax
400703:   0f 1f 44 00 00         nopl  0x0(%rax,%rax,1)
```

```
400708:   5d                    pop    %rbp
400709:   c3                    retq
40070a:   66 0f 1f 44 00 00     nopw   0x0(%rax,%rax,1)

0000000000400710 <register_tm_clones>:
400710:   be 50 10 60 00        mov    $0x601050,%esi
400715:   55                    push   %rbp
400716:   48 81 ee 50 10 60 00  sub    $0x601050,%rsi
40071d:   48 89 e5              mov    %rsp,%rbp
400720:   48 c1 fe 03           sar    $0x3,%rsi
400724:   48 89 f0              mov    %rsi,%rax
400727:   48 c1 e8 3f           shr    $0x3f,%rax
40072b:   48 01 c6              add    %rax,%rsi
40072e:   48 d1 fe              sar    %rsi
400731:   74 15                 je     400748 <register_tm_clones+0x38>
400733:   b8 00 00 00 00        mov    $0x0,%eax
400738:   48 85 c0              test   %rax,%rax
40073b:   74 0b                 je     400748 <register_tm_clones+0x38>
40073d:   5d                    pop    %rbp
40073e:   bf 50 10 60 00        mov    $0x601050,%edi
400743:   ff e0                 jmpq   *%rax
400745:   0f 1f 00              nopl   (%rax)
400748:   5d                    pop    %rbp
400749:   c3                    retq
40074a:   66 0f 1f 44 00 00     nopw   0x0(%rax,%rax,1)

0000000000400750 <__do_global_dtors_aux>:
400750:   80 3d 19 0a 20 00 00  cmpb   $0x0,0x200a19(%rip)     # 601170
    <completed.6991>
400757:   75 17                 jne    400770 <__do_global_dtors_aux+0x20>
400759:   55                    push   %rbp
40075a:   48 89 e5              mov    %rsp,%rbp
40075d:   e8 7e ff ff ff        callq  4006e0 <deregister_tm_clones>
400762:   c6 05 07 0a 20 00 01  movb   $0x1,0x200a07(%rip)     # 601170
    <completed.6991>
400769:   5d                    pop    %rbp
40076a:   c3                    retq
40076b:   0f 1f 44 00 00        nopl   0x0(%rax,%rax,1)
400770:   f3 c3                 repz retq
400772:   0f 1f 40 00           nopl   0x0(%rax)
400776:   66 2e 0f 1f 84 00 00  nopw   %cs:0x0(%rax,%rax,1)
40077d:   00 00 00

0000000000400780 <frame_dummy>:
400780:   55                    push   %rbp
400781:   48 89 e5              mov    %rsp,%rbp
400784:   5d                    pop    %rbp
400785:   eb 89                 jmp    400710 <register_tm_clones>
```

```
0000000000400787 <main>:
400787:   55                      push   %rbp
400788:   48 89 e5                mov    %rsp,%rbp
40078b:   be a1 08 40 00          mov    $0x4008a1,%esi
400790:   bf 60 10 60 00          mov    $0x601060,%edi
400795:   e8 e6 fe ff ff          callq  400680
    <_ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@plt>
40079a:   be a0 06 40 00          mov    $0x4006a0,%esi
40079f:   48 89 c7                mov    %rax,%rdi
4007a2:   e8 e9 fe ff ff          callq  400690 <_ZNSolsEPFRSoS_E@plt>
4007a7:   b8 00 00 00 00          mov    $0x0,%eax
4007ac:   5d                      pop    %rbp
4007ad:   c3                      retq

00000000004007ae <_Z41__static_initialization_and_destruction_0ii>:
4007ae:   55                      push   %rbp
4007af:   48 89 e5                mov    %rsp,%rbp
4007b2:   48 83 ec 10             sub    $0x10,%rsp
4007b6:   89 7d fc                mov    %edi,-0x4(%rbp)
4007b9:   89 75 f8                mov    %esi,-0x8(%rbp)
4007bc:   83 7d fc 01             cmpl   $0x1,-0x4(%rbp)
4007c0:   75 27                   jne    4007e9
    <_Z41__static_initialization_and_destruction_0ii+0x3b>
4007c2:   81 7d f8 ff ff 00 00    cmpl   $0xffff,-0x8(%rbp)
4007c9:   75 1e                   jne    4007e9
    <_Z41__static_initialization_and_destruction_0ii+0x3b>
4007cb:   bf 71 11 60 00          mov    $0x601171,%edi
4007d0:   e8 7b fe ff ff          callq  400650 <_ZNSt8ios_base4InitC1Ev@plt>
4007d5:   ba 98 08 40 00          mov    $0x400898,%edx
4007da:   be 71 11 60 00          mov    $0x601171,%esi
4007df:   bf 70 06 40 00          mov    $0x400670,%edi
4007e4:   e8 77 fe ff ff          callq  400660 <__cxa_atexit@plt>
4007e9:   90                      nop
4007ea:   c9                      leaveq
4007eb:   c3                      retq

00000000004007ec <_GLOBAL__sub_I_main>:
4007ec:   55                      push   %rbp
4007ed:   48 89 e5                mov    %rsp,%rbp
4007f0:   be ff ff 00 00          mov    $0xffff,%esi
4007f5:   bf 01 00 00 00          mov    $0x1,%edi
4007fa:   e8 af ff ff ff          callq  4007ae
    <_Z41__static_initialization_and_destruction_0ii>
4007ff:   5d                      pop    %rbp
400800:   c3                      retq
400801:   66 2e 0f 1f 84 00 00    nopw   %cs:0x0(%rax,%rax,1)
400808:   00 00 00
40080b:   0f 1f 44 00 00          nopl   0x0(%rax,%rax,1)
```

```
0000000000400810 <__libc_csu_init>:
400810:   41 57                 push   %r15
400812:   41 56                 push   %r14
400814:   49 89 d7              mov    %rdx,%r15
400817:   41 55                 push   %r13
400819:   41 54                 push   %r12
40081b:   4c 8d 25 b6 05 20 00  lea    0x2005b6(%rip),%r12     # 600dd8
    <__frame_dummy_init_array_entry>
400822:   55                    push   %rbp
400823:   48 8d 2d be 05 20 00  lea    0x2005be(%rip),%rbp     # 600de8
    <__init_array_end>
40082a:   53                    push   %rbx
40082b:   41 89 fd              mov    %edi,%r13d
40082e:   49 89 f6              mov    %rsi,%r14
400831:   4c 29 e5              sub    %r12,%rbp
400834:   48 83 ec 08           sub    $0x8,%rsp
400838:   48 c1 fd 03           sar    $0x3,%rbp
40083c:   e8 df fd ff ff        callq  400620 <_init>
400841:   48 85 ed              test   %rbp,%rbp
400844:   74 20                 je     400866 <__libc_csu_init+0x56>
400846:   31 db                 xor    %ebx,%ebx
400848:   0f 1f 84 00 00 00 00  nopl   0x0(%rax,%rax,1)
40084f:   00
400850:   4c 89 fa              mov    %r15,%rdx
400853:   4c 89 f6              mov    %r14,%rsi
400856:   44 89 ef              mov    %r13d,%edi
400859:   41 ff 14 dc           callq  *(%r12,%rbx,8)
40085d:   48 83 c3 01           add    $0x1,%rbx
400861:   48 39 dd              cmp    %rbx,%rbp
400864:   75 ea                 jne    400850 <__libc_csu_init+0x40>
400866:   48 83 c4 08           add    $0x8,%rsp
40086a:   5b                    pop    %rbx
40086b:   5d                    pop    %rbp
40086c:   41 5c                 pop    %r12
40086e:   41 5d                 pop    %r13
400870:   41 5e                 pop    %r14
400872:   41 5f                 pop    %r15
400874:   c3                    retq
400875:   90                    nop
400876:   66 2e 0f 1f 84 00 00  nopw   %cs:0x0(%rax,%rax,1)
40087d:   00 00 00

0000000000400880 <__libc_csu_fini>:
400880:   f3 c3                 repz retq

Desensamblado de la seccion .fini:

0000000000400884 <_fini>:
400884:   48 83 ec 08           sub    $0x8,%rsp
```

```
400888:   48 83 c4 08           add    $0x8,%rsp
40088c:   c3                    retq
```

3. Modifica el programa realizado en el ejercicio anterior para que el programa se detenga durante un rato, por ejemplo con un sleep(), al objeto de que podamos visualizar el archivo maps de su ejecución. Ahora analiza la información de su ELF para entrever cómo se ha construido dicho proceso a través de la información del ELF, por ejemplo, las direcciones y permisos de las regiones de texto y datos, etc.

Código del programa en C:[5]

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

main()
{
   printf("Hola mundo\n");
   sleep(1500);
}
```

Para obtener su PID podemos usar por ejemplo la orden ps con la opción -ax:

```
[root@ivancito ivancito]# ps -ax | grep holamundoC
4991 pts/0   S+     0:00 ./holamundoC
4993 pts/1   S+     0:00 grep --color=auto holamundoC
```

Después de obtener su PID mostramos el contenido:

```
[root@ivancito ivancito]# cat /proc/4991/maps
00400000-00401000 r-xp 00000000 fd:02 792123
    /home/ivancito/Documentos/SSo/P2S1/holamundoC
00600000-00601000 r--p 00000000 fd:02 792123
    /home/ivancito/Documentos/SSo/P2S1/holamundoC
00601000-00602000 rw-p 00001000 fd:02 792123
    /home/ivancito/Documentos/SSo/P2S1/holamundoC
02202000-02223000 rw-p 00000000 00:00 0                                  [heap]
7f3e2fc52000-7f3e2fe1d000 r-xp 00000000 fd:00 2498956
    /usr/lib64/libc-2.25.so
7f3e2fe1d000-7f3e3001d000 ---p 001cb000 fd:00 2498956
    /usr/lib64/libc-2.25.so
7f3e3001d000-7f3e30021000 r--p 001cb000 fd:00 2498956
    /usr/lib64/libc-2.25.so
7f3e30021000-7f3e30023000 rw-p 001cf000 fd:00 2498956
    /usr/lib64/libc-2.25.so
```

```
7f3e30023000-7f3e30027000 rw-p 00000000 00:00 0
7f3e30027000-7f3e3004e000 r-xp 00000000 fd:00 2502120
    /usr/lib64/ld-2.25.so
7f3e30231000-7f3e30234000 rw-p 00000000 00:00 0
7f3e3024b000-7f3e3024d000 rw-p 00000000 00:00 0
7f3e3024d000-7f3e3024e000 r--p 00026000 fd:00 2502120
    /usr/lib64/ld-2.25.so
7f3e3024e000-7f3e30250000 rw-p 00027000 fd:00 2502120
    /usr/lib64/ld-2.25.so
7ffe4dd26000-7ffe4dd47000 rw-p 00000000 00:00 0                  [stack]
7ffe4dd58000-7ffe4dd5b000 r--p 00000000 00:00 0                  [vvar]
7ffe4dd5b000-7ffe4dd5d000 r-xp 00000000 00:00 0                  [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0          [vsyscall]
```

Por último realizamos las siguientes comparaciones:
Vemos como desde la línea 1 hasta la línea 3 de este archivo representan las líneas de readelf –sections desde la primera hasta .init_array, que son las direcciones(Primera columna del fichero /proc/4991/maps). También podemos ver aspectos de este archivo como los permisos que corresponden a la segunda columna. Por ejemplo rw-p que viene a ser lectura, escritura y canal. También tenemos r-xp que es lectura, escritura y canal. Por otro lado está r–p que sería lectura y canal.

Referencias

[1] https://refspecs.linuxfoundation.org/LSB_1.2.0/gLSB/specialsections.html, consultado el 1 de Diciembre de 2017.

[2] http://refspecs.linuxfoundation.org/LSB_3.1.0/LSB-Core-PPC32/LSB-Core-PPC32/sections.html, consultado el 1 de Diciembre de 2017.

[3] https://systemoverlord.com/2017/03/19/got-and-plt-for-pwning.html, consultado el 1 de Diciembre de 2017.

[4] https://sourceware.org/binutils/docs/binutils/objdump.html, consultado el 1 de Diciembre de 2017.

[5] https://linux.die.net/man/3/sleep, consultado el 1 de Diciembre de 2017.