nope

Name: Nope
Category: Cryptography
Description: *Chuck Testa loves Stego!*

Enumeration

Relevant information obtain by enumeration on the target file:
→ **Type of File**: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, not stripped
→ **Execution**:

⇒ **Input**: None

⇒ **Output**: *"La clave no es flag{ASM} }:)"*

→ **Contents**:

⇒ 2 Relevant strings:
- *La clave no es flag{ASM} }:)*
- *La flag no es flag{asm} }:)*

⇒ The previous strings are barely different, but this difference may be a hint to find the flag. ¿What is ASM/asm?
- ASM seems to fit for assembly code. I find no obvious relation between ASM and asm but the ascii difference between caps.
- The strings command shows *"the.asm"* which we can assume to be an assembly file, maybe hidden in the elf.

⇒ While running the strings command we notice a series of strings "flagX" with X in [1,25] and a string "the_flag", we suppose these are variable names whose content we could see in gdb.

⇒ Also, from the 2 relevant strings found earlier, only one is visible by the command strings. This is, the strings with "asm" instead of "ASM" is not declared as an explicit string, or hidden somehow. "La flag no es flag{asm} }:)" has become a potential hint to find the flag.
→ **GDB Debugging**: Check the "Enumeration/gdb" file for a detailed report of the GDB analysis.

⇒ Our assumptions about the strings command output were true, each flagX variable contains a character from the "La flag no es flag{asm} }".

⇒ Nevertheless, the_flag contains the character '\n', but is also the argument of the asm instruction **callq**, meaning the_flag is also a procedure. This information is confusing.

⇒ There are no signs of a variable containing the string "La clave no es flag{ASM} }:)", so we assume its hard-coded in the program.
→ **ELFTOC**: Tool that creates a C struct that replicates the memory state of an elf. The file "Enumeration/nope.c" has further details.

⇒ Most of the data seems invaluable for the problem but in the data section we can find two strings:
- "La clave no es flag{ASM} }:)\n" → Normal hard-coded string.
- "\0L\0a\0 \0f\0l\0a\0g\0 \0n\0o\0 \0e\0s\0 \0f\0l\0a\0g\0{\0a\0s\0m\0}\0 \0}:)" → "La flag no es flag{asm} }:)" hidden with '\0' characters.

⇒ In the symbol table section, we discovered:
- the.asm actually is an assembly file we should try to open.
- All the flagX vars are defined as data while the_flag is defined as text, both defined in ReadOnly data.

→ **HEXEDIT**: We obtained the same reuslts as ELFTOC.

First-Contact

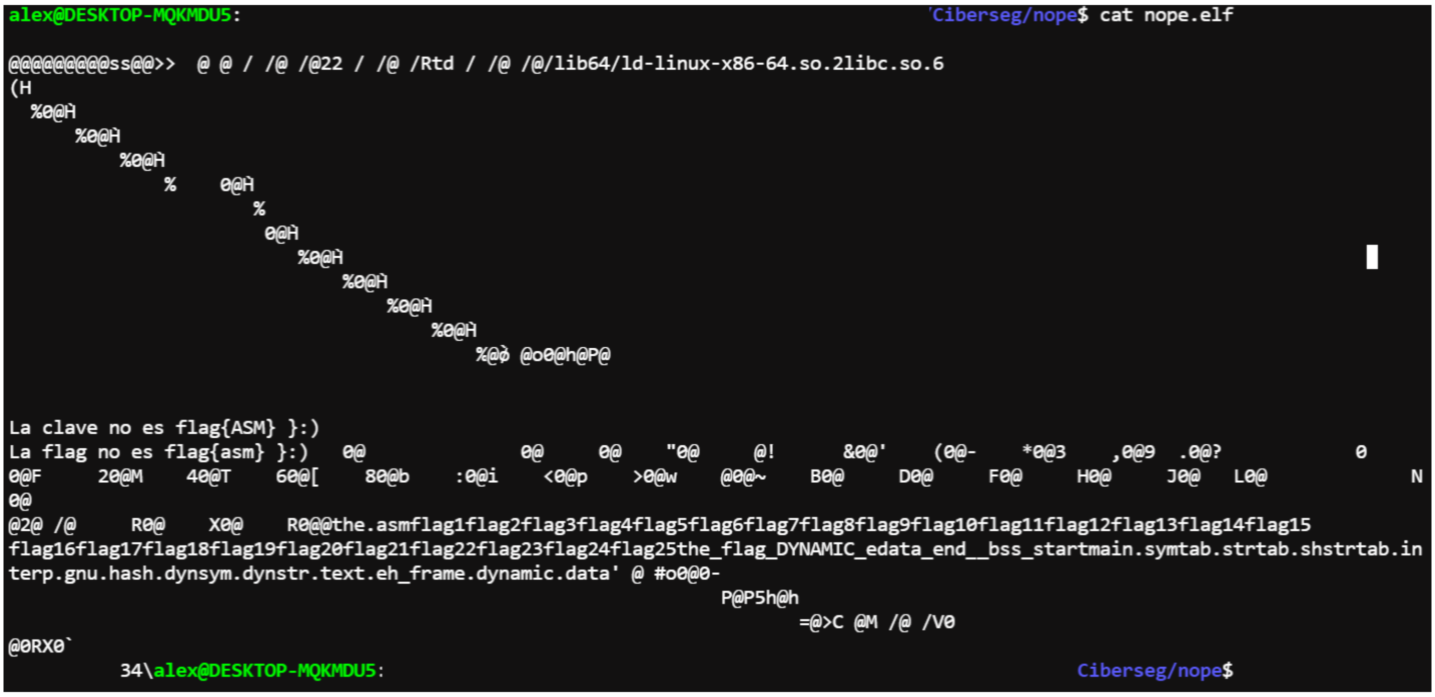
Type of File: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, not stripped (*file* command)

Execution:

- **Input**: None
- **Output**: *"La clave no es flag{ASM} }:)"*

Contents:

- In order to have a better understanding of the file, let's have a look at its contents, starting with cat.
- *Cat* command output:

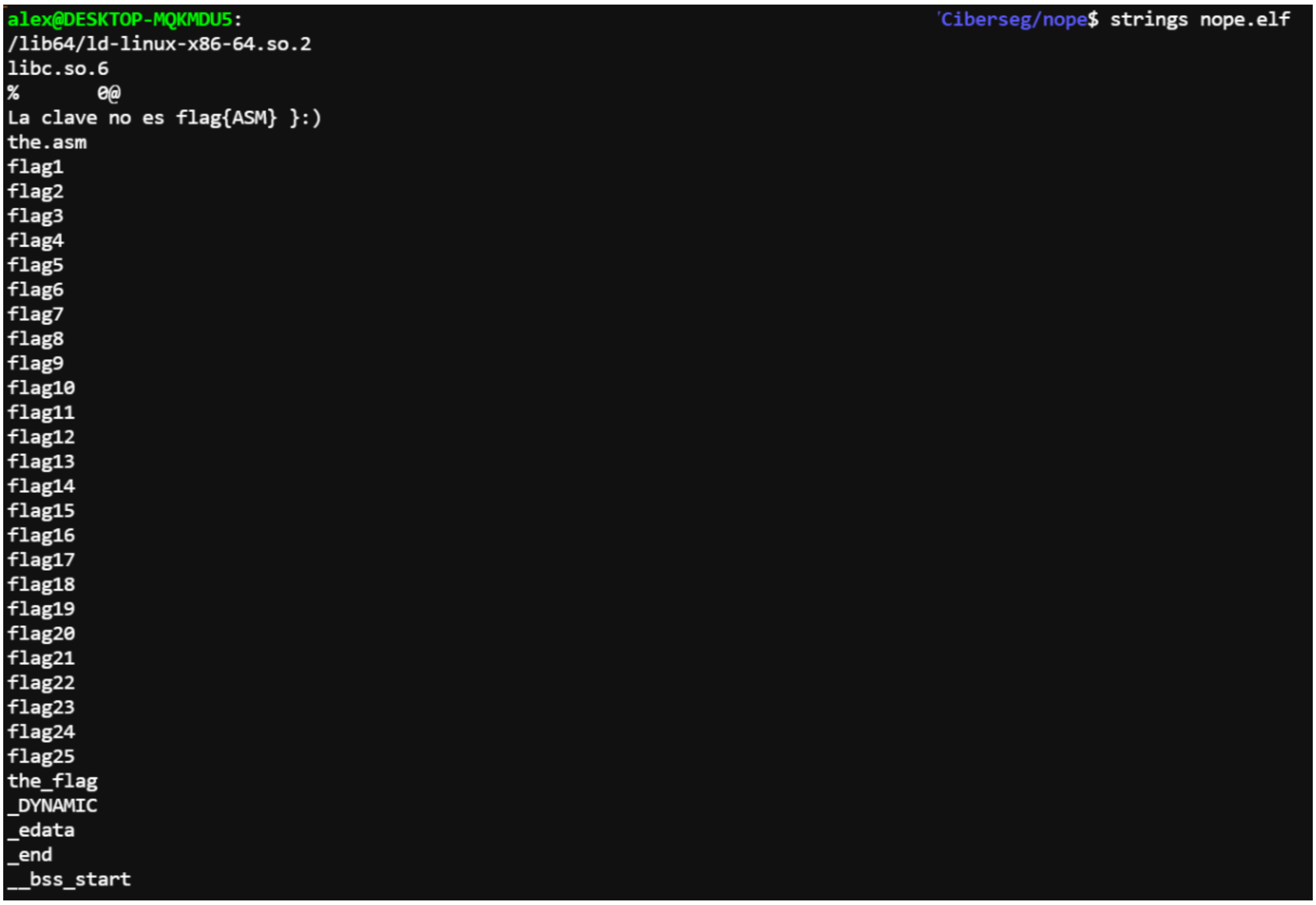


```
alex@DESKTOP-MQKMDU5:                                                   'Ciberseg/nope$ cat nope.elf

@@@@@@@@@@ss@@>>  @ @ / /@ /@22 / /@ /Rtd / /@ /@/lib64/ld-linux-x86-64.so.2libc.so.6
(H
  %0@H
      %0@H
            %0@H
                  %    0@H
                        %
                         0@H
                              %0@H
                                    %0@H
                                          %0@H
                                                %0@H
                                                      %@ǿ @o0@h@P@


La clave no es flag{ASM} }:)
La flag no es flag{asm} }:)   0@                    0@       0@      "0@       @!        &0@'      (0@-      *0@3     ,0@9  .0@?              0
0@F      20@M     40@T     60@[     80@b     :0@i     <0@p     >0@w     @0@~     B0@      D0@      F0@      H0@      J0@   L0@               N
0@
@2@ /@      R0@     X0@     R0@@the.asmflag1flag2flag3flag4flag5flag6flag7flag8flag9flag10flag11flag12flag13flag14flag15
flag16flag17flag18flag19flag20flag21flag22flag23flag24flag25the_flag_DYNAMIC_edata_end__bss_startmain.symtab.strtab.shstrtab.in
terp.gnu.hash.dynsym.dynstr.text.eh_frame.dynamic.data' @ #o0@0-
                                                                 P@P5h@h
                                                                          =@>C @M /@ /V0
@0RX0`
          34\alex@DESKTOP-MQKMDU5:                                                          Ciberseg/nope$
```

- 2 Relevant strings:
  - *La clave no es flag{ASM} }:)*
  - *La flag no es flag{asm} }:)*
- The previous strings are barely different, but this difference may be a hint to find the flag. ¿What is ASM/asm?
  - ASM seems to fit for assembly code. I find no obvious relation between ASM and asm but the ascii difference between caps.
  - The strings command shows *"the.asm"* which we can assume to be an assembly file, maybe hidden in the elf.
- The file seems to have explicit strings in it, lets have a better look with the *strings* command:



```
alex@DESKTOP-MQKMDU5:                                      'Ciberseg/nope$ strings nope.elf
/lib64/ld-linux-x86-64.so.2
libc.so.6
%       0@
La clave no es flag{ASM} }:)
the.asm
flag1
flag2
flag3
flag4
flag5
flag6
flag7
flag8
flag9
flag10
flag11
flag12
flag13
flag14
flag15
flag16
flag17
flag18
flag19
flag20
flag21
flag22
flag23
flag24
flag25
the_flag
_DYNAMIC
_edata
_end
__bss_start
```

- Although the output continues, the relevant data is shown above. First, we notice a series of strings "flagX" with X in [1,25] and a string "the_flag", we suppose these are variable names whose content we could see in gdb later.

- Also, notice that from the 2 strings we recovered in the cat command, only one is visible by the command strings. This is, the strings with "asm" instead of "ASM" is not declared as an explicit string, or hidden somehow. "La flag no es flag{asm} }:)" has become a potential hint to find the flag.

gdb

```
# GDB execution for the file nope.elf
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from nope.elf...
(No debugging symbols found in nope.elf)
(gdb)
(gdb) b main # Breakpoint at main function
Breakpoint 1 at 0x401000
(gdb) run # Start execution
Starting program: /mnt/c/Users/Alex/OneDrive - Universidad de Alcala/
Documentos/Ciberseg/nope/nope.elf

Breakpoint 1, 0x0000000000401000 in main () # Main function starts at 0x0401000
(gdb)disassemble main # Obtain assembly code equivalent to main
Dump of assembler code for function main:
   0x0000000000401000 <+0>:     callq  0x40100a <the_flag> # Call to a
procedure named "the_flag" with no args
   0x0000000000401005 <+5>:     jmpq   0x401132 <end>
End of assembler dump.
(gdb)disassemble 0x40100a # Disassemble the_flag method
Dump of assembler code for function the_flag:
   0x000000000040100a <+0>:     nop
   0x000000000040100b <+1>:     mov    $0x4,%eax
   0x0000000000401010 <+6>:     mov    $0x1,%ebx
   0x0000000000401015 <+11>:    nop
   0x0000000000401016 <+12>:    nop
   0x0000000000401017 <+13>:    nop
   0x0000000000401018 <+14>:    lea    0x403000,%rcx
   0x0000000000401020 <+22>:    nop
   0x0000000000401021 <+23>:    nop
   0x0000000000401022 <+24>:    mov    $0x3,%edx
   0x0000000000401027 <+29>:    int    $0x80
   0x0000000000401029 <+31>:    nop
   0x000000000040102a <+32>:    mov    $0x4,%eax
   0x000000000040102f <+37>:    nop
   0x0000000000401030 <+38>:    nop
   0x0000000000401031 <+39>:    mov    $0x1,%ebx
```

```
0x0000000000401036 <+44>:    nop
0x0000000000401037 <+45>:    lea    0x403003,%rcx
0x000000000040103f <+53>:    mov    $0x3,%edx
0x0000000000401044 <+58>:    nop
0x0000000000401045 <+59>:    int    $0x80
0x0000000000401047 <+61>:    mov    $0x4,%eax
0x000000000040104c <+66>:    mov    $0x1,%ebx
0x0000000000401051 <+71>:    nop
0x0000000000401052 <+72>:    nop
0x0000000000401053 <+73>:    lea    0x403006,%rcx
0x000000000040105b <+81>:    mov    $0x3,%edx
0x0000000000401060 <+86>:    nop
0x0000000000401061 <+87>:    int    $0x80
0x0000000000401063 <+89>:    nop
0x0000000000401064 <+90>:    nop
0x0000000000401065 <+91>:    mov    $0x4,%eax
0x000000000040106a <+96>:    nop
0x000000000040106b <+97>:    mov    $0x1,%ebx
0x0000000000401070 <+102>:   lea    0x403009,%rcx
0x0000000000401078 <+110>:   nop
0x0000000000401079 <+111>:   mov    $0x3,%edx
0x000000000040107e <+116>:   int    $0x80
0x0000000000401080 <+118>:   mov    $0x4,%eax
0x0000000000401085 <+123>:   nop
0x0000000000401086 <+124>:   nop
0x0000000000401087 <+125>:   mov    $0x1,%ebx
0x000000000040108c <+130>:   lea    0x40300c,%rcx
0x0000000000401094 <+138>:   mov    $0x3,%edx
0x0000000000401099 <+143>:   nop
0x000000000040109a <+144>:   int    $0x80
0x000000000040109c <+146>:   nop
0x000000000040109d <+147>:   nop
0x000000000040109e <+148>:   nop
0x000000000040109f <+149>:   mov    $0x4,%eax
0x00000000004010a4 <+154>:   mov    $0x1,%ebx
0x00000000004010a9 <+159>:   nop
0x00000000004010aa <+160>:   lea    0x40300f,%rcx
0x00000000004010b2 <+168>:   nop
0x00000000004010b3 <+169>:   nop
0x00000000004010b4 <+170>:   nop
0x00000000004010b5 <+171>:   nop
0x00000000004010b6 <+172>:   mov    $0x3,%edx
0x00000000004010bb <+177>:   int    $0x80
0x00000000004010bd <+179>:   nop
0x00000000004010be <+180>:   nop
0x00000000004010bf <+181>:   mov    $0x4,%eax
0x00000000004010c4 <+186>:   nop
0x00000000004010c5 <+187>:   mov    $0x1,%ebx
0x00000000004010ca <+192>:   nop
0x00000000004010cb <+193>:   lea    0x403012,%rcx
0x00000000004010d3 <+201>:   mov    $0x3,%edx
0x00000000004010d8 <+206>:   nop
0x00000000004010d9 <+207>:   int    $0x80
```

```
   0x00000000004010db <+209>:   mov    $0x4,%eax
   0x00000000004010e0 <+214>:   mov    $0x1,%ebx
   0x00000000004010e5 <+219>:   nop
   0x00000000004010e6 <+220>:   nop
   0x00000000004010e7 <+221>:   lea    0x403015,%rcx
   0x00000000004010ef <+229>:   mov    $0x3,%edx
   0x00000000004010f4 <+234>:   nop
   0x00000000004010f5 <+235>:   int    $0x80
   0x00000000004010f7 <+237>:   mov    $0x4,%eax
   0x00000000004010fc <+242>:   mov    $0x1,%ebx
   0x0000000000401101 <+247>:   lea    0x403018,%rcx
   0x0000000000401109 <+255>:   nop
   0x000000000040110a <+256>:   mov    $0x3,%edx
   0x000000000040110f <+261>:   int    $0x80
   0x0000000000401111 <+263>:   mov    $0x4,%eax
   0x0000000000401116 <+268>:   nop
   0x0000000000401117 <+269>:   nop
   0x0000000000401118 <+270>:   nop
   0x0000000000401119 <+271>:   nop
   0x000000000040111a <+272>:   nop
   0x000000000040111b <+273>:   mov    $0x1,%ebx
   0x0000000000401120 <+278>:   lea    0x40301b,%rcx
   0x0000000000401128 <+286>:   mov    $0x3,%edx
   0x000000000040112d <+291>:   nop
   0x000000000040112e <+292>:   nop
   0x000000000040112f <+293>:   int    $0x80
   0x0000000000401131 <+295>:   retq
End of assembler dump.
# Non-relevant information recovered, trying accessing to content of flagX
names.
(gdb)print flag1
'flag1' has unknown type; cast it to its declared type
(gdb)print (char) flag1
$6 = 76 'L' # flag1 is a variable which contains the char 'L'
print (char) flag2
$7 = 97 'a'
(gdb) print (char) flag3
$8 = 32 ' '
(gdb) print (char) flag4
$9 = 102 'f'
(gdb) print (char) flag5
$10 = 108 'l'
(gdb) print (char) flag6
$11 = 97 'a'
(gdb) print (char) flag7
$12 = 103 'g'
(gdb) print (char) flag8
$13 = 32 ' '
(gdb) print (char) flag9
$14 = 110 'n'
# print (char) flag{10-22}
(gdb) print (char) flag23
$15 = 125 '}'
```

```
(gdb) print (char) flag24
$16 = 32 ' '
(gdb) print (char) flag25
$17 = 125 '}'
(gdb) print (char) flag26 # Corresponding to strings output, there are only 25
flag variables. Forming "La flag no es flag{asm} }"
No symbol table is loaded.  Use the "file" command.
(gdb)print (char) the_flag # The flag is defined as the character '\n' but also
called as a procedure in main...
$18 = 10 '\n'
(gdb)print &the_flag # Obtain the memory address of the_flag
$19 = (<text variable, no debug info> *) 0x40100a <the_flag>
(gdb) print (char*) the_flag # Try to print the string in the_flag in case it
contains one
$20 = 0x40100a <the_flag> "\220\270\004" # Octal to ASCII: É¸EOT --> Not
relevant
(gdb)print &flag # Obtain the memory address of the start of the string
$23 = (<data variable, no debug info> *) 0x403000
(gdb) print (char*) 0x403000 # Try to print the complete string.
$24 = 0x403000 "La clave no es flag{ASM} }:)\n" # Success, we obtained the
first string
(gdb)
```

nope.c

```
// Using the program elftoc we generated a c code that creates data structures
to replicate the memory state during the execution
#include <stddef.h>
#include <elf.h>

#define ADDR_RODATA 0x00400000

enum sections
{
  SHN_INTERP = 1, SHN_HASH, SHN_GNU_HASH, SHN_DYNSYM, SHN_DYNSTR, SHN_TEXT,
  SHN_EH_FRAME, SHN_DYNAMIC, SHN_DATA, SHN_SYMTAB, SHN_STRTAB, SHN_SHSTRTAB,
  SHN_COUNT
};

typedef struct elf
{
  Elf64_Ehdr      ehdr;
  Elf64_Phdr      phdrs[8];
  unsigned char   interp[28];
  unsigned char   pad1[4];
  Elf64_Word      hash[4];
  Elf64_Word      gnu_hash[7];
  unsigned char   pad2[4];
  Elf64_Sym       dynsym[1];
  char            dynstr[11];
  unsigned char   pad3[3469];
  unsigned char   text[318];
  unsigned char   pad4[7650];
  Elf64_Dyn       dynamic[14];
  unsigned char   data[82];
  unsigned char   pad5[6];
  Elf64_Sym       symtab[36];
  char            strtab[222];
  char            shstrtab[92];
  unsigned char   pad6[6];
  Elf64_Shdr      shdrs[SHN_COUNT];
} elf;

elf foo =
{
  /* ehdr */
  {
    { 0x7F, 'E', 'L', 'F', ELFCLASS64, ELFDATA2LSB, EV_CURRENT, ELFOSABI_SYSV,
      0, 0, 0, 0, 0, 0, 0, 0 },
    ET_EXEC, EM_X86_64, EV_CURRENT, ADDR_RODATA + offsetof(elf, text),
    offsetof(elf, phdrs), offsetof(elf, shdrs), 0, sizeof(Elf64_Ehdr),
    sizeof(Elf64_Phdr), sizeof foo.phdrs / sizeof *foo.phdrs,
    sizeof(Elf64_Shdr), sizeof foo.shdrs / sizeof *foo.shdrs, SHN_SHSTRTAB
  },
  /* phdrs */
  {
    { PT_PHDR, PF_R, offsetof(elf, phdrs), ADDR_RODATA + offsetof(elf, phdrs),
      ADDR_RODATA + offsetof(elf, phdrs), sizeof foo.phdrs, sizeof foo.phdrs,
      sizeof(Elf64_Addr) },
    { PT_INTERP, PF_R, offsetof(elf, interp),
      ADDR_RODATA + offsetof(elf, interp),
      ADDR_RODATA + offsetof(elf, interp), sizeof foo.interp,
      sizeof foo.interp, 1 },
    { PT_LOAD, PF_R, 0, ADDR_RODATA, ADDR_RODATA, offsetof(elf, pad3),
      offsetof(elf, pad3), 0x1000 },
    { PT_LOAD, PF_R | PF_X, offsetof(elf, text),
      ADDR_RODATA + offsetof(elf, text), ADDR_RODATA + offsetof(elf, text),
      sizeof foo.text, sizeof foo.text, 0x1000 },
    { PT_LOAD, PF_R, offsetof(elf, pad4) + 0x0EC2,
      ADDR_RODATA + offsetof(elf, pad4) + 0x0EC2,
      ADDR_RODATA + offsetof(elf, pad4) + 0x0EC2, 0, 0, 0x1000 },
    { PT_LOAD, PF_R | PF_W, offsetof(elf, dynamic),
      ADDR_RODATA + offsetof(elf, dynamic),
      ADDR_RODATA + offsetof(elf, dynamic),
      offsetof(elf, pad5) - offsetof(elf, dynamic),
      offsetof(elf, pad5) - offsetof(elf, dynamic), 0x1000 },
    { PT_DYNAMIC, PF_R | PF_W, offsetof(elf, dynamic),
      ADDR_RODATA + offsetof(elf, dynamic),
      ADDR_RODATA + offsetof(elf, dynamic), sizeof foo.dynamic,
      sizeof foo.dynamic, sizeof(Elf64_Addr) },
    { PT_GNU_RELRO, PF_R, offsetof(elf, dynamic),
      ADDR_RODATA + offsetof(elf, dynamic),
      ADDR_RODATA + offsetof(elf, dynamic), sizeof foo.dynamic,
      sizeof foo.dynamic, 1 }
  },
  /* interp */
  "/lib64/ld-linux-x86-64.so.2",
  /* pad1 */
  { 0 },
  /* hash */
  {
    1, 1,
    0,
    0
  },
  /* gnu_hash */
  {
    1, 1, 1, 0,
    0x00000000, 0x00000000,
    0
  },
  /* pad2 */
  { 0 },
  /* dynsym */
  {
    { 0, 0, 0, SHN_UNDEF, 0, 0 }
  },
  /* dynstr */
  "\0libc.so.6",
  /* pad3 */
  { 0 },
  /* text */
  {
    0xE8, 0x05, 0x00, 0x00, 0x00, 0xE9, 0x28, 0x01, 0x00, 0x00, 0x90, 0xB8,
    0x04, 0x00, 0x00, 0x00, 0xBB, 0x01, 0x00, 0x00, 0x00, 0x90, 0x90, 0x90,
    0x48, 0x8D, 0x0C, 0x25, 0x00, 0x30, 0x40, 0x00, 0x90, 0x90, 0xBA, 0x03,
    0x00, 0x00, 0x00, 0xCD, 0x80, 0x90, 0xB8, 0x04, 0x00, 0x00, 0x00, 0x90,
    0x90, 0xBB, 0x01, 0x00, 0x00, 0x00, 0x90, 0x48, 0x8D, 0x0C, 0x25, 0x03,
    0x30, 0x40, 0x00, 0xBA, 0x03, 0x00, 0x00, 0x00, 0x90, 0xCD, 0x80, 0xB8,
    0x04, 0x00, 0x00, 0x00, 0xBB, 0x01, 0x00, 0x00, 0x00, 0x90, 0x90, 0x48,
    0x8D, 0x0C, 0x25, 0x06, 0x30, 0x40, 0x00, 0xBA, 0x03, 0x00, 0x00, 0x00,
    0x90, 0xCD, 0x80, 0x90, 0x90, 0xB8, 0x04, 0x00, 0x00, 0x00, 0x90, 0xBB,
    0x01, 0x00, 0x00, 0x00, 0x48, 0x8D, 0x0C, 0x25, 0x09, 0x30, 0x40, 0x00,
    0x90, 0xBA, 0x03, 0x00, 0x00, 0x00, 0xCD, 0x80, 0xB8, 0x04, 0x00, 0x00,
    0x00, 0x90, 0x90, 0xBB, 0x01, 0x00, 0x00, 0x00, 0x48, 0x8D, 0x0C, 0x25,
    0x0C, 0x30, 0x40, 0x00, 0xBA, 0x03, 0x00, 0x00, 0x00, 0x90, 0xCD, 0x80,
    0x90, 0x90, 0x90, 0xB8, 0x04, 0x00, 0x00, 0x00, 0xBB, 0x01, 0x00, 0x00,
    0x00, 0x90, 0x48, 0x8D, 0x0C, 0x25, 0x0F, 0x30, 0x40, 0x00, 0x90, 0x90,
    0x90, 0x90, 0xBA, 0x03, 0x00, 0x00, 0x00, 0xCD, 0x80, 0x90, 0x90, 0xB8,
    0x04, 0x00, 0x00, 0x00, 0x90, 0xBB, 0x01, 0x00, 0x00, 0x00, 0x90, 0x48,
    0x8D, 0x0C, 0x25, 0x12, 0x30, 0x40, 0x00, 0xBA, 0x03, 0x00, 0x00, 0x00,
    0x90, 0xCD, 0x80, 0xB8, 0x04, 0x00, 0x00, 0x00, 0xBB, 0x01, 0x00, 0x00,
    0x00, 0x90, 0x90, 0x48, 0x8D, 0x0C, 0x25, 0x15, 0x30, 0x40, 0x00, 0xBA,
    0x03, 0x00, 0x00, 0x00, 0x90, 0xCD, 0x80, 0xB8, 0x04, 0x00, 0x00, 0x00,
    0xBB, 0x01, 0x00, 0x00, 0x00, 0x48, 0x8D, 0x0C, 0x25, 0x18, 0x30, 0x40,
    0x00, 0x90, 0xBA, 0x03, 0x00, 0x00, 0x00, 0xCD, 0x80, 0xB8, 0x04, 0x00,
    0x00, 0x00, 0x90, 0x90, 0x90, 0x90, 0x90, 0xBB, 0x01, 0x00, 0x00, 0x00,
    0x48, 0x8D, 0x0C, 0x25, 0x1B, 0x30, 0x40, 0x00, 0xBA, 0x03, 0x00, 0x00,
    0x00, 0x90, 0x90, 0xCD, 0x80, 0xC3, 0xB8, 0x01, 0x00, 0x00, 0x00, 0xBB,
    0x00, 0x00, 0x00, 0x00, 0xCD, 0x80
  },
  /* pad4 */
  { 0 },
  /* dynamic */
  {
    { DT_NEEDED, { 1 } }, /* libc.so.6 */
    { DT_HASH, { ADDR_RODATA + offsetof(elf, hash) } },
    { DT_GNU_HASH, { ADDR_RODATA + offsetof(elf, gnu_hash) } },
    { DT_STRTAB, { ADDR_RODATA + offsetof(elf, dynstr) } },
    { DT_SYMTAB, { ADDR_RODATA + offsetof(elf, dynsym) } },
    { DT_STRSZ, { sizeof foo.dynstr } },
    { DT_ADDRNUM, { sizeof(Elf64_Sym) } },
    { DT_DEBUG, { 0 } },
    { DT_NULL, { 0 } },
    { DT_NULL, { 0 } },
    { DT_NULL, { 0 } },
    { DT_NULL, { 0 } },
    { DT_NULL, { 0 } },
    { DT_NULL, { 0 } }
  },
    /* data */
    "La clave no es flag{ASM} }:)\n" // Visible string while executing
    "\0L\0a\0 \0f\0l\0a\0g\0 \0n\0o\0 \0e\0s\0 \0f\0l\0a\0g\0{\0a\0s\0m\0}\0
  \0}:)", // String hidden with '\0' chars.
    /* pad5 */
    { 0 },
    /* symtab */
    {
      { 0, 0, 0, SHN_UNDEF, 0, 0 },
      /* the.asm */
      { 1, ELF64_ST_INFO(STB_LOCAL, STT_FILE), STV_DEFAULT, SHN_ABS, 0, 0 }, //
  the.asm is a file we should access
      /* flag */
      { 179, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data), 0 },
      /* flag1 */
      { 9, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data) + 0x1E, 0 },
      /* flag2 */
      { 15, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data) + 0x20, 0 },
      /* flag3 */
      { 21, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data) + 0x22, 0 },
      /* flag4 */
      { 27, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data) + 0x24, 0 },
      /* flag5 */
      { 33, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data) + 0x26, 0 },
      /* flag6 */
      { 39, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data) + 0x28, 0 },
      /* flag7 */
      { 45, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data) + 0x2A, 0 },
      /* flag8 */
      { 51, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data) + 0x2C, 0 },
      /* flag9 */
      { 57, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data) + 0x2E, 0 },
      /* flag10 */
      { 63, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data) + 0x30, 0 },
      /* flag11 */
      { 70, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data) + 0x32, 0 },
      /* flag12 */
      { 77, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, data) + 0x34, 0 },
      /* flag13 */
      { 84, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x36, 0 },
  /* flag14 */
  { 91, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x38, 0 },
  /* flag15 */
  { 98, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x3A, 0 },
  /* flag16 */
  { 105, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x3C, 0 },
  /* flag17 */
  { 112, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x3E, 0 },
  /* flag18 */
  { 119, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x40, 0 },
  /* flag19 */
  { 126, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x42, 0 },
  /* flag20 */
  { 133, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x44, 0 },
  /* flag21 */
  { 140, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x46, 0 },
  /* flag22 */
  { 147, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x48, 0 },
  /* flag23 */
  { 154, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x4A, 0 },
  /* flag24 */
  { 161, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x4C, 0 },
  /* flag25 */
  { 168, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, data) + 0x4E, 0 },
  /* the_flag */
  { 175, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_TEXT,
    ADDR_RODATA + offsetof(elf, text) + 10, 0 },
  /* end */
  { 201, ELF64_ST_INFO(STB_LOCAL, STT_NOTYPE), STV_DEFAULT, SHN_TEXT,
    ADDR_RODATA + offsetof(elf, text) + 0x0132, 0 },
  { 0, ELF64_ST_INFO(STB_LOCAL, STT_FILE), STV_DEFAULT, SHN_ABS, 0, 0 },
  /* _DYNAMIC */
  { 184, ELF64_ST_INFO(STB_LOCAL, STT_OBJECT), STV_DEFAULT, SHN_DYNAMIC,
    ADDR_RODATA + offsetof(elf, dynamic), 0 },
  /* _edata */
  { 193, ELF64_ST_INFO(STB_GLOBAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    ADDR_RODATA + offsetof(elf, pad5), 0 },
  /* _end */
  { 200, ELF64_ST_INFO(STB_GLOBAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
    0x403058, 0 },
      /* __bss_start */
      { 205, ELF64_ST_INFO(STB_GLOBAL, STT_NOTYPE), STV_DEFAULT, SHN_DATA,
        ADDR_RODATA + offsetof(elf, pad5), 0 },
      /* main */
      { 217, ELF64_ST_INFO(STB_GLOBAL, STT_NOTYPE), STV_DEFAULT, SHN_TEXT,
        ADDR_RODATA + offsetof(elf, text), 0 }
    },
    /* strtab */
    "\0the.asm\0flag1\0flag2\0flag3\0flag4\0flag5\0flag6\0flag7\0flag8\0flag9\0"
      "flag10\0flag11\0flag12\0flag13\0flag14\0flag15\0flag16\0flag17\0flag18\0"
      "flag19\0flag20\0flag21\0flag22\0flag23\0flag24\0flag25\0the_flag\0_DYNA"
      "MIC\0_edata\0_end\0__bss_start\0main",
    /* shstrtab */
    "\0.symtab\0.strtab\0.shstrtab\0.interp\0.gnu.hash\0.dynsym\0.dynstr\0.tex"
      "t\0.eh_frame\0.dynamic\0.data",
    /* pad6 */
    { 0 },
    /* shdrs */
    {
      { 0, SHT_NULL, 0, 0, 0, 0, SHN_UNDEF, 0, 0, 0 },
      /* .interp */
      { 27, SHT_PROGBITS, SHF_ALLOC, ADDR_RODATA + offsetof(elf, interp),
        offsetof(elf, interp), sizeof foo.interp, SHN_UNDEF, 0, 1, 0 },
      /* .hash */
      { 39, SHT_HASH, SHF_ALLOC, ADDR_RODATA + offsetof(elf, hash),
        offsetof(elf, hash), sizeof foo.hash, SHN_DYNSYM, 0, sizeof(Elf64_Addr),
        sizeof(Elf64_Word) },
      /* .gnu.hash */
      { 35, SHT_GNU_HASH, SHF_ALLOC, ADDR_RODATA + offsetof(elf, gnu_hash),
        offsetof(elf, gnu_hash), sizeof foo.gnu_hash, SHN_DYNSYM, 0,
        sizeof(Elf64_Addr), 0 },
      /* .dynsym */
      { 45, SHT_DYNSYM, SHF_ALLOC, ADDR_RODATA + offsetof(elf, dynsym),
        offsetof(elf, dynsym), sizeof foo.dynsym, SHN_DYNSTR, 1,
        sizeof(Elf64_Addr), sizeof(Elf64_Sym) },
      /* .dynstr */
      { 53, SHT_STRTAB, SHF_ALLOC, ADDR_RODATA + offsetof(elf, dynstr),
        offsetof(elf, dynstr), sizeof foo.dynstr, SHN_UNDEF, 0, 1, 0 },
      /* .text */
      { 61, SHT_PROGBITS, SHF_EXECINSTR | SHF_ALLOC,
        ADDR_RODATA + offsetof(elf, text), offsetof(elf, text), sizeof foo.text,
        SHN_UNDEF, 0, 0x10, 0 },
      /* .eh_frame */
      { 67, SHT_PROGBITS, SHF_ALLOC, ADDR_RODATA + offsetof(elf, pad4) + 0x0EC2,
        offsetof(elf, pad4) + 0x0EC2, 0, SHN_UNDEF, 0, sizeof(Elf64_Addr), 0 },
      /* .dynamic */
      { 77, SHT_DYNAMIC, SHF_WRITE | SHF_ALLOC,
        ADDR_RODATA + offsetof(elf, dynamic), offsetof(elf, dynamic),
        sizeof foo.dynamic, SHN_DYNSTR, 0, sizeof(Elf64_Addr),
        sizeof(Elf64_Dyn) },
      /* .data */
      { 86, SHT_PROGBITS, SHF_WRITE | SHF_ALLOC,
        ADDR_RODATA + offsetof(elf, data), offsetof(elf, data), sizeof foo.data,
        SHN_UNDEF, 0, 4, 0 },
      /* .symtab */
      { 1, SHT_SYMTAB, 0, 0, offsetof(elf, symtab), sizeof foo.symtab,
        SHN_STRTAB, 32, sizeof(Elf64_Addr), sizeof(Elf64_Sym) },
      /* .strtab */
      { 9, SHT_STRTAB, 0, 0, offsetof(elf, strtab), sizeof foo.strtab,
        SHN_UNDEF, 0, 1, 0 },
      /* .shstrtab */
      { 17, SHT_STRTAB, 0, 0, offsetof(elf, shstrtab), sizeof foo.shstrtab,
        SHN_UNDEF, 0, 1, 0 }
    }
  };
```

hexdump

```
0000000 457f 464c 0102 0001 0000 0000 0000 0000
0000010 0002 003e 0001 0000 1000 0040 0000 0000
0000020 0040 0000 0000 0000 34f8 0000 0000 0000
0000030 0000 0000 0040 0038 0008 0040 000d 000c
0000040 0006 0000 0004 0000 0040 0000 0000 0000
0000050 0040 0040 0000 0000 0040 0040 0000 0000
0000060 01c0 0000 0000 0000 01c0 0000 0000 0000
0000070 0008 0000 0000 0000 0003 0000 0004 0000
0000080 0200 0000 0000 0000 0200 0040 0000 0000
0000090 0200 0040 0000 0000 001c 0000 0000 0000
00000a0 001c 0000 0000 0000 0001 0000 0000 0000
00000b0 0001 0000 0004 0000 0000 0000 0000 0000
00000c0 0000 0040 0000 0000 0000 0040 0000 0000
00000d0 0273 0000 0000 0000 0273 0000 0000 0000
00000e0 1000 0000 0000 0000 0001 0000 0005 0000
00000f0 1000 0000 0000 0000 1000 0040 0000 0000
0000100 1000 0040 0000 0000 013e 0000 0000 0000
0000110 013e 0000 0000 0000 1000 0000 0000 0000
0000120 0001 0000 0004 0000 2000 0000 0000 0000
0000130 2000 0040 0000 0000 2000 0040 0000 0000
0000140 0000 0000 0000 0000 0000 0000 0000 0000
0000150 1000 0000 0000 0000 0001 0000 0006 0000
0000160 2f20 0000 0000 0000 2f20 0040 0000 0000
0000170 2f20 0040 0000 0000 0132 0000 0000 0000
0000180 0132 0000 0000 0000 1000 0000 0000 0000
0000190 0002 0000 0006 0000 2f20 0000 0000 0000
00001a0 2f20 0040 0000 0000 2f20 0040 0000 0000
00001b0 00e0 0000 0000 0000 00e0 0000 0000 0000
00001c0 0008 0000 0000 0000 e552 6474 0004 0000
00001d0 2f20 0000 0000 0000 2f20 0040 0000 0000
00001e0 2f20 0040 0000 0000 00e0 0000 0000 0000
00001f0 00e0 0000 0000 0000 0001 0000 0000 0000
0000200 6c2f 6269 3436 6c2f 2d64 696c 756e 2d78
0000210 3878 2d36 3436 732e 2e6f 0032 0000 0000
0000220 0001 0000 0001 0000 0000 0000 0000 0000
0000230 0001 0000 0001 0000 0001 0000 0000 0000
0000240 0000 0000 0000 0000 0000 0000 0000 0000
*
0000260 0000 0000 0000 0000 6c00 6269 2e63 6f73
0000270 362e 0000 0000 0000 0000 0000 0000 0000
0000280 0000 0000 0000 0000 0000 0000 0000 0000
*
0001000 05e8 0000 e900 0128 0000 b890 0004 0000
0001010 01bb 0000 9000 9090 8d48 250c 3000 0040
0001020 9090 03ba 0000 cd00 9080 04b8 0000 9000
0001030 bb90 0001 0000 4890 0c8d 0325 4030 ba00
0001040 0003 0000 cd90 b880 0004 0000 01bb 0000
0001050 9000 4890 0c8d 0625 4030 ba00 0003 0000
0001060 cd90 9080 b890 0004 0000 bb90 0001 0000
0001070 8d48 250c 3009 0040 ba90 0003 0000 80cd
```

```
0001080 04b8 0000 9000 bb90 0001 0000 8d48 250c
0001090 300c 0040 03ba 0000 9000 80cd 9090 b890
00010a0 0004 0000 01bb 0000 9000 8d48 250c 300f
00010b0 0040 9090 9090 03ba 0000 cd00 9080 b890
00010c0 0004 0000 bb90 0001 0000 4890 0c8d 1225
00010d0 4030 ba00 0003 0000 cd90 b880 0004 0000
00010e0 01bb 0000 9000 4890 0c8d 1525 4030 ba00
00010f0 0003 0000 cd90 b880 0004 0000 01bb 0000
0001100 4800 0c8d 1825 4030 9000 03ba 0000 cd00
0001110 b880 0004 0000 9090 9090 bb90 0001 0000
0001120 8d48 250c 301b 0040 03ba 0000 9000 cd90
0001130 c380 01b8 0000 bb00 0000 0000 80cd 0000
0001140 0000 0000 0000 0000 0000 0000 0000 0000
*
0002f20 0001 0000 0000 0000 0001 0000 0000 0000
0002f30 0004 0000 0000 0000 0220 0040 0000 0000
0002f40 fef5 6fff 0000 0000 0230 0040 0000 0000
0002f50 0005 0000 0000 0000 0268 0040 0000 0000
0002f60 0006 0000 0000 0000 0250 0040 0000 0000
0002f70 000a 0000 0000 0000 000b 0000 0000 0000
0002f80 000b 0000 0000 0000 0018 0000 0000 0000
0002f90 0015 0000 0000 0000 0000 0000 0000 0000
0002fa0 0000 0000 0000 0000 0000 0000 0000 0000
*
0003000 614c 6320 616c 6576 6e20 206f 7365 6620
0003010 616c 7b67 5341 7d4d 7d20 293a 000a 004c
0003020 0061 0020 0066 006c 0061 0067 0020 006e
0003030 006f 0020 0065 0073 0020 0066 006c 0061
0003040 0067 007b 0061 0073 006d 007d 0020 3a7d
0003050 0029 0000 0000 0000 0000 0000 0000 0000
0003060 0000 0000 0000 0000 0000 0000 0000 0000
0003070 0001 0000 0004 fff1 0000 0000 0000 0000
0003080 0000 0000 0000 0000 00b3 0000 0000 0009
0003090 3000 0040 0000 0000 0000 0000 0000 0000
00030a0 0009 0000 0000 0009 301e 0040 0000 0000
00030b0 0000 0000 0000 0000 000f 0000 0000 0009
00030c0 3020 0040 0000 0000 0000 0000 0000 0000
00030d0 0015 0000 0000 0009 3022 0040 0000 0000
00030e0 0000 0000 0000 0000 001b 0000 0000 0009
00030f0 3024 0040 0000 0000 0000 0000 0000 0000
0003100 0021 0000 0000 0009 3026 0040 0000 0000
0003110 0000 0000 0000 0000 0027 0000 0000 0009
0003120 3028 0040 0000 0000 0000 0000 0000 0000
0003130 002d 0000 0000 0009 302a 0040 0000 0000
0003140 0000 0000 0000 0000 0033 0000 0000 0009
0003150 302c 0040 0000 0000 0000 0000 0000 0000
0003160 0039 0000 0000 0009 302e 0040 0000 0000
0003170 0000 0000 0000 0000 003f 0000 0000 0009
0003180 3030 0040 0000 0000 0000 0000 0000 0000
0003190 0046 0000 0000 0009 3032 0040 0000 0000
00031a0 0000 0000 0000 0000 004d 0000 0000 0009
00031b0 3034 0040 0000 0000 0000 0000 0000 0000
00031c0 0054 0000 0000 0009 3036 0040 0000 0000
```

```
00031d0 0000 0000 0000 0000 005b 0000 0000 0009
00031e0 3038 0040 0000 0000 0000 0000 0000 0000
00031f0 0062 0000 0000 0009 303a 0040 0000 0000
0003200 0000 0000 0000 0000 0069 0000 0000 0009
0003210 303c 0040 0000 0000 0000 0000 0000 0000
0003220 0070 0000 0000 0009 303e 0040 0000 0000
0003230 0000 0000 0000 0000 0077 0000 0000 0009
0003240 3040 0040 0000 0000 0000 0000 0000 0000
0003250 007e 0000 0000 0009 3042 0040 0000 0000
0003260 0000 0000 0000 0000 0085 0000 0000 0009
0003270 3044 0040 0000 0000 0000 0000 0000 0000
0003280 008c 0000 0000 0009 3046 0040 0000 0000
0003290 0000 0000 0000 0000 0093 0000 0000 0009
00032a0 3048 0040 0000 0000 0000 0000 0000 0000
00032b0 009a 0000 0000 0009 304a 0040 0000 0000
00032c0 0000 0000 0000 0000 00a1 0000 0000 0009
00032d0 304c 0040 0000 0000 0000 0000 0000 0000
00032e0 00a8 0000 0000 0009 304e 0040 0000 0000
00032f0 0000 0000 0000 0000 00af 0000 0000 0006
0003300 100a 0040 0000 0000 0000 0000 0000 0000
0003310 00c9 0000 0000 0006 1132 0040 0000 0000
0003320 0000 0000 0000 0000 0000 0000 0004 fff1
0003330 0000 0000 0000 0000 0000 0000 0000 0000
0003340 00b8 0000 0001 0008 2f20 0040 0000 0000
0003350 0000 0000 0000 0000 00c1 0000 0010 0009
0003360 3052 0040 0000 0000 0000 0000 0000 0000
0003370 00c8 0000 0010 0009 3058 0040 0000 0000
0003380 0000 0000 0000 0000 00cd 0000 0010 0009
0003390 3052 0040 0000 0000 0000 0000 0000 0000
00033a0 00d9 0000 0010 0006 1000 0040 0000 0000
00033b0 0000 0000 0000 0000 7400 6568 612e 6d73
00033c0 6600 616c 3167 6600 616c 3267 6600 616c
00033d0 3367 6600 616c 3467 6600 616c 3567 6600
00033e0 616c 3667 6600 616c 3767 6600 616c 3867
00033f0 6600 616c 3967 6600 616c 3167 0030 6c66
0003400 6761 3131 6600 616c 3167 0032 6c66 6761
0003410 3331 6600 616c 3167 0034 6c66 6761 3531
0003420 6600 616c 3167 0036 6c66 6761 3731 6600
0003430 616c 3167 0038 6c66 6761 3931 6600 616c
0003440 3267 0030 6c66 6761 3132 6600 616c 3267
0003450 0032 6c66 6761 3332 6600 616c 3267 0034
0003460 6c66 6761 3532 7400 6568 665f 616c 0067
0003470 445f 4e59 4d41 4349 5f00 6465 7461 0061
0003480 655f 646e 5f00 625f 7373 735f 6174 7472
0003490 6d00 6961 006e 2e00 7973 746d 6261 2e00
00034a0 7473 7472 6261 2e00 6873 7473 7472 6261
00034b0 2e00 6e69 6574 7072 2e00 6e67 2e75 6168
00034c0 6873 2e00 7964 736e 6d79 2e00 7964 736e
00034d0 7274 2e00 6574 7478 2e00 6865 665f 6172
00034e0 656d 2e00 7964 616e 696d 0063 642e 7461
00034f0 0061 0000 0000 0000 0000 0000 0000 0000
0003500 0000 0000 0000 0000 0000 0000 0000 0000
*
```

```
0003530 0000 0000 0000 0000 001b 0000 0001 0000
0003540 0002 0000 0000 0000 0200 0040 0000 0000
0003550 0200 0000 0000 0000 001c 0000 0000 0000
0003560 0000 0000 0000 0000 0001 0000 0000 0000
0003570 0000 0000 0000 0000 0027 0000 0005 0000
0003580 0002 0000 0000 0000 0220 0040 0000 0000
0003590 0220 0000 0000 0000 0010 0000 0000 0000
00035a0 0004 0000 0000 0000 0008 0000 0000 0000
00035b0 0004 0000 0000 0000 0023 0000 fff6 6fff
00035c0 0002 0000 0000 0000 0230 0040 0000 0000
00035d0 0230 0000 0000 0000 001c 0000 0000 0000
00035e0 0004 0000 0000 0000 0008 0000 0000 0000
00035f0 0000 0000 0000 0000 002d 0000 000b 0000
0003600 0002 0000 0000 0000 0250 0040 0000 0000
0003610 0250 0000 0000 0000 0018 0000 0000 0000
0003620 0005 0000 0001 0000 0008 0000 0000 0000
0003630 0018 0000 0000 0000 0035 0000 0003 0000
0003640 0002 0000 0000 0000 0268 0040 0000 0000
0003650 0268 0000 0000 0000 000b 0000 0000 0000
0003660 0000 0000 0000 0000 0001 0000 0000 0000
0003670 0000 0000 0000 0000 003d 0000 0001 0000
0003680 0006 0000 0000 0000 1000 0040 0000 0000
0003690 1000 0000 0000 0000 013e 0000 0000 0000
00036a0 0000 0000 0000 0000 0010 0000 0000 0000
00036b0 0000 0000 0000 0000 0043 0000 0001 0000
00036c0 0002 0000 0000 0000 2000 0040 0000 0000
00036d0 2000 0000 0000 0000 0000 0000 0000 0000
00036e0 0000 0000 0000 0000 0008 0000 0000 0000
00036f0 0000 0000 0000 0000 004d 0000 0006 0000
0003700 0003 0000 0000 0000 2f20 0040 0000 0000
0003710 2f20 0000 0000 0000 00e0 0000 0000 0000
0003720 0005 0000 0000 0000 0008 0000 0000 0000
0003730 0010 0000 0000 0000 0056 0000 0001 0000
0003740 0003 0000 0000 0000 3000 0040 0000 0000
0003750 3000 0000 0000 0000 0052 0000 0000 0000
0003760 0000 0000 0000 0000 0004 0000 0000 0000
0003770 0000 0000 0000 0000 0001 0000 0002 0000
0003780 0000 0000 0000 0000 0000 0000 0000 0000
0003790 3058 0000 0000 0000 0360 0000 0000 0000
00037a0 000b 0000 0020 0000 0008 0000 0000 0000
00037b0 0018 0000 0000 0000 0009 0000 0003 0000
00037c0 0000 0000 0000 0000 0000 0000 0000 0000
00037d0 33b8 0000 0000 0000 00de 0000 0000 0000
00037e0 0000 0000 0000 0000 0001 0000 0000 0000
00037f0 0000 0000 0000 0000 0011 0000 0003 0000
0003800 0000 0000 0000 0000 0000 0000 0000 0000
0003810 3496 0000 0000 0000 005c 0000 0000 0000
0003820 0000 0000 0000 0000 0001 0000 0000 0000
*
0003838
```