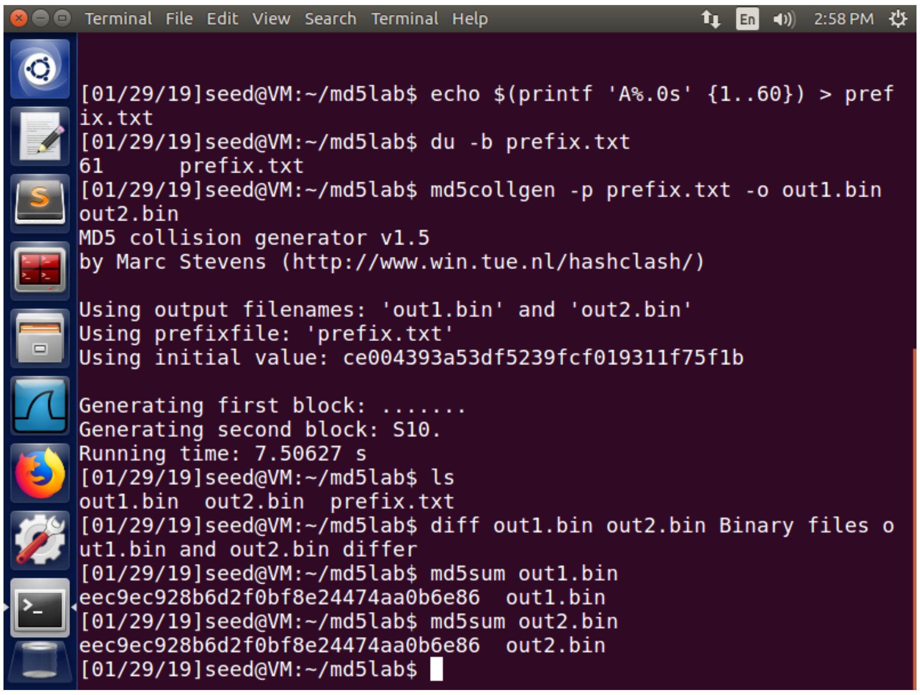
# MD5 Collision Attack Lab

## Task 1: Generating Two Different Files with the Same MD5 Hash

For this task, we use the following command to write arbitrary A's into the prefix.txt file.

```
$ echo $(printf 'A%.0s' {1..x}) > prefix.txt
```

**Question 1**. If the length of your prefix file is not multiple of 64, what is going to happen?



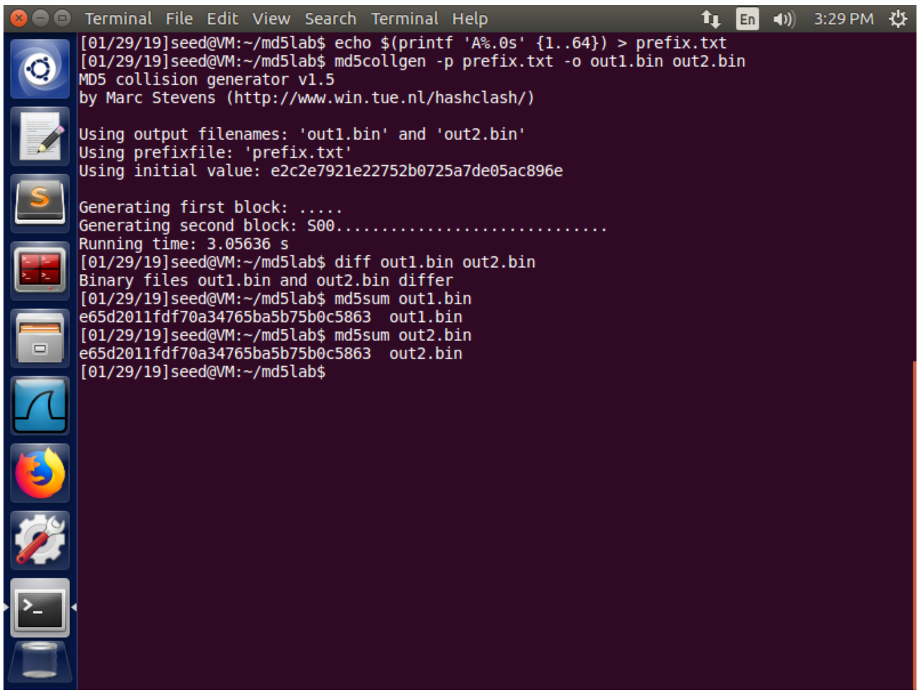
```
[01/29/19]seed@VM:~/md5lab$ echo $(printf 'A%.0s' {1..60}) > pref
ix.txt
[01/29/19]seed@VM:~/md5lab$ du -b prefix.txt
61      prefix.txt
[01/29/19]seed@VM:~/md5lab$ md5collgen -p prefix.txt -o out1.bin
out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: ce004393a53df5239fcf019311f75f1b

Generating first block: .......
Generating second block: S10.
Running time: 7.50627 s
[01/29/19]seed@VM:~/md5lab$ ls
out1.bin  out2.bin  prefix.txt
[01/29/19]seed@VM:~/md5lab$ diff out1.bin out2.bin Binary files o
ut1.bin and out2.bin differ
[01/29/19]seed@VM:~/md5lab$ md5sum out1.bin
eec9ec928b6d2f0bf8e24474aa0b6e86  out1.bin
[01/29/19]seed@VM:~/md5lab$ md5sum out2.bin
eec9ec928b6d2f0bf8e24474aa0b6e86  out2.bin
[01/29/19]seed@VM:~/md5lab$ 
```

As shown in the screenshot, the prefix file has 60 A's. Although out1.bin are different from out2.bin, they have the same md5 hash string.

**Question 2**. Create a prefix file with exactly 64 bytes, and run the collision tool again, and see what happens.



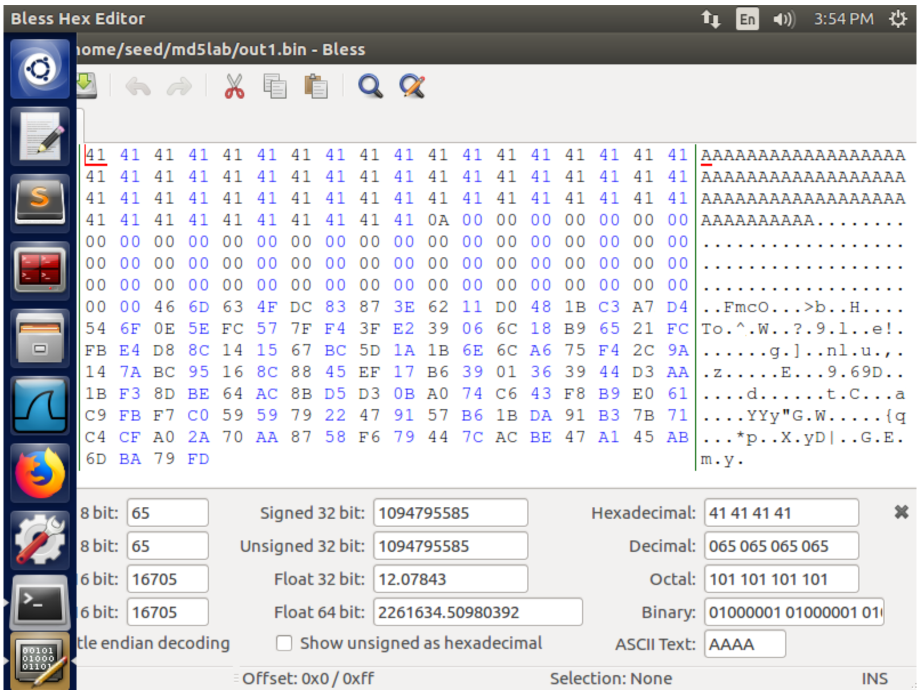
```
[01/29/19]seed@VM:~/md5lab$ echo $(printf 'A%.0s' {1..64}) > prefix.txt
[01/29/19]seed@VM:~/md5lab$ md5collgen -p prefix.txt -o out1.bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: e2c2e7921e22752b0725a7de05ac896e

Generating first block: .....
Generating second block: S00............................
Running time: 3.05636 s
[01/29/19]seed@VM:~/md5lab$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[01/29/19]seed@VM:~/md5lab$ md5sum out1.bin
e65d2011fdf70a34765ba5b75b0c5863  out1.bin
[01/29/19]seed@VM:~/md5lab$ md5sum out2.bin
e65d2011fdf70a34765ba5b75b0c5863  out2.bin
[01/29/19]seed@VM:~/md5lab$
```

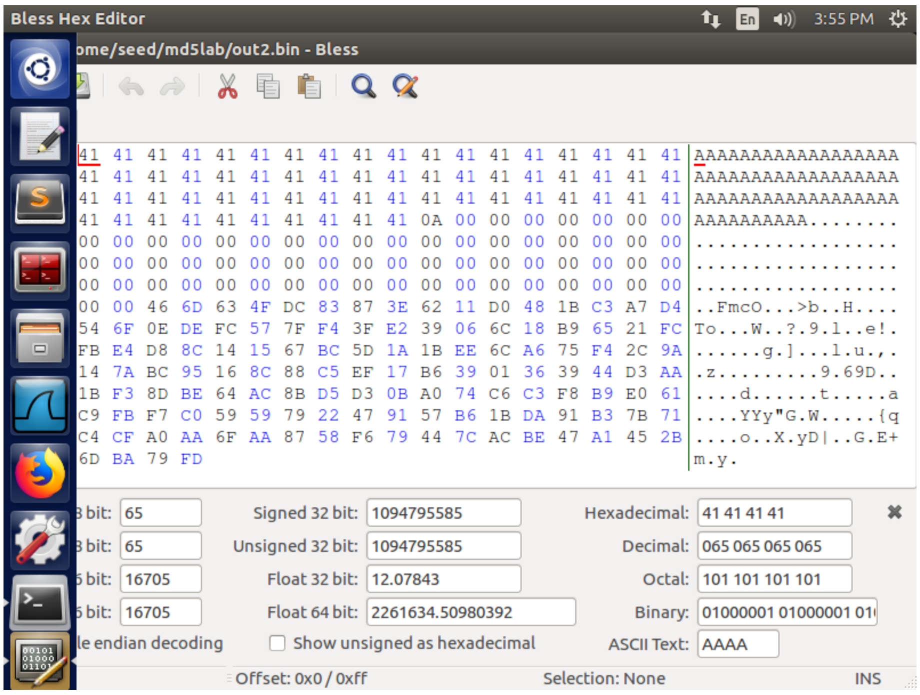
As shown in the screenshot, out1.bin is still different from out2.bin, but they have the same hash string.

**Question 3**. Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different.



```
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 0A 00 00 00 00 00 00 00 AAAAAAAAAA........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00 00 46 6D 63 4F DC 83 87 3E 62 11 D0 48 1B C3 A7 D4 ..FmcO...>b..H....
54 6F 0E 5E FC 57 7F F4 3F E2 39 06 6C 18 B9 65 21 FC To.^.W..?.9.l..e!.
FB E4 D8 8C 14 15 67 BC 5D 1A 1B 6E 6C A6 75 F4 2C 9A ......g.]..nl.u.,.
14 7A BC 95 16 8C 88 45 EF 17 B6 39 01 36 39 44 D3 AA .z.....E...9.69D..
1B F3 8D BE 64 AC 8B D5 D3 0B A0 74 C6 43 F8 B9 E0 61 ....d......t.C...a
C9 FB F7 C0 59 59 79 22 47 91 57 B6 1B DA 91 B3 7B 71 ....YYy"G.W.....{q
C4 CF A0 2A 70 AA 87 58 F6 79 44 7C AC BE 47 A1 45 AB ...*p..X.yD|..G.E.
6D BA 79 FD                                           m.y.
```

```
8 bit: 65      Signed 32 bit: 1094795585        Hexadecimal: 41 41 41 41
8 bit: 65      Unsigned 32 bit: 1094795585      Decimal: 065 065 065 065
6 bit: 16705   Float 32 bit: 12.07843           Octal: 101 101 101 101
6 bit: 16705   Float 64 bit: 2261634.50980392   Binary: 01000001 01000001 01
tle endian decoding   [ ] Show unsigned as hexadecimal   ASCII Text: AAAA
Offset: 0x0 / 0xff          Selection: None          INS
```



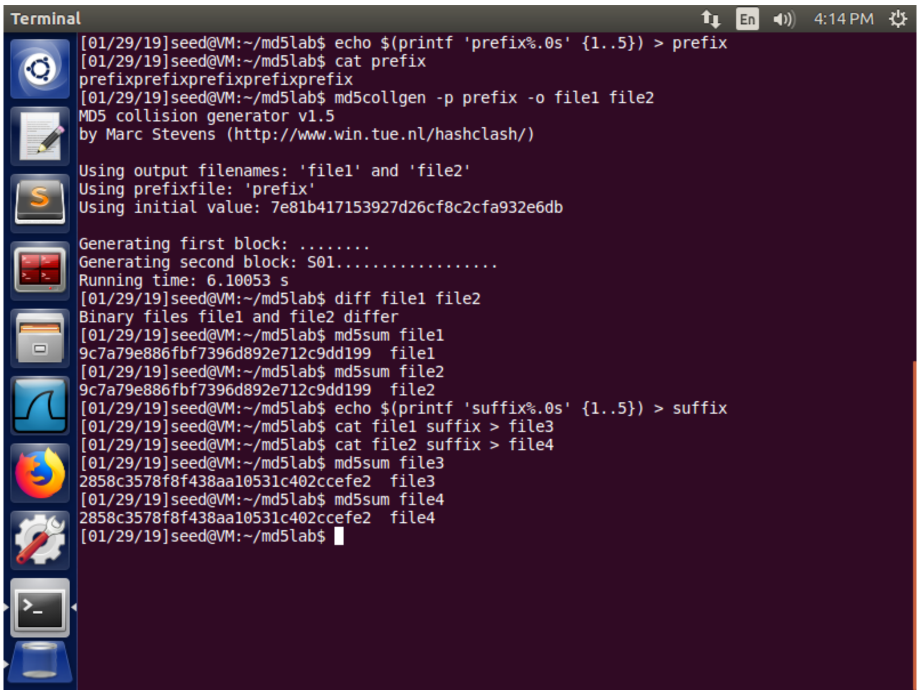
```
ome/seed/md5lab/out2.bin - Bless

41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 0A 00 00 00 00 00 00 00 AAAAAAAAAA........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00 00 46 6D 63 4F DC 83 87 3E 62 11 D0 48 1B C3 A7 D4 ..FmcO...>b..H....
54 6F 0E DE FC 57 7F F4 3F E2 39 06 6C 18 B9 65 21 FC To...W..?.9.l..e!.
FB E4 D8 8C 14 15 67 BC 5D 1A 1B EE 6C A6 75 F4 2C 9A ......g.]...l.u.,.
14 7A BC 95 16 8C 88 C5 EF 17 B6 39 01 36 39 44 D3 AA .z.........9.69D..
1B F3 8D BE 64 AC 8B D5 D3 0B A0 74 C6 C3 F8 B9 E0 61 ....d......t.....a
C9 FB F7 C0 59 59 79 22 47 91 57 B6 1B DA 91 B3 7B 71 ....YYy"G.W.....{q
C4 CF A0 AA 6F AA 87 58 F6 79 44 7C AC BE 47 A1 45 2B ....o..X.yD|..G.E+
6D BA 79 FD                                           m.y.

8 bit: 65          Signed 32 bit: 1094795585          Hexadecimal: 41 41 41 41
8 bit: 65          Unsigned 32 bit: 1094795585        Decimal: 065 065 065 065
6 bit: 16705       Float 32 bit: 12.07843             Octal: 101 101 101 101
6 bit: 16705       Float 64 bit: 2261634.50980392     Binary: 01000001 01000001 01
le endian decoding   ☐ Show unsigned as hexadecimal   ASCII Text: AAAA
Offset: 0x0 / 0xff          Selection: None          INS
```

As shown in `bless`, different bytes are located 93, AD, BB, F7, F8, FB.

### Task 2: Understanding MD5's Property



```
[01/29/19]seed@VM:~/md5lab$ echo $(printf 'prefix%.0s' {1..5}) > prefix
[01/29/19]seed@VM:~/md5lab$ cat prefix
prefixprefixprefixprefixprefix
[01/29/19]seed@VM:~/md5lab$ md5collgen -p prefix -o file1 file2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'file1' and 'file2'
Using prefixfile: 'prefix'
Using initial value: 7e81b417153927d26cf8c2cfa932e6db

Generating first block: ........
Generating second block: S01.................
Running time: 6.10053 s
[01/29/19]seed@VM:~/md5lab$ diff file1 file2
Binary files file1 and file2 differ
[01/29/19]seed@VM:~/md5lab$ md5sum file1
9c7a79e886fbf7396d892e712c9dd199  file1
[01/29/19]seed@VM:~/md5lab$ md5sum file2
9c7a79e886fbf7396d892e712c9dd199  file2
[01/29/19]seed@VM:~/md5lab$ echo $(printf 'suffix%.0s' {1..5}) > suffix
[01/29/19]seed@VM:~/md5lab$ cat file1 suffix > file3
[01/29/19]seed@VM:~/md5lab$ cat file2 suffix > file4
[01/29/19]seed@VM:~/md5lab$ md5sum file3
2858c3578f8f438aa10531c402ccefe2  file3
[01/29/19]seed@VM:~/md5lab$ md5sum file4
2858c3578f8f438aa10531c402ccefe2  file4
[01/29/19]seed@VM:~/md5lab$ 
```

As shown in the screenshot, we generate two different files with the same MD5 hash string and concatenate with same suffix. The new files, file3 and file4, also have the same MD5 hash string.

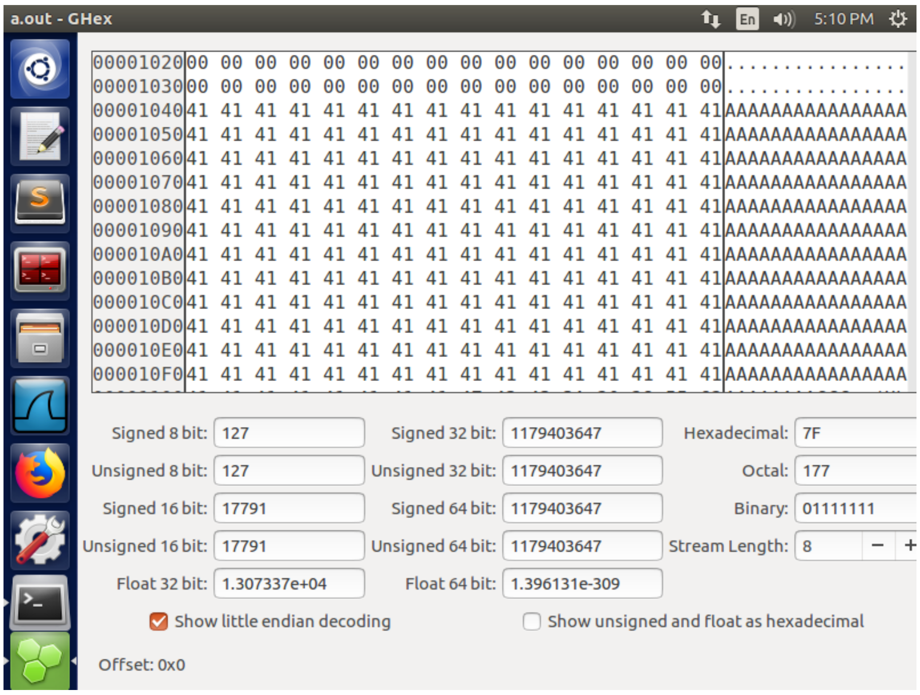
### Task 3 : Generating two executable files with the same MD5 hash

Sample code from guide:

```c
#include <stdio.h>

unsigned char xyz[200] = {
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    ... (omitted) ...
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
};

int main()
{
    int i;
    for (i=0; i<200; i++){
        printf("%x", xyz[i]);
    }
    printf("\n");
}
```



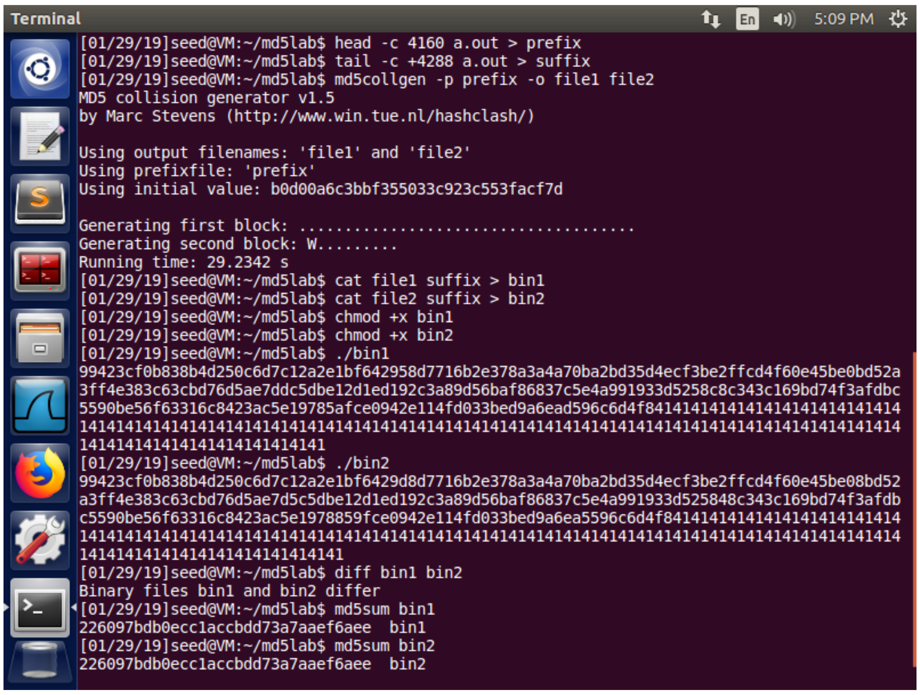
a.out - GHex

```
0000102000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00................
0000103000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00................
0000104041 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41AAAAAAAAAAAAAAAA
0000105041 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41AAAAAAAAAAAAAAAA
0000106041 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41AAAAAAAAAAAAAAAA
0000107041 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41AAAAAAAAAAAAAAAA
0000108041 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41AAAAAAAAAAAAAAAA
0000109041 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41AAAAAAAAAAAAAAAA
000010A041 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41AAAAAAAAAAAAAAAA
000010B041 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41AAAAAAAAAAAAAAAA
000010C041 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41AAAAAAAAAAAAAAAA
000010D041 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41AAAAAAAAAAAAAAAA
000010E041 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41AAAAAAAAAAAAAAAA
000010F041 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41AAAAAAAAAAAAAAAA
```

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

[✓] Show little endian decoding  [ ] Show unsigned and float as hexadecimal

Offset: 0x0



```
[01/29/19]seed@VM:~/md5lab$ head -c 4160 a.out > prefix
[01/29/19]seed@VM:~/md5lab$ tail -c +4288 a.out > suffix
[01/29/19]seed@VM:~/md5lab$ md5collgen -p prefix -o file1 file2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'file1' and 'file2'
Using prefixfile: 'prefix'
Using initial value: b0d00a6c3bbf355033c923c553facf7d

Generating first block: ..................................
Generating second block: W.........
Running time: 29.2342 s
[01/29/19]seed@VM:~/md5lab$ cat file1 suffix > bin1
[01/29/19]seed@VM:~/md5lab$ cat file2 suffix > bin2
[01/29/19]seed@VM:~/md5lab$ chmod +x bin1
[01/29/19]seed@VM:~/md5lab$ chmod +x bin2
[01/29/19]seed@VM:~/md5lab$ ./bin1
99423cf0b838b4d250c6d7c12a2e1bf642958d7716b2e378a3a4a70ba2bd35d4ecf3be2ffcd4f60e45be0bd52a
3ff4e383c63cbd76d5ae7ddc5dbe12d1ed192c3a89d56baf86837c5e4a991933d5258c8c343c169bd74f3afdbc
5590be56f63316c8423ac5e19785afce0942e114fd033bed9a6ead596c6d4f8414141414141414141414
141414141414141414141414141414141414141414141414141414141414141414141414141414141414
1414141414141414141414141
[01/29/19]seed@VM:~/md5lab$ ./bin2
99423cf0b838b4d250c6d7c12a2e1bf6429d8d7716b2e378a3a4a70ba2bd35d4ecf3be2ffcd4f60e45be08bd52
a3ff4e383c63cbd76d5ae7d5c5dbe12d1ed192c3a89d56baf86837c5e4a991933d525848c343c169bd74f3afdb
c5590be56f63316c8423ac5e1978859fce0942e114fd033bed9a6ea5596c6d4f84141414141414141414
141414141414141414141414141414141414141414141414141414141414141414141414141414141414
141414141414141414141414141
[01/29/19]seed@VM:~/md5lab$ diff bin1 bin2
Binary files bin1 and bin2 differ
[01/29/19]seed@VM:~/md5lab$ md5sum bin1
226097bdb0ecc1accbdd73a7aaef6aee  bin1
[01/29/19]seed@VM:~/md5lab$ md5sum bin2
226097bdb0ecc1accbdd73a7aaef6aee  bin2
```

From the first screenshot, we notice that the array `xyz` is stored begin with offset 1040 (4160 in decimal) in the binary file. Thus we cut the first 4160 bytes as prefix and use `md5collgen` to generate two files with different extra 128 bytes. Concatenated with the suffix file, two binary files are now executable and have the same hash string. Note that printed string still have 72 `0x41`s at the end since they are in the suffix file.

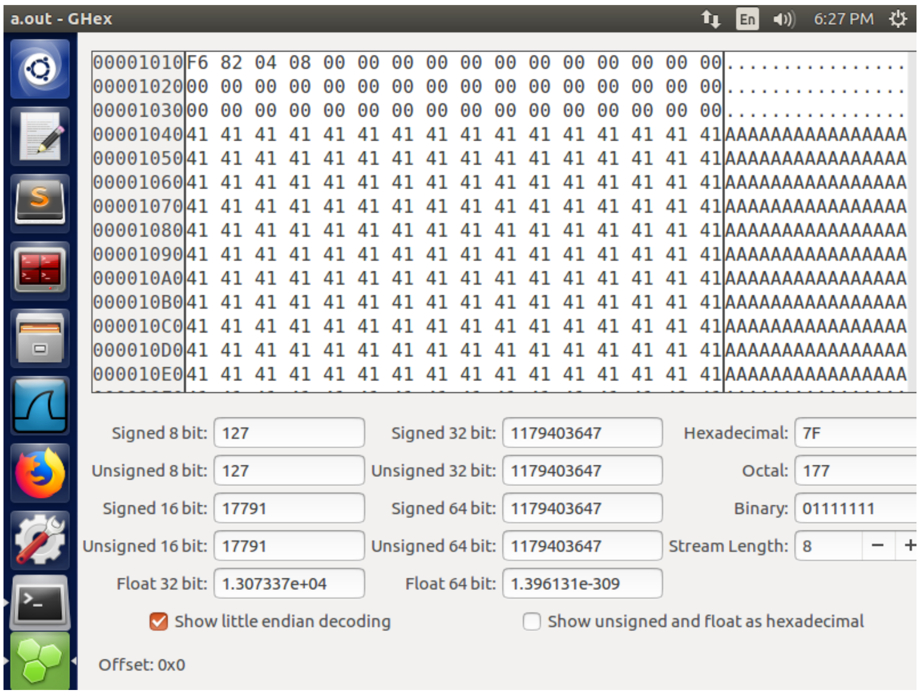
### Task 4 : Making two programs behave differently

```c
#include<stdio.h>

unsigned char a[200] = {
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    ... (omitted) ...
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
};
unsigned char b[200] = {
    0x42, 0x41, 0x41, 0x41, 0x41, 0x41,
    ... (omitted) ...
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
};

int main()
{
    int flag = 1;
    for(int i=0;i<200;i++)
    {
        if(a[i] != b[i])
        {
            flag = 0;
```

```
                break;
            }
        }
        if(flag)
            printf("run benign code\n");
        else
            printf("run malicious code\n");
        return 0;
}
```



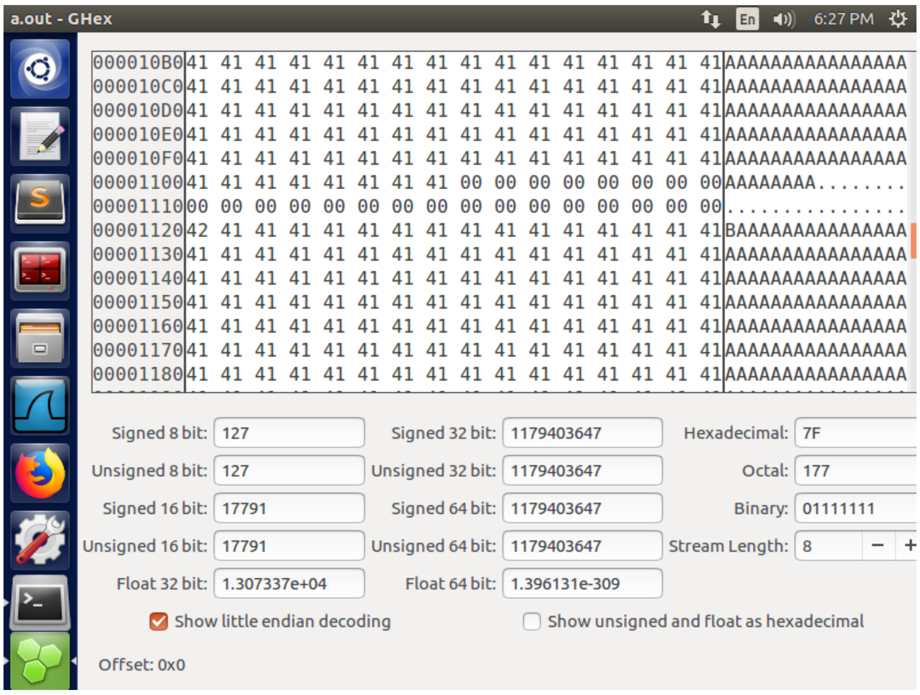
a.out - GHex

```
00001010 F6 82 04 08 00 00 00 00 00 00 00 00 00 00 00 00 ................
00001020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00001030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00001040 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00001050 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00001060 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00001070 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00001080 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00001090 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
000010A0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
000010B0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
000010C0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
000010D0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
000010E0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
```

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

[✓] Show little endian decoding  [ ] Show unsigned and float as hexadecimal

Offset: 0x0



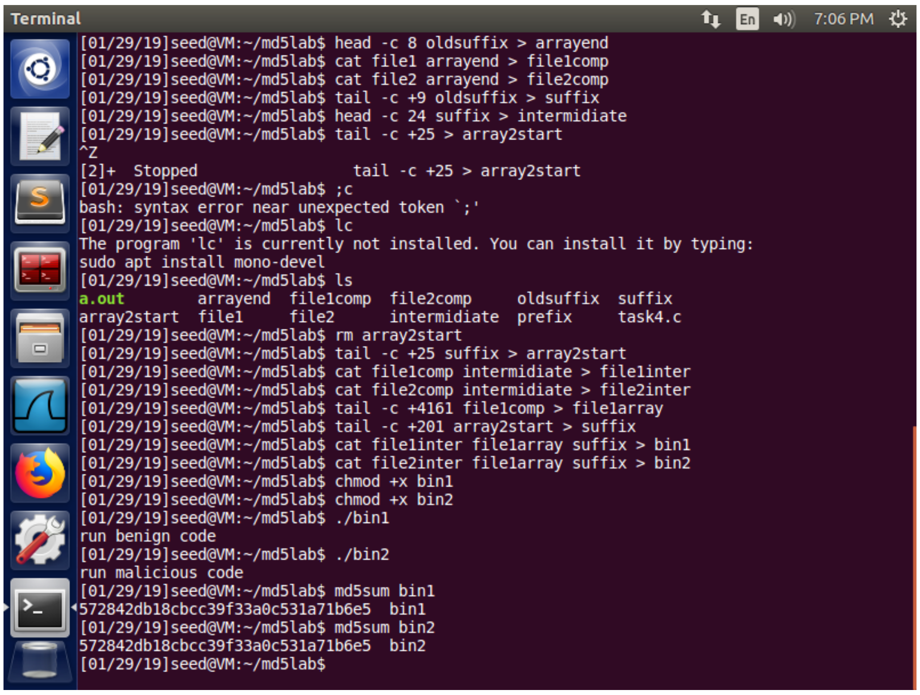
a.out - GHex

```
000010B0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
000010C0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
000010D0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
000010E0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
000010F0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00001100 41 41 41 41 41 41 41 41 00 00 00 00 00 00 00 00 AAAAAAAA........
00001110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00001120 42 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 BAAAAAAAAAAAAAAA
00001130 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00001140 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00001150 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00001160 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00001170 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00001180 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
```

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

[✓] Show little endian decoding  [ ] Show unsigned and float as hexadecimal

Offset: 0x0



```
[01/29/19]seed@VM:~/md5lab$ head -c 8 oldsuffix > arrayend
[01/29/19]seed@VM:~/md5lab$ cat file1 arrayend > file1comp
[01/29/19]seed@VM:~/md5lab$ cat file2 arrayend > file2comp
[01/29/19]seed@VM:~/md5lab$ tail -c +9 oldsuffix > suffix
[01/29/19]seed@VM:~/md5lab$ head -c 24 suffix > intermidiate
[01/29/19]seed@VM:~/md5lab$ tail -c +25 > array2start
^Z
[2]+  Stopped                 tail -c +25 > array2start
[01/29/19]seed@VM:~/md5lab$ ;c
bash: syntax error near unexpected token `;'
[01/29/19]seed@VM:~/md5lab$ lc
The program 'lc' is currently not installed. You can install it by typing:
sudo apt install mono-devel
[01/29/19]seed@VM:~/md5lab$ ls
a.out       arrayend  file1comp  file2comp     oldsuffix  suffix
array2start  file1     file2      intermidiate  prefix     task4.c
[01/29/19]seed@VM:~/md5lab$ rm array2start
[01/29/19]seed@VM:~/md5lab$ tail -c +25 suffix > array2start
[01/29/19]seed@VM:~/md5lab$ cat file1comp intermidiate > file1inter
[01/29/19]seed@VM:~/md5lab$ cat file2comp intermidiate > file2inter
[01/29/19]seed@VM:~/md5lab$ tail -c +4161 file1comp > file1array
[01/29/19]seed@VM:~/md5lab$ tail -c +201 array2start > suffix
[01/29/19]seed@VM:~/md5lab$ cat file1inter file1array suffix > bin1
[01/29/19]seed@VM:~/md5lab$ cat file2inter file1array suffix > bin2
[01/29/19]seed@VM:~/md5lab$ chmod +x bin1
[01/29/19]seed@VM:~/md5lab$ chmod +x bin2
[01/29/19]seed@VM:~/md5lab$ ./bin1
run benign code
[01/29/19]seed@VM:~/md5lab$ ./bin2
run malicious code
[01/29/19]seed@VM:~/md5lab$ md5sum bin1
572842db18cbcc39f33a0c531a71b6e5  bin1
[01/29/19]seed@VM:~/md5lab$ md5sum bin2
572842db18cbcc39f33a0c531a71b6e5  bin2
[01/29/19]seed@VM:~/md5lab$
```

From the first screenshot, we notice that the array `a` start at offset 1040. So we first cut the file until the first 64 bytes of array `a` and generate 128 bytes using `md5collgen` command.

```
$ head -c 4224 a.out > prefix
$ md5collgen -p prefix -o file1 file2
```

Now we need to fill the rest part, we first complete array `a`:

```
$ tail -c +4353 a.out > oldsuffix
$ head -c 8 oldsuffix > arrayend
$ cat file1 arrayend > file1comp
$ cat file2 arrayend > file2comp
```

Then we fill the gap between array `a` and `b`:

```
$ tail -c +9 oldsuffix > suffix
$ head -c 24 suffix > intermidiate
$ tail -c +25 suffix > array2start
$ cat file1comp intermidiate > file1inter
$ cat file2comp intermidiate > file2inter
```

Then we replace array `b` with the array in file1 and wrap up:

```
$ tail -c +4161 file1comp > file1array
$ tail -c +201 array2start > suffix
$ cat file1inter file1array suffix > bin1
$ cat file2inter file1array suffix > bin2
```

As shown in the last screenshot, `bin1` and `bin2` have the same value, but they execute different branches.