Teensy Tiny ELF Programs

inspired by Brian Raiter

Roland Hieber

Stratum 0 e. V.

March 15, 2013

Hello World

```
#include <stdio.h>

int main(int argc, char** argv) {
printf("Hello World!\n");
return 42;
}
```

Well, how big can it be?

Hello World

```
#include <stdio.h>

int main(int argc, char** argv) {
printf("Hello World!\n");
return 42;
}
```

Well, how big can it be?

```
$ gcc hello.c
$ wc -c a.out
4483 a.out
```

Oops.

Hello World

- Okay. Maybe don't print anything, just return a value.

```
$ echo 'main(){return 42;}' | gcc -x c -
$ wc -c a.out
4359 a.out
```

Hello World

- Okay. Maybe don't print anything, just return a value.

```
$ echo 'main(){return 42;}' | gcc -x c -
$ wc -c a.out
4359 a.out
```

- Oh okay, we forgot to optimize for size and strip the executable.

```
$ echo 'main(){return 42;}' | gcc -x c -s -Os -
$ wc -c a.out
2756 a.out
```

Hello World

- Okay. Maybe don't print anything, just return a value.

```
$ echo 'main(){return 42;}' | gcc -x c -
$ wc -c a.out
4359 a.out
```

- Oh okay, we forgot to optimize for size and strip the executable.

```
$ echo 'main(){return 42;}' | gcc -x c -s -Os -
$ wc -c a.out
2756 a.out
```

- Minimal C program has still 2.7 KB. Meh.

Next step: Assembler

```
; tiny.asm
BITS 32
GLOBAL main
SECTION .text
main:
   mov      eax, 42
   ret
```

Next step: Assembler

```
; tiny.asm
BITS 32
GLOBAL main
SECTION .text
main:
   mov      eax, 42
   ret

$ nasm -f elf tiny.asm
$ gcc -Wall -s tiny.o
$ ./a.out ; echo $?
42
$ wc -c a.out
    2604 a.out
```

Deeper into the Rabbit Hole: libc

```
; tiny.asm
BITS 32
EXTERN _exit
GLOBAL _start
SECTION .text
_start:
  push    dword 42
  call    _exit
```

Deeper into the Rabbit Hole: libc

```
; tiny.asm
BITS 32
EXTERN _exit
GLOBAL _start
SECTION .text
_start:
  push    dword 42
  call    _exit

$ nasm -f elf tiny.asm
$ gcc -Wall -s -nostartfiles tiny.o
$ ./a.out ; echo $?
42
$ wc -c a.out
   1340 a.out
```

But...do we even need libc?

```
; tiny.asm
BITS 32
GLOBAL _start
SECTION .text
_start:
  mov     eax, 1   ; "exit" syscall, see unistd.h
  mov     ebx, 42
  int     0x80
```

But...do we even need libc?

```
; tiny.asm
BITS 32
GLOBAL _start
SECTION .text
_start:
  mov      eax, 1   ; "exit" syscall, see unistd.h
  mov      ebx, 42
  int      0x80

$ nasm -f elf tiny.asm
$ gcc -Wall -s -nostdlib tiny.o
$ ./a.out ; echo $?
42
$ wc -c a.out
  372 a.out
```

Okay, what does our executable contain?

```
$ objdump -x a.out | less
[...]
  Sections:
  Idx Name          Size      VMA       LMA       File off  Algn
    0 .text         00000007  08048080  08048080  00000080  2**4
                    CONTENTS, ALLOC, LOAD, READONLY, CODE
    1 .comment      0000001c  00000000  00000000  00000087  2**0
                    CONTENTS, READONLY
[...]

$ hexdump a.out
[...]
00000080: 31C0 40B3 2ACD 8000 5468 6520 4E65 7477  1.@.*...The Netw
00000090: 6964 6520 4173 7365 6D62 6C65 7220 302E  ide Assembler 0.
000000A0: 3938 0000 2E73 796D 7461 6200 2E73 7472  98...symtab..str
[...]
```

NASM, that bitch.

Time for some black magic: let's write ELF directly.

```
; tiny.asm
BITS 32
        org  0x08048000
ehdr:                                           ; Elf32_Ehdr, see <linux/elf.h>
         db  0x7F, "ELF", 1, 1, 1, 0 ;  e_ident
times 8 db  0                        ;    (padding)
         dw  2                       ;  e_type
         dw  3                       ;  e_machine
         dd  1                       ;  e_version
         dd  _start                  ;  e_entry
         dd  phdr - $$               ;  e_phoff
         dd  0                       ;  e_shoff
         dd  0                       ;  e_flags
         dw  ehdrsize                ;  e_ehsize
         dw  phdrsize                ;  e_phentsize
         dw  1                       ;  e_phnum
         dw  0                       ;  e_shentsize
         dw  0                       ;  e_shnum
         dw  0                       ;  e_shstrndx
ehdrsize  equ  $ - ehdr
```

Time for some black magic: let's write ELF directly.

```
; tiny.asm, cont'd
phdr:                                   ; Elf32_Phdr
        dd  1                           ;  p_type
        dd  0                           ;  p_offset
        dd  $$                          ;  p_vaddr
        dd  $$                          ;  p_paddr
        dd  filesize                    ;  p_filesz
        dd  filesize                    ;  p_memsz
        dd  5                           ;  p_flags
        dd  0x1000                      ;  p_align
phdrsize  equ $ - phdr

_start:
  mov     eax, 1  ; "exit" syscall, see unistd.h
  mov     ebx, 42
  int     0x80
filesize      equ     $ - $$
```

Time for some black magic: let's write ELF directly.

```
$ nasm -f bin -o a.out tiny.asm
$ chmod +x a.out
$ ./a.out ; echo $?
42
$ wc -c a.out
      91 a.out
```

91 Bytes, not bad!
But the ELF header still contains to many unused bytes.

Wait. . . the spec doesn't forbid *overlapping* headers. . .

```
; tiny.asm
BITS 32
          org     0x00200000
          db      0x7F, "ELF"          ; e_ident
          db      1, 1, 1, 0, 0
_start:   mov     bl, 42               ;  (padding)
          xor     eax, eax             ;  (no wait)
          inc     eax                  ;  (what?)
          int     0x80                 ;  (ohhh, cunning plan!)
          dw      2                    ; e_type
          dw      3                    ; e_machine
          dd      1                    ; e_version
          dd      _start               ; e_entry
          dd      phdr - $$            ; e_phoff
phdr:     dd      1                    ; e_shoff        ; p_type
          dd      0                    ; e_flags        ; p_offset
          dd      $$                   ; e_ehsize       ; p_vaddr
                                       ; e_phentsize
          dw      1                    ; e_phnum        ; p_paddr
          dw      0                    ; e_shentsize
          dd      filesize             ; e_shnum        ; p_filesz
                                       ; e_shstrndx
          dd      filesize                              ; p_memsz
          dd      5                                     ; p_flags
          dd      0x1000                                ; p_align
filesize  equ     $ - $$
```

Wait... the spec doesn't forbid *overlapping* headers...

...other dirty hacks like eliminating bytes that are not read by the loader anyway...

```
$ nasm -f bin -o a.out tiny.asm
$ chmod +x a.out
$ ./a.out ; echo $?
42
$ wc -c a.out
      45 a.out
```

45 bytes for a valid Linux executable?! \o/
(okay, "valid"... probably only works on Linux, but hey, it works!)

So, can we still do better?

Unfortunately not.
There is no way to eliminate the last byte at file offset 45, which specifies the location of the program header. This byte must be at position 45, and there is no way around it.
On the other hand, we started out with 4.3 Kilobyte, and now we have 45 Byte. That is quite an achievement.

Sources

Brian Raiter: A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux. July 2, 1999.
http://www.muppetlabs.com/~breadbox/software/tiny/teensy.html

This presentation was carefully copy-pasted from there by hand. No code segments were harmed in the making of this presentation.

(And since the original post did not specify a licence, I am probably doomed now. Brian may forgive me, but his write-up is just so excellent.)