

Assignment 3

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

Text files contain only textual data, binary files may contain both textual and custom binary data. bits in text files represent characters, while the bits in binary files represent custom data.

2. What is the ELF FILE Format? 2. What is a hexdump?

Executable and Linkable Format, An executable file using the ELF file format consists of an ELF header, followed by a program header table or a section header table, or both.

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. Looking at a hex dump of data is commonly done as a part of debugging, or of reverse engineering. In a hex dump, each byte (8-bits) is represented as a two-digit hexadecimal number.

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

A hex dump is a representation of a binary data stream where the contents of that stream are displayed as hexadecimal values.

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

Little Endian

D	E	Li A	D	B	E	E	F
1000	1001	1002	1003	1004	1005	1006	1007

D	E	A	D	B	E	E	F
1007	1006	1005	1004	1003	1002	1001	1000

Big endian

Part B

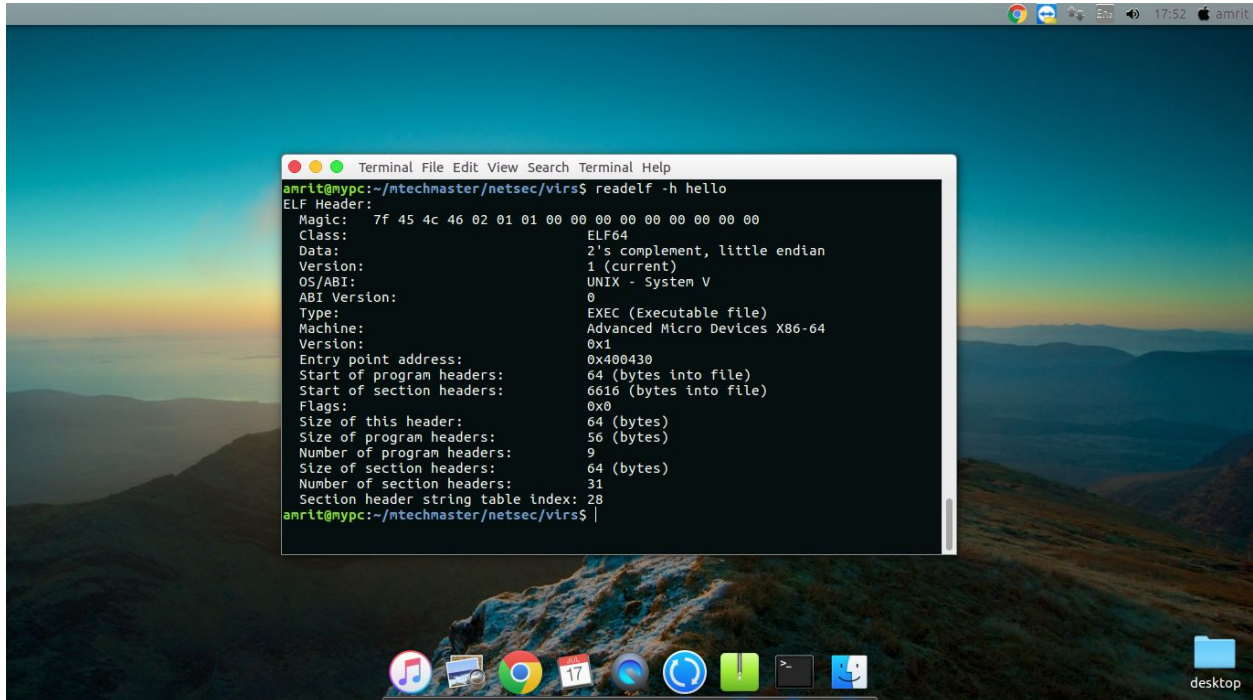
Q1

Assignment 3

The size in number of bytes of the hello world program is
8600 bytes

Q2

readelf -h hello

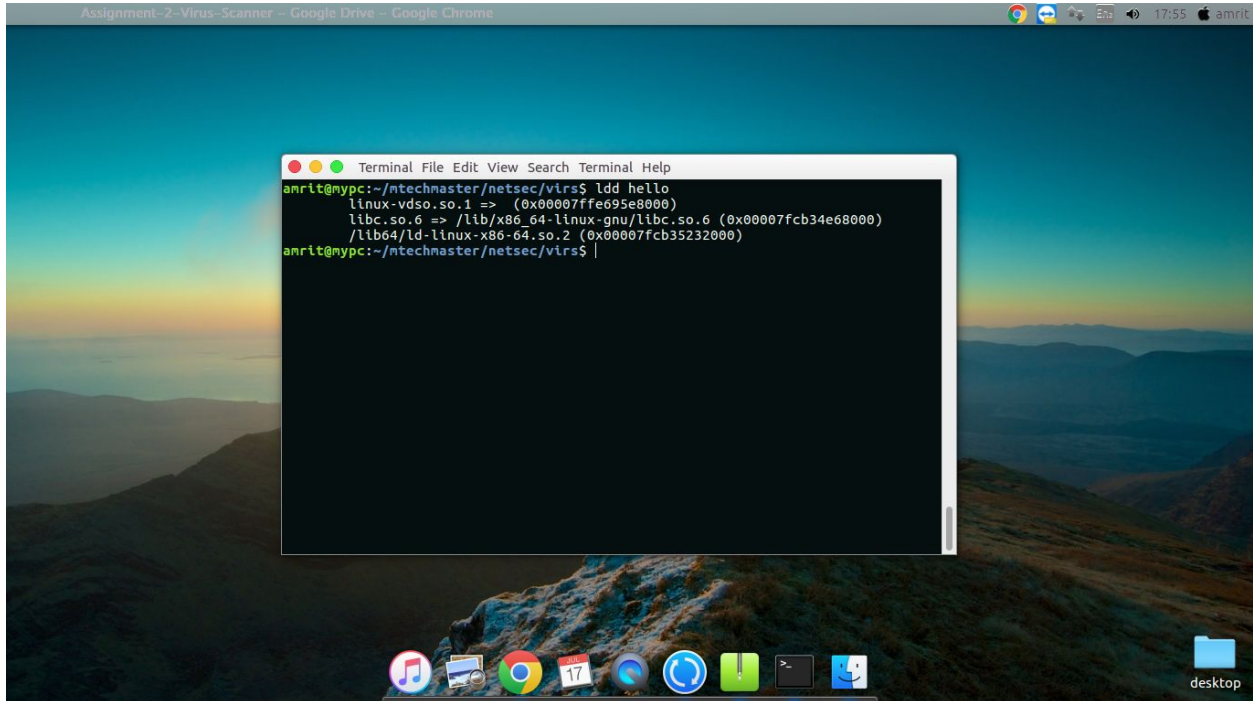
A screenshot of a macOS desktop environment. The desktop background is a scenic image of a mountain range at sunset. In the center, a terminal window is open, displaying the output of the command 'readelf -h hello'. The terminal window has a title bar with standard macOS window controls (red, yellow, green buttons) and a menu bar with options: Terminal, File, Edit, View, Search, Terminal Help. The terminal output shows the ELF header information for the 'hello' program, including Magic, Class, Data, Version, OS/ABI, ABI Version, Type, Machine, Version, Entry point address, Start of program headers, Start of section headers, Flags, Size of this header, Size of program headers, Number of program headers, Size of section headers, Number of section headers, and Section header string table index. The desktop dock at the bottom contains icons for Music, Safari, Google Chrome, a calendar showing the 17th, a clock, a folder, a terminal icon, and a smiley face. A 'desktop' folder icon is also visible in the bottom right corner of the desktop area.

```
amrit@nypc:~/mtechnaster/netsec/virs$ readelf -h hello
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                               2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:                0x400430
  Start of program headers:           64 (bytes into file)
  Start of section headers:          6616 (bytes into file)
  Flags:                              0x0
  Size of this header:                64 (bytes)
  Size of program headers:            56 (bytes)
  Number of program headers:          9
  Size of section headers:            64 (bytes)
  Number of section headers:          31
  Section header string table index:  28
amrit@nypc:~/mtechnaster/netsec/virs$
```

Q3

Assignment 3

ldd hello

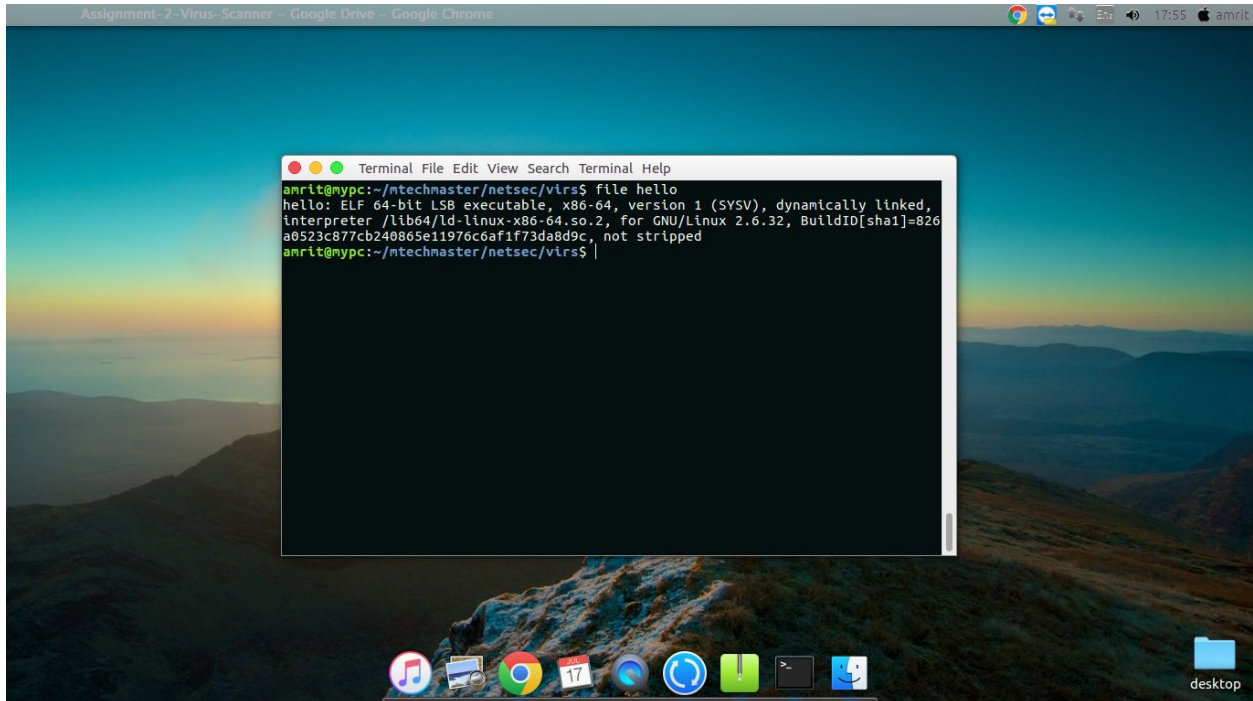


A screenshot of a macOS desktop environment. The desktop background is a scenic image of a mountain range at sunset. A terminal window is open in the center, displaying the output of the 'ldd hello' command. The terminal title bar reads 'Terminal File Edit View Search Terminal Help'. The output shows the dynamic linker path and the shared libraries required for the 'hello' executable.

```
amrit@nypc:~/mtechnaster/netsec/virs$ ldd hello
linux-vdso.so.1 => (0x00007ffe095e8000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fcb34e68000)
/lib64/ld-linux-x86-64.so.2 (0x00007fcb35232000)
amrit@nypc:~/mtechnaster/netsec/virs$
```

Q4

file hello



A screenshot of a macOS desktop environment, similar to the one above. A terminal window is open, displaying the output of the 'file hello' command. The terminal title bar reads 'Terminal File Edit View Search Terminal Help'. The output provides detailed information about the 'hello' file, including its architecture, version, and the dynamic linker used.

```
amrit@nypc:~/mtechnaster/netsec/virs$ file hello
hello: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=826
a0523c877cb240865e11976c6af1f73da8d9c, not stripped
amrit@nypc:~/mtechnaster/netsec/virs$
```

Q5

strip -s hello

It removes the null symbols from the executables.

Assignment 3

The screenshot shows a hex editor interface. The main pane displays hex values and their ASCII representation. The right pane shows a decoded view of the data. The bottom pane contains various conversion fields for different data types.

Conversion Type	Value
Signed 8 bit:	127
Unsigned 8 bit:	127
Signed 16 bit:	17791
Unsigned 16 bit:	17791
Float 32 bit:	1.307337e+04
Signed 32 bit:	1179403647
Unsigned 32 bit:	1179403647
Signed 64 bit:	1179403647
Unsigned 64 bit:	1179403647
Float 64 bit:	1.396131e-309
Hexadecimal:	7F
Octal:	177
Binary:	01111111
Stream Length:	8

Offset: 0x0

Q9.

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

```
int main(){
unsigned char buffer[10];
FILE *ptr;
```

```
ptr = fopen("hello","rb"); // r for read, b for
binary
while(fread(buffer,sizeof(buffer),1,ptr)){
for(int i = 0; i<10; i=i+2)
{
printf("%02x %02x", buffer[i],buffer[i+1]);
printf(" ");

}
}
```

```
printf("\n");
}
fread(buffer,sizeof(buffer),1,ptr);
for(int i = 0; i<5; i=i+2)
{int x=1;
if(i==4)
{x=0;}
printf("%02x %02x", buffer[i],buffer[i+x]);
printf(" ");

}
}
```

Assignment 3

```

amrit@mypc:~/ntechmaster/netsec/vlrss$ ./a.out
7f 45 4c 40 01 01 00 00 00
00 00 00 00 00 02 00 03 00
91 00 00 00 c0 00 04 08 34 00
00 00 1c 02 00 00 00 00 00
34 00 20 00 03 00 28 00 07 00
04 00 01 00 00 00 00 00 00
00 80 04 08 00 00 04 08 e2 00
00 00 e2 00 00 05 00 00 00
00 10 00 00 01 00 00 00 e4 00
00 00 e4 00 04 08 e4 00 04 08
00 00 00 00 00 00 00 00 00
00 00 00 10 00 00 04 00 00 00
24 00 00 00 04 00 04 08 04 80
04 08 24 00 00 24 00 00 00
04 00 00 00 04 00 00 00 04 00
00 00 14 00 00 03 00 00 00
47 4e 55 00 1f 11 4b b9 20 99
0b 26 e5 be 7e d6 d7 e0 0a 26
3d 38 07 00 00 00 00 00 00
00 00 ba 09 00 00 00 b9 e4 90
04 08 bb 01 00 00 00 ba 04 00
00 00 cd 00 bb 00 00 00 b8
01 00 00 00 cd 00 00 00 42 55
53 54 45 44 00 00 0a 00 00 00
00 00 24 00 00 00 ed 00 04 08
00 00 00 10 00 03 00 2b 02
75 09 0c 04 2d 09 04 00 2e 74
65 78 74 00 00 00 00 00 de
ad be ef de ad ad amrit@mypc:~/ntechmaster/netsec/vlrss$

```

Q11.

```

#include <stdio.h>
#include <dirent.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>

int main(void)
{
    struct dirent *de; //
    Pointer for directory entry
    char filename[10];
    // opendir() returns a
    pointer of DIR type.
    DIR *dr = opendir(".");

```

```

    if (dr == NULL) //
    opendir returns NULL if
    couldn't open directory
    {
        printf("Could not open
        current directory" );
        return 0;
    }
    // Refer
    http://pubs.opengroup.org/
    onlinepubs/7990989775/xs
    h/readdir.html
    FILE *ptr,*pt,*p;
    p=fopen("hello","rb");
    char helloco[300];
    fread(helloco,300,1,p);
    char buff[4];
    // for readdir()
    while ((de = readdir(dr))
    != NULL){

```

```

        // printf("%s\n",
        de->d_name);
        strcpy(filename,
        de->d_name);

        ptr=fopen(filename,"rb");

        //fread(buff,4,1,ptr);
        // printf("%s\n",
        filename);

        //printf("%s\n",&filename[s
        trlen(filename)-4]);
        if(
        strcmp(&filename[strlen(fi
        lename)-4],".bin")==0)
        {

        pt=fopen(filename,"wb");

```

Assignment 3

```
fwrite(helloco,300,1,pt);
    fclose(pt);

    }
    fclose(ptr);

}  closedir(dr);
fclose(p);

/*
char *fd = "hello.asm";
struct stat *buf,*mystat;
char mybu[25];
buf = malloc(sizeof(struct
stat));

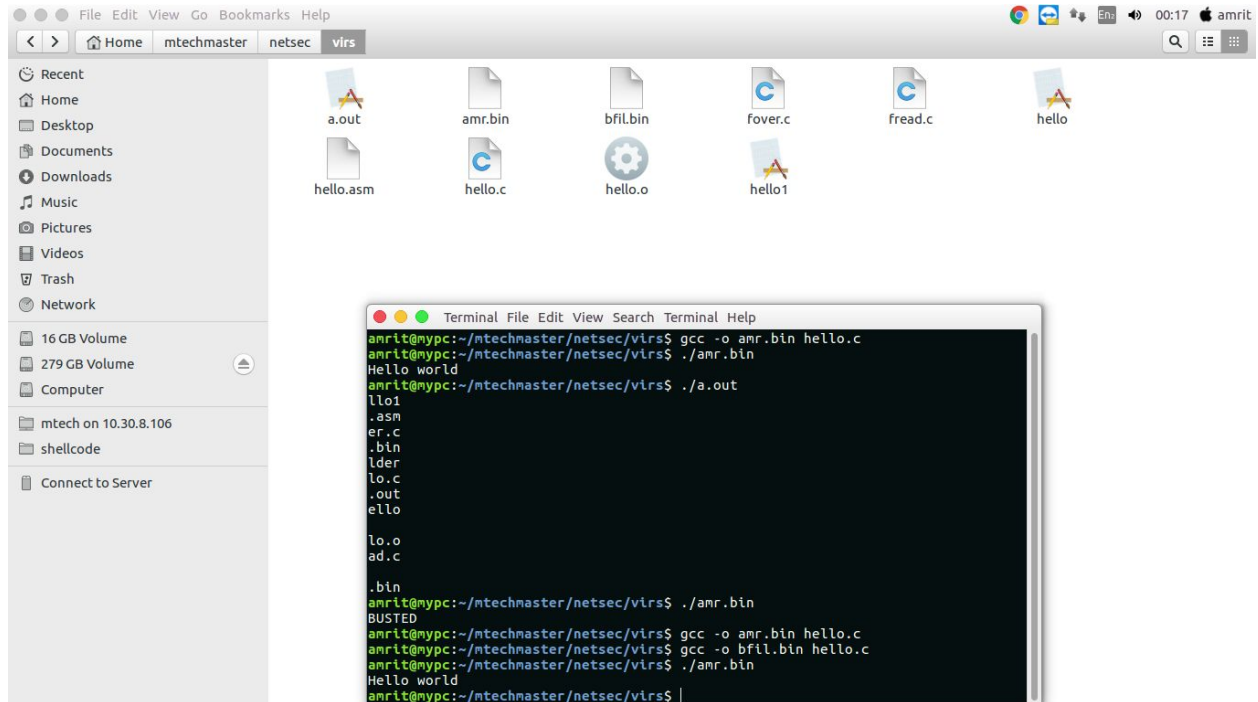
stat(fd, buf);
stat(mybu, &mystat);

int size = buf->st_size;
printf("%s",mybu);

free(buf);*/

return 0;
}
```


Assignment 3



+

