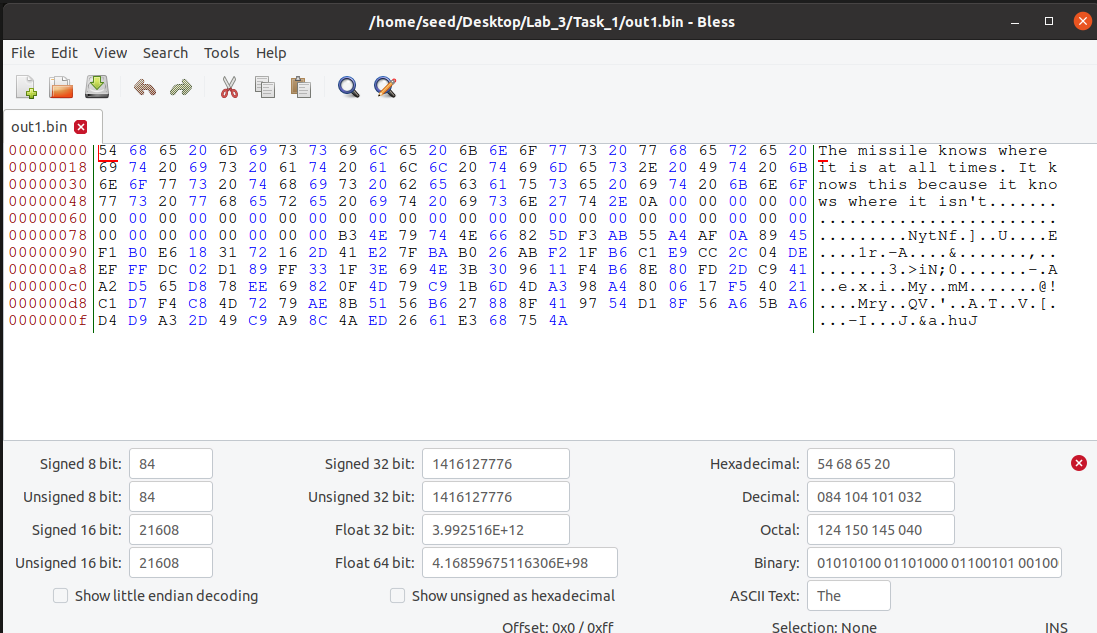
**Lab 3**

**Task 1**

1. If the length of prefix file is not multiple of 64 bytes, there will be ‘0’ padding for the binary file.



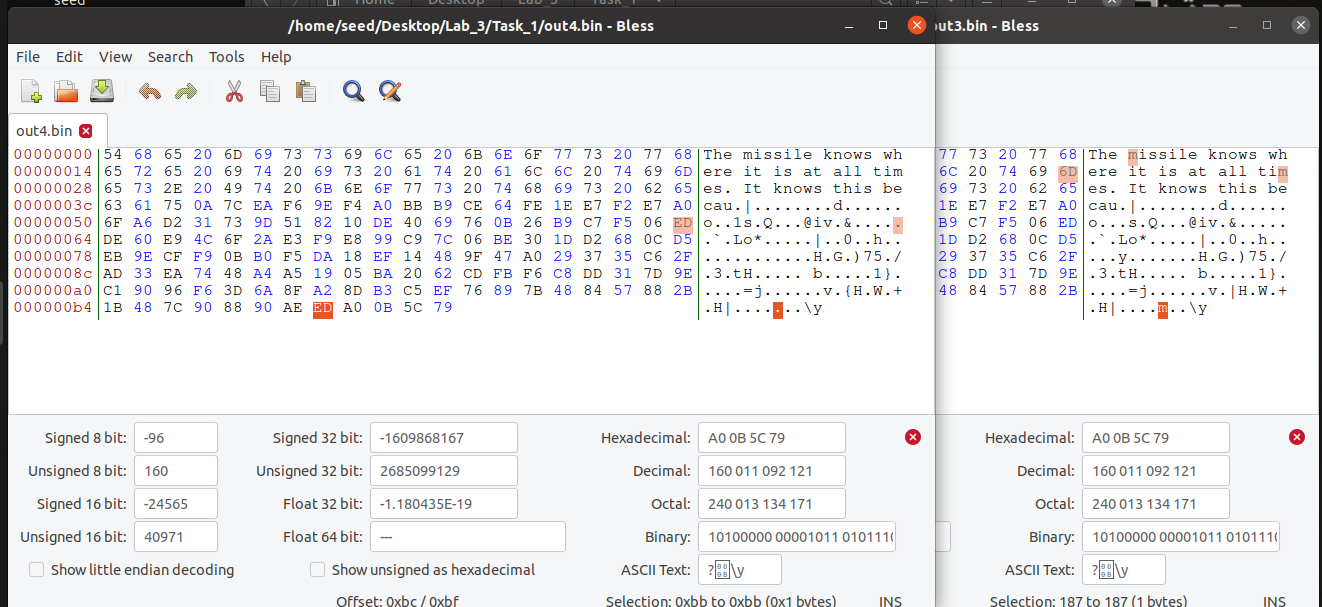
1. If the prefix file is exactly 64 bytes, there will be no padding for the binary file.

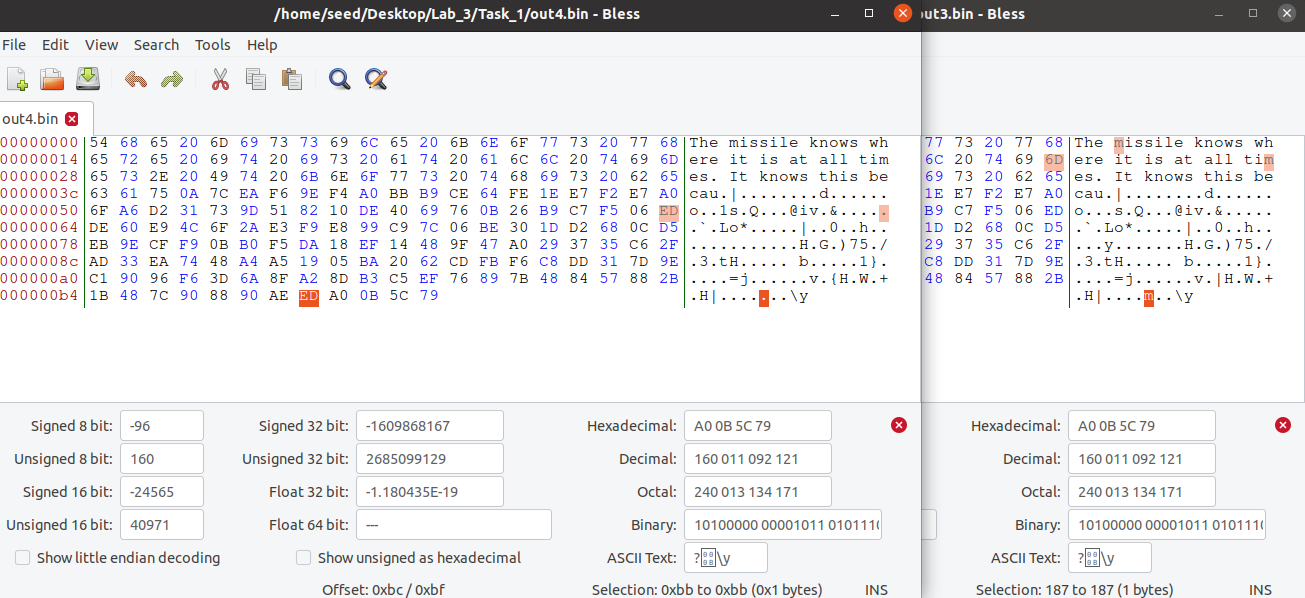
Graphical user interface, text

Description automatically generated

1. Data generated by md5collgen only have slightly differences. Byte 60 ,112 ,and 125 have difference. Graphical user interface, text

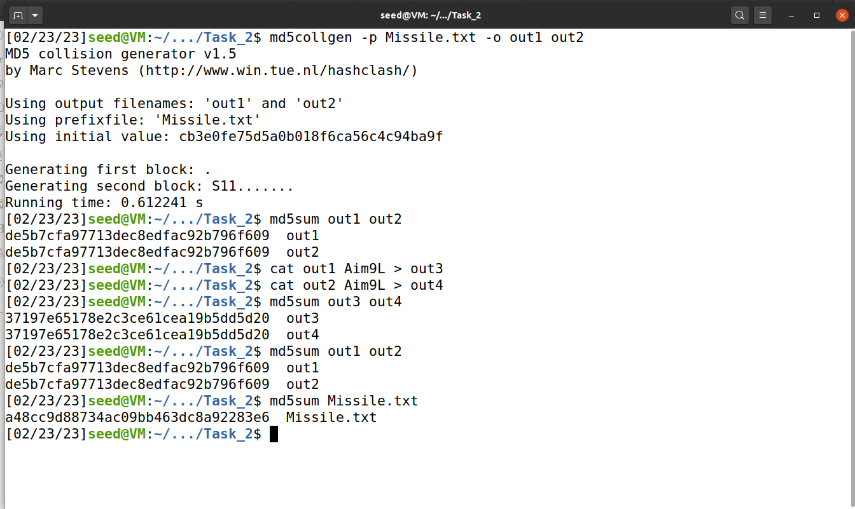
   Description automatically generated with medium confidence

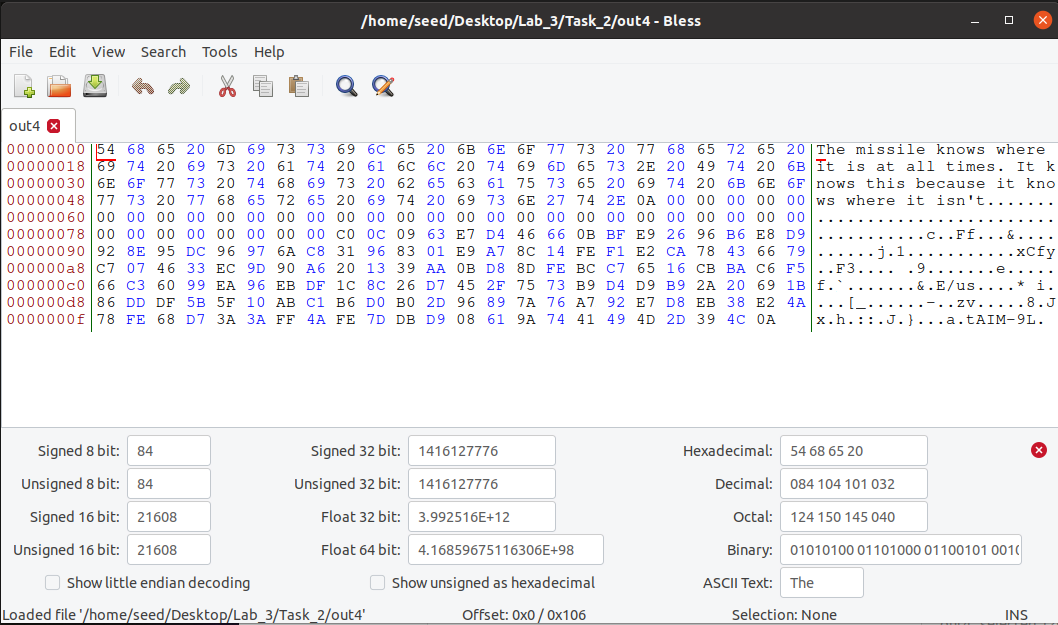
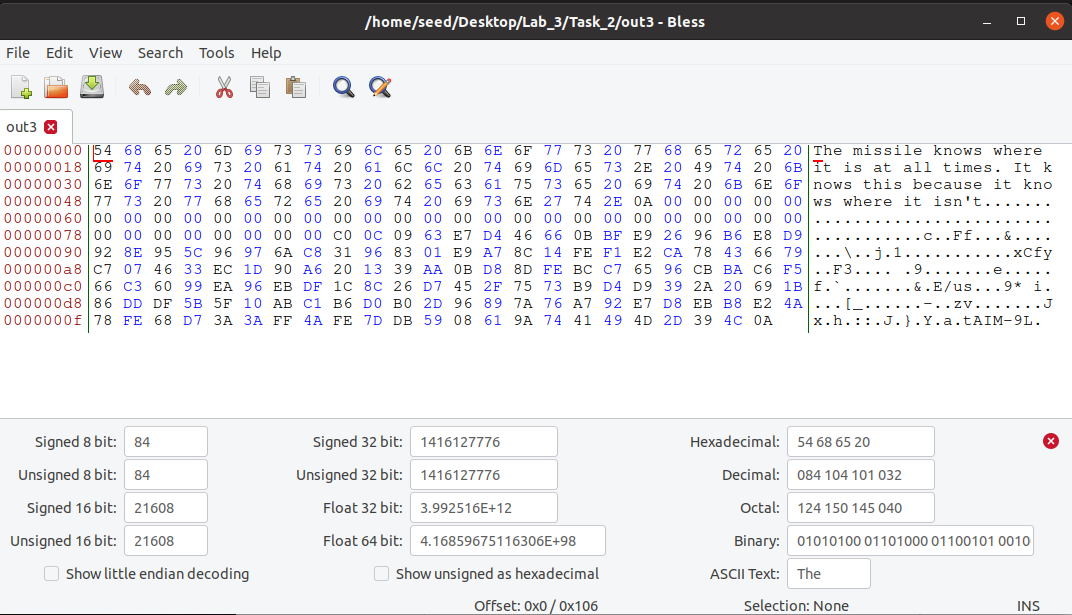




**Task 2**

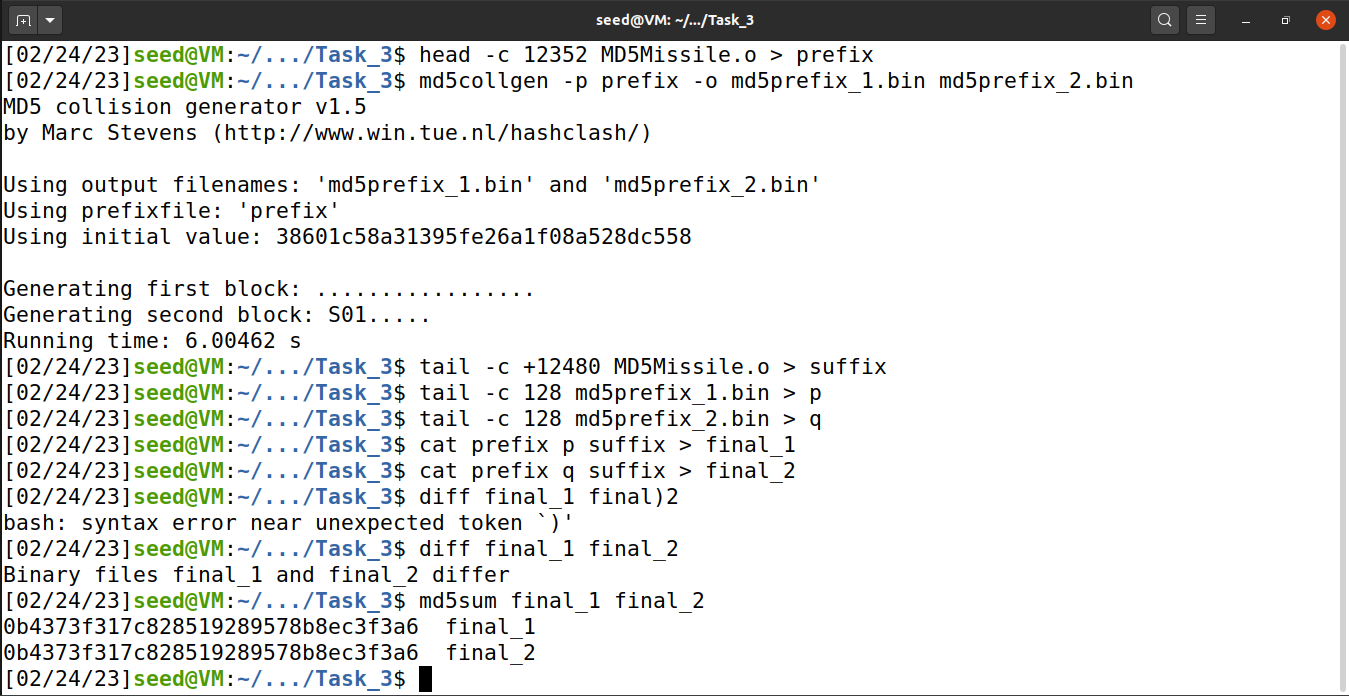
First, I create 2 text files then md5collgen them. Run **md5sum** Then I append new text to those md5 binary files. Run **md5sum** to both files again. It appears those md5 hash remains same.



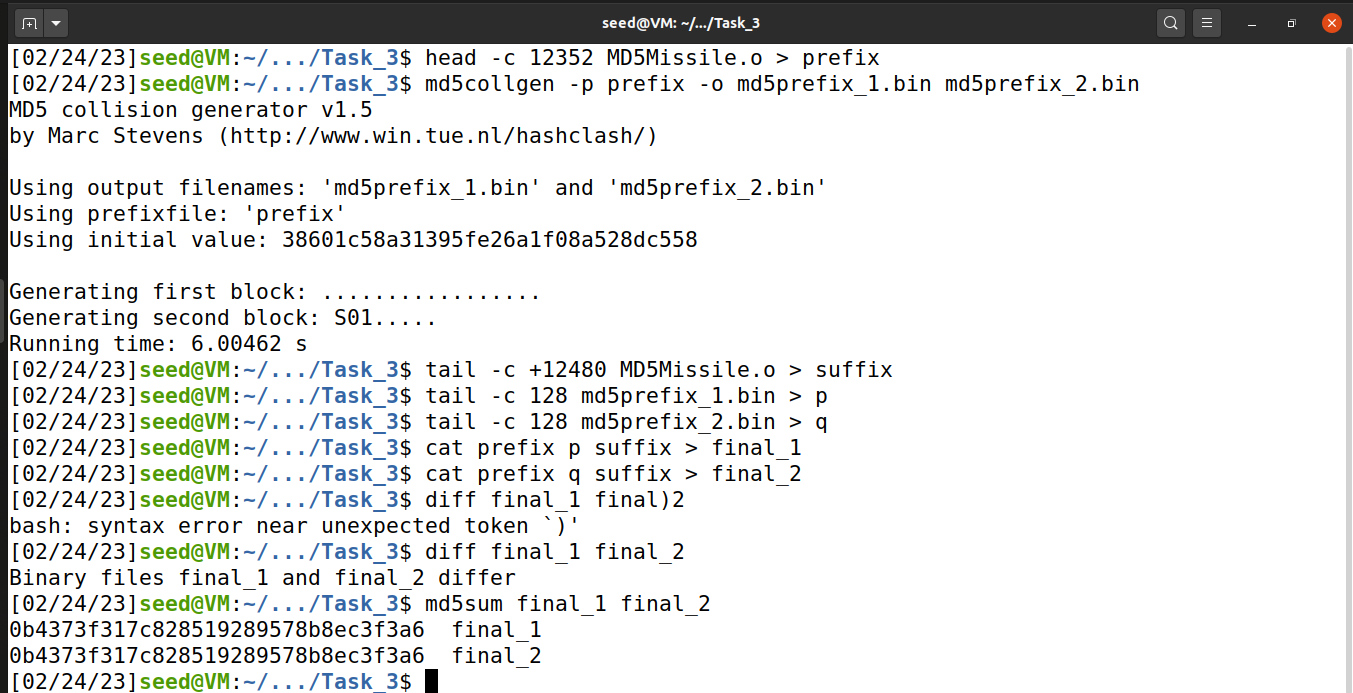
.

**Task 3**

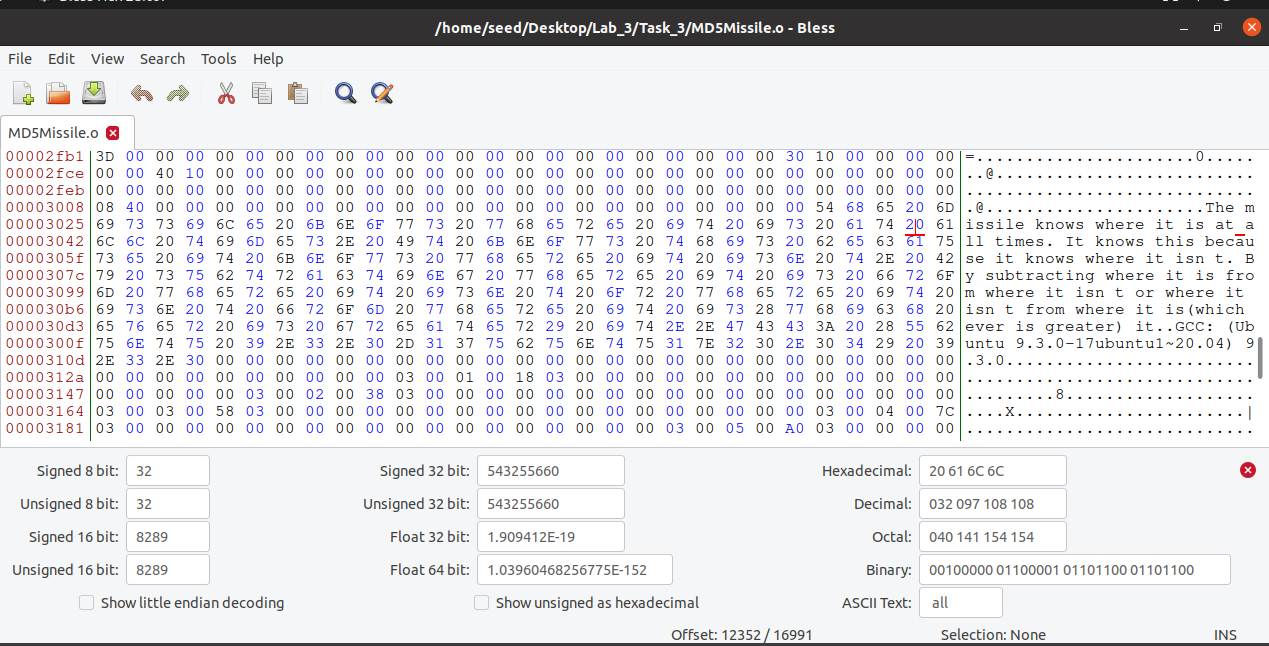
I used C program to create a 200 bytes string array. Then count 12352 bytes as multiple of 64. In addition, that truncates a part of array to the suffix. Then, cut 128 bytes after prefix as p and q; after 128 bytes, the rest of the file is suffix. Concatenate prefix, p/q(as 2 version of final output), and suffix.

****

After concatenation, both final version still shares same md5 hash despite the 128 bytes part is completely different from the original ones.

****

The red line indicates the end of prefix and start of 128 byte p/q.



End of 128 byte p/q.

A screenshot of a computer

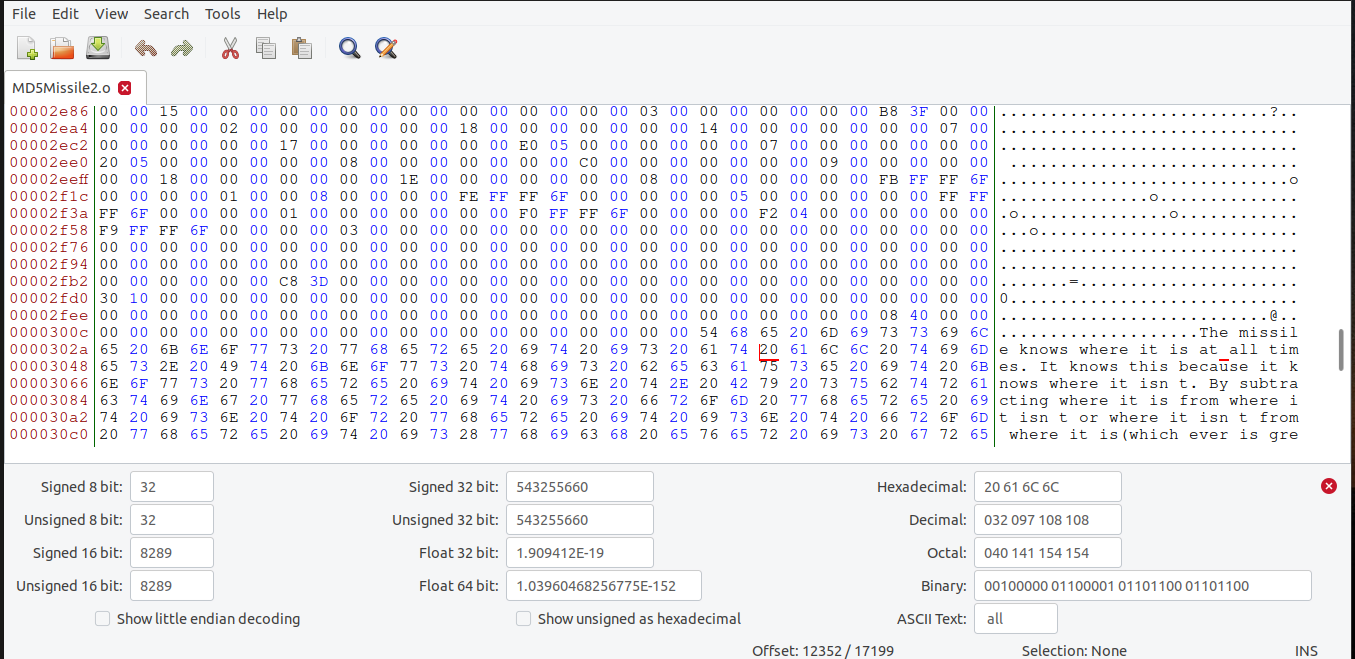
Description automatically generated with medium confidence

**Task 4**

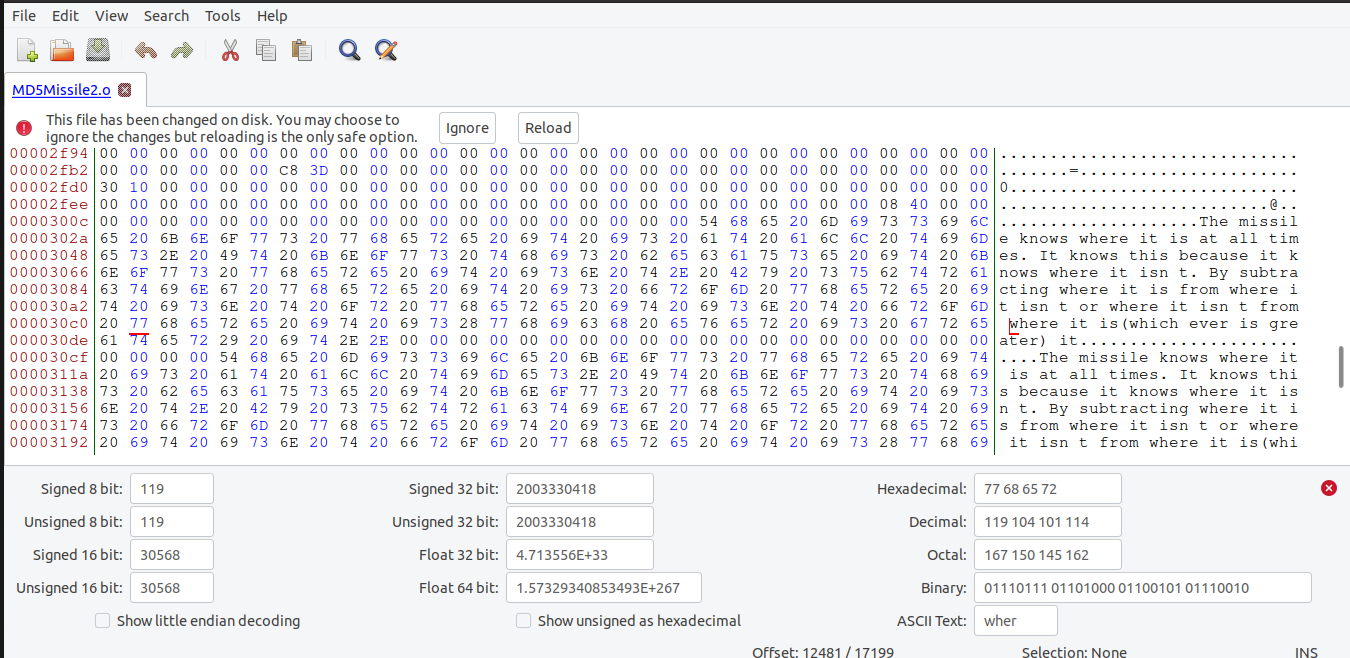
I create 2 arrays in C program. Just like previous one, count 12352 bytes then cut bytes before that byte as prefix. Then count 128 byte as p/q in X-array. After the 128 byte, starts the suffix part. However, 1st 96 bytes of the suffix is the 1st segment of suffix. Therefore, After 96 bytes of the suffix, the 128 bytes after that is Y-array. At this point, 128 + 96 bytes(1st segment of suffix + 128 byte Y-array part) leads to the starting point of rest of segment of suffix. Concatenate **prefix + p/q + 1st segment of suffix + p + rest of suffix** will lead to final\_1 and final\_2 program. Final\_1 has **both p’s**, and Final\_2 has **p and q**. Run them, it will be shown that even though the p/q part has changed, if rest of the binary files are not tampered, the outcome is still same. If the array are different, then the terminal will shot ‘Run Malicious Code’ instead of ‘run benign code’.

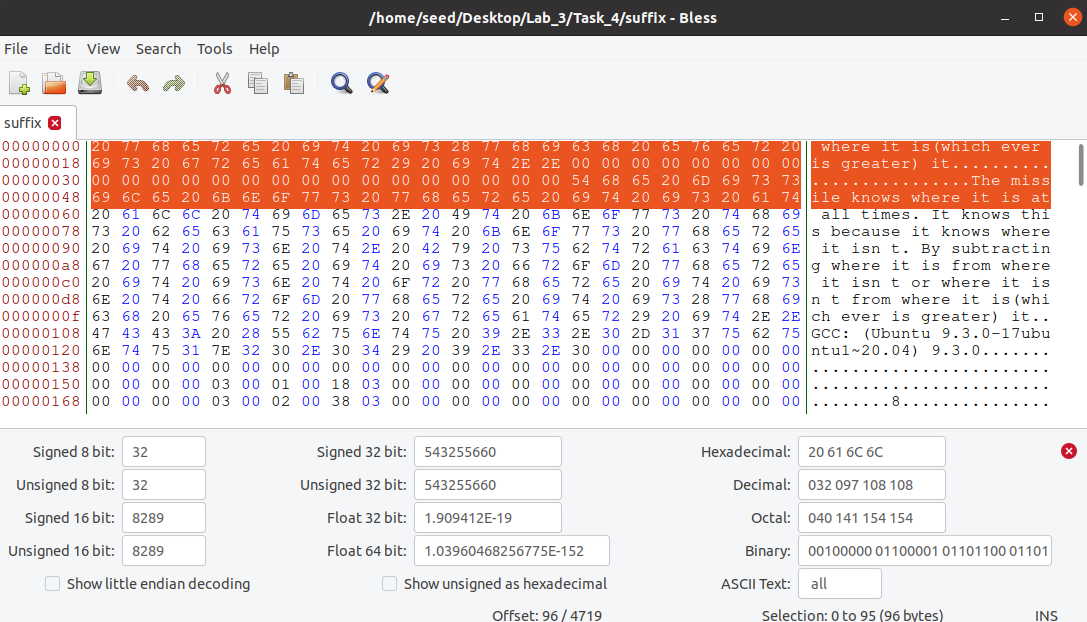


End of prefix, start of 1st p/q 128 byte part. 12352 byte point.

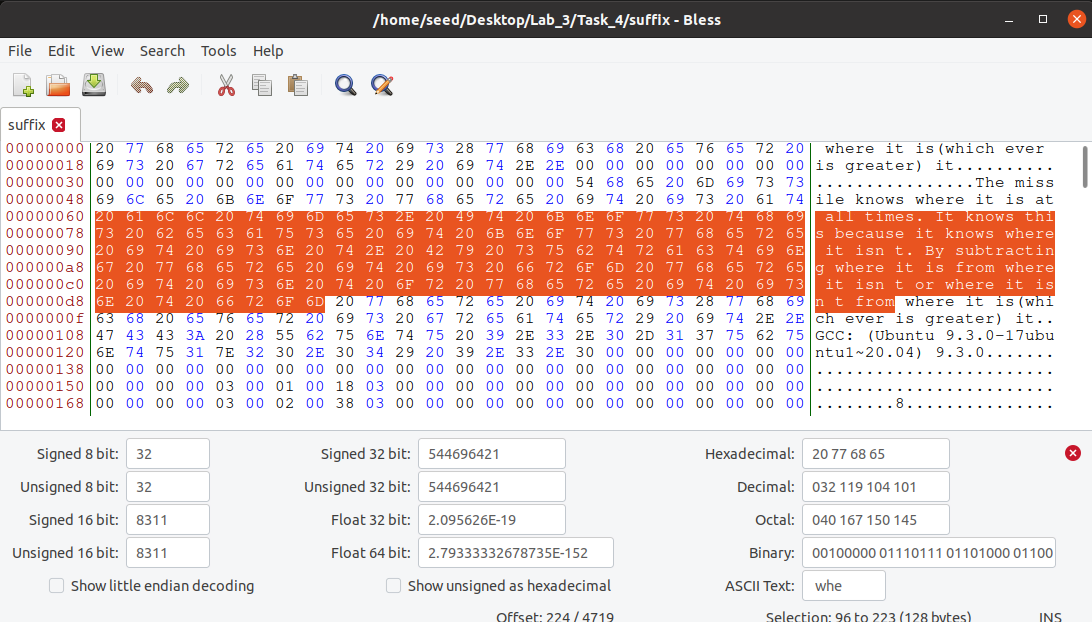


End of 1st p/q 128 byte part. Start of suffix. 12481 byte point.



First segment of suffix. 96 bytes.

2nd p 128 byte part. 224 byte point. After that, the remains are final segments of suffix.



After cutting pieces of those files, concatenate prefix + p/q + 1st segment of suffix + p + rest of suffix will lead to final\_1 and final\_2 program. Final\_1 has both p’s, and Final\_2 has p and q. Run them, it will be shown that even though the p/q part has changed, as long as rest of the binary files are not tampered, the outcome is still same. (Run benign code).