

Network Security Assignment #2

JOHN THOMAS
P2CSN17017

PART A - Written Assignment -1

1. What is the difference between a binary file and a text file?

A text file stores data in ASCII characters i.e, with a maximum length of 255 characters . Each line of a text file is terminated (delimited) with a special character known as EOL (end of line) character. Some internal translations take place when this EOL character is read or written.

By default our files are created and opened in text mode.

A binary file contains information in the same format as it is held in memory i.e, (0 or 1). There is no delimiter for a line. Also, no translations occur in binary files. Binary files are faster and easier for a program to read and write than are text files.

2. What is the ELF FILE Format?

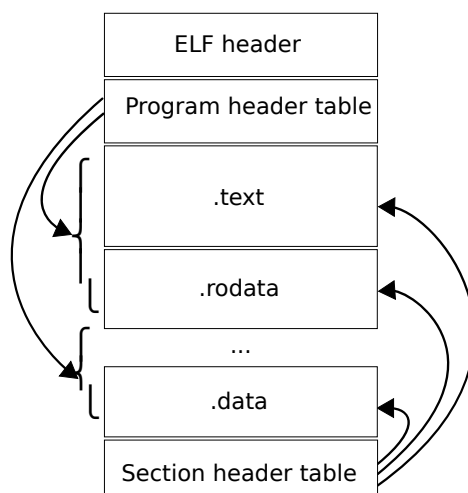
In computing, the Executable and Linkable Format (ELF, formerly named Extensible Linking Format), is a common standard file format for executable files, object code, shared libraries, and core dumps.

Each ELF file is made up of one ELF header, followed by file data. The data can include:

Program header table, describing zero or more memory segment.

Section header table, describing zero or more section.

Data referred to by entries in the program header table or section header table.



3. What is a hexdump? What kind of information is stored in a hexdump?

In computing, a hex dump is a hexadecimal view (on screen or paper) of computer data, from RAM or from a file or storage device. In a hex dump, each byte (8-bits) is represented as a two-digit hexadecimal number.

Hex dumps are commonly organized into rows of 8 or 16 bytes, sometimes separated by whitespaces. Some hex dumps have the hexadecimal memory address at the beginning and/or a checksum byte at the end of each line.

4. Diagrammatically represent how the hex characters “DEADBEEF” will be represented in Big Endian and Little Endian formats.

Endianness means the order in which the bytes of a value larger than one byte are stored in memory.

Little-endian machines store the least significant byte on the lowest memory address (the word is stored little-end-first).

Big-endian machines store the most significant byte on the lowest memory address (the word is stored big-end-first).

Let the start address be 100 for storing hex characters.

In Big Endian

100	101	102	103	104	105	106
DE	AD	BE	EF			

In Little Endian

100	101	102	103	104	105	106
EF	BE	AD	DE			

PART B - Programming Assignment -2

1. Write a simple Hello World program in c. The program should have nothing but a #include and a print statement in main. Write a makefile to generate an executable called 'hello' Record the size in number of bytes of the hello world program

```
john@johnndt:~/Desktop
john@johnndt:~/Desktop$ cd Desktop/
john@johnndt:~/Desktop$ gcc helloworld.c
john@johnndt:~/Desktop$ ./a.out
Hello, World!john@johnndt:~/Desktop$
```

```
(~/Desktop) - gedit
1
2 hello: Hello.c
3 gcc -m32 -o hello Hello.c
4
5 Hello.o: Hello.c
6 gcc -c Hello.c
```

```
johnndt:~/Desktop
john@johnndt:~/Desktop$ cd Desktop/
john@johnndt:~/Desktop$ ls
a2.odt a.out example1a.tcl hello Hello.c Makefile out.nam
john@johnndt:~/Desktop$ make
make: 'hello' is up to date.
john@johnndt:~/Desktop$ ./hello
Hello, World!john@johnndt:~/Desktop$
```

Size in bytes = 7.3 kB (7348 bytes)

```
Properties
Basic Permissions Open With
Name: hello
Type: executable (application/x-executable)
Size: 7.3 kB (7,348 bytes)
Location: /home/john/Desktop
Accessed: Wed, Feb 14 2018 21:25:39
Modified: Wed, Feb 14 2018 21:25:39
```

2. Read and dump the ELF header from the executable type the command

readelf -h hello

```
hello, World!john@johnndt:~/Desktop$ readelf -h hello
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:             ELF32
  Data:              2's complement, little endian
  Version:           1 (current)
  OS/ABI:            UNIX - System V
  ABI Version:       0
  Type:              EXEC (Executable file)
  Machine:           Intel 80386
  Version:           0x1
  Entry point address: 0x8048310
  Start of program headers: 52 (bytes into file)
  Start of section headers: 6108 (bytes into file)
  Flags:             0x0
  Size of this header: 52 (bytes)
  Size of program headers: 32 (bytes)
  Number of program headers: 9
  Size of section headers: 40 (bytes)
  Number of section headers: 31
  Section header string table index: 28
```

3. ldd hello

```
john@johnndt:~/Desktop$ ldd hello
linux-gate.so.1 => (0xf7f9d000)
libc.so.6 => /lib32/libc.so.6 (0xf7dc5000)
/lib/ld-linux.so.2 (0xf7f9f000)
```

4. file hello

```
/lib/ld-linux.so.2 (0xf7f9f000)
john@johnndt:~/Desktop$ file hello
hello: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=915f10943a149710fe979e7f27524c7ee424b1b3, not stripped
john@johnndt:~/Desktop$
```

5. What does the strip command do?

strip -s hello

```
john@johnndt:~/Desktop$ strip -s hello
john@johnndt:~/Desktop$ file hello
hello: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=915f10943a149710fe979e7f27524c7ee424b1b3, stripped
john@johnndt:~/Desktop$
```

The symbol table can be stripped from an object file using -s option of strip command

6.,7.,8

```
johng@johndt:~/Desktop$ cd Desktop/
johng@johndt:~/Desktop$ ls
a2.odt  a.out  example1a.tcl  hello  hello.asm  Hello.c  Makefile  out.nam
johng@johndt:~/Desktop$ nasm -f elf hello.asmjohng@johndt:~/Desktop$ gcc -o m32 hello hello.o -nostartfiles -nostdlib -nodefaultlibs
hello.o: In function `main':
hello.asm:(.text+0x0): multiple definition of `main'
hello:(.text+0x0): first defined here
/usr/bin/ld: warning: Cannot create .note.gnu.build-id section, --build-id ignored.
/usr/bin/ld: i386 architecture of input file `hello' is incompatible with i386:x86-64 output
/usr/bin/ld: i386 architecture of input file `hello.o' is incompatible with i386:x86-64 output
/usr/bin/ld: warning: cannot find entry symbol _start; defaulting to 000000000400110
collect2: error: ld returned 1 exit status
johng@johndt:~/Desktop$ gcc -o m32 hello hello.o -nostartfiles -nostdlib -nodefaultlibs
/usr/bin/ld: unrecognized emulation mode: 32
Supported emulations: elf_x86_64 elf32_x86_64 elf_i386 elf_iarmcu i386linux elf_l1om elf_k1om i386pep i386pe
collect2: error: ld returned 1 exit status
johng@johndt:~/Desktop$ gcc -m32 -o hello hello.o -nostartfiles -nostdlib -nodefaultlibs
/usr/bin/ld: warning: cannot find entry symbol _start; defaulting to 00000000080480c0
johng@johndt:~/Desktop$ strip -s hello
strip: '-s': No such file
johng@johndt:~/Desktop$ strip -s hello
johng@johndt:~/Desktop$ ldd hello
not a dynamic executable
johng@johndt:~/Desktop$
```

9.

```
hello - GHex
00000000 7F 45 4C 46 01 01 01 00 00 00 00 00 00 00 02 00 03 00 01 00 00 00 C0 80 04 08 34
00000010 00 00 00 18 01 00 00 00 00 00 00 00 34 00 20 00 03 00 28 00 05 00 04 00 01 00 00 00 00
0000003A 00 00 00 80 04 08 00 80 04 08 E2 00 00 E2 00 05 00 00 00 00 10 00 00 00 01 00 00
00000057 00 E4 00 00 00 E4 90 04 08 E4 90 04 08 09 00 00 00 09 00 00 00 06 00 00 00 10 00 00
00000074 04 00 00 00 94 00 00 00 94 80 04 08 94 80 04 08 24 00 00 00 24 00 00 00 04 00 00 04
00000091 00 00 00 04 00 00 00 14 00 00 00 03 00 00 00 47 4E 55 00 1F 11 4B B9 20 09 08 26 05 BE
000000AE 7E D6 D7 E0 0A 26 3D B8 07 66 00 00 00 00 00 00 00 00 BA 09 00 00 00 B9 E4 90 04 08 BB
000000CB 01 00 00 00 88 04 00 00 00 CD 80 BB 00 00 00 00 88 01 00 00 00 CD 80 00 00 42 55 53 54
000000E8 45 44 74 72 74 61 62 00 2E 6F 74 65 2E 67 6E 75 2E 62 75 69 6C 64 2D 69 64 00 2E 74
00000105 65 78 74 00 2E 64 61 74 61 44 45 41 44 42 45 45 46 44 45 41 44
```

10. Record the number of bytes on the executable. It should have come down to less than 300 bytes

```
john@johnnt:~/Desktop$ wc -c hello
282 hello
john@johnnt:~/Desktop$
```

282 bytes