Assignment On File Pointer

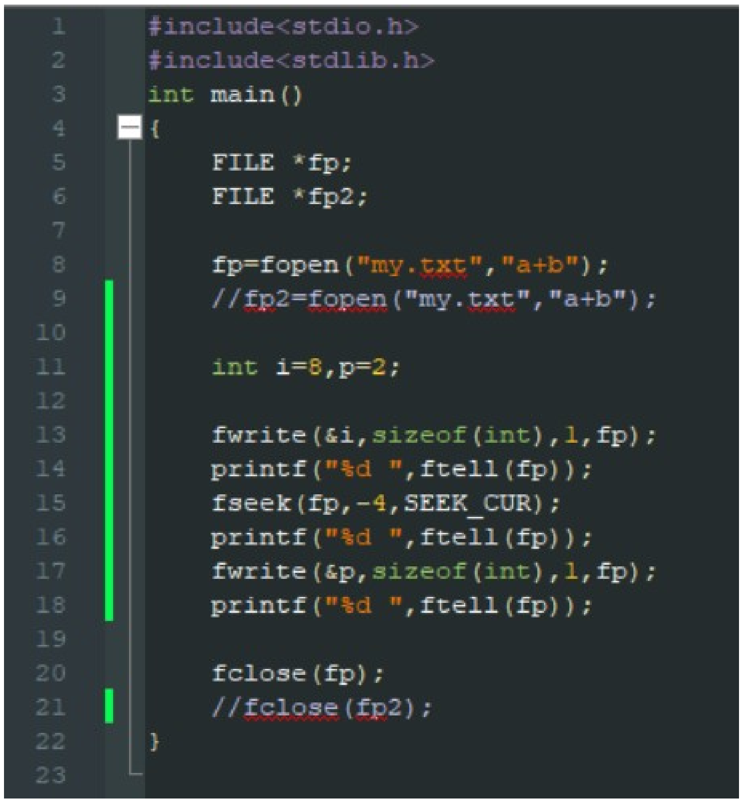
Group 11
File opening mode “a+b”

20/20

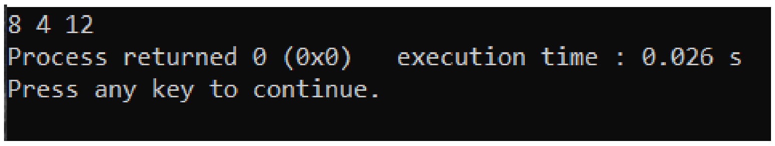
Student Roll Numbers:
1805071
1805083
1805095
1805107
1805119

0. fseek in a+b mode

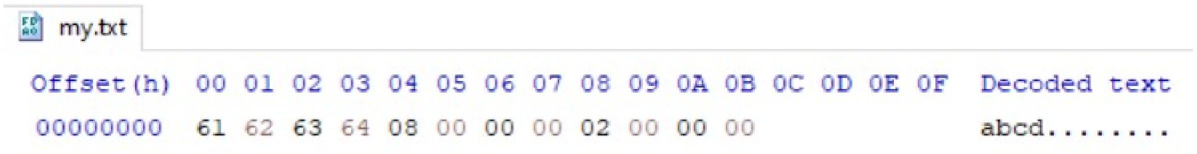
For the beginning, we did **fseek** to see how append mode works.
At first, we moved it ahead.



```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    FILE *fp;
    FILE *fp2;

    fp=fopen("my.txt","a+b");
    //fp2=fopen("my.txt","a+b");

    int i=8,p=2;

    fwrite(&i,sizeof(int),1,fp);
    printf("%d ",ftell(fp));
    fseek(fp,-4,SEEK_CUR);
    printf("%d ",ftell(fp));
    fwrite(&p,sizeof(int),1,fp);
    printf("%d ",ftell(fp));

    fclose(fp);
    //fclose(fp2);
}
```

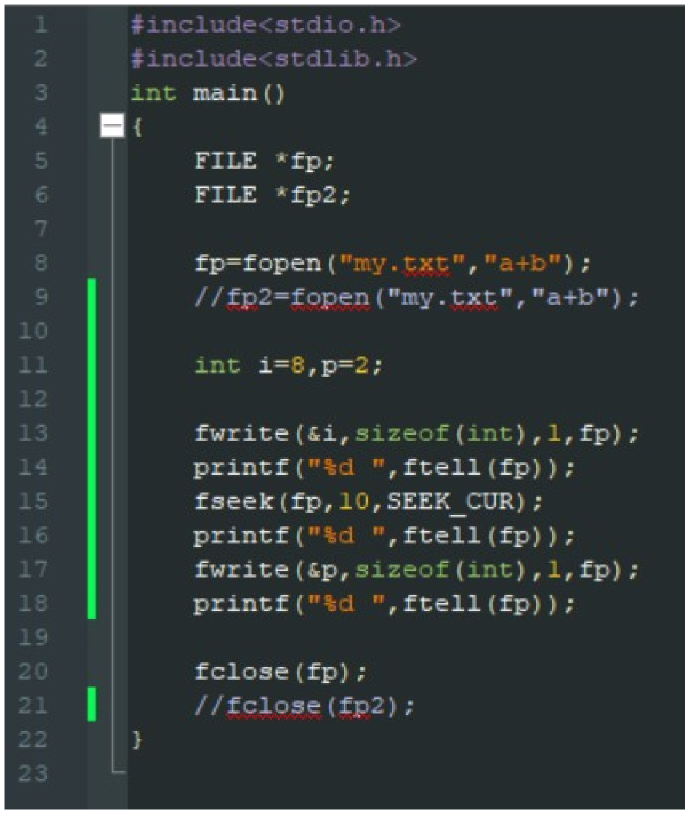


```
8 4 12
Process returned 0 (0x0)   execution time : 0.026 s
Press any key to continue.
```

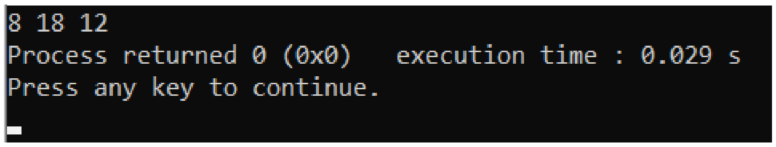


```
my.txt
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  61 62 63 64 08 00 00 00 02 00 00 00              abcd........
```

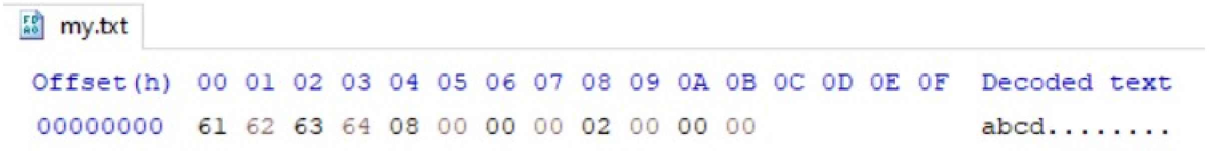
Then we moved the pointer outside the scope of the file.



```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    FILE *fp;
    FILE *fp2;

    fp=fopen("my.txt","a+b");
    //fp2=fopen("my.txt","a+b");

    int i=8,p=2;

    fwrite(&i,sizeof(int),1,fp);
    printf("%d ",ftell(fp));
    fseek(fp,10,SEEK_CUR);
    printf("%d ",ftell(fp));
    fwrite(&p,sizeof(int),1,fp);
    printf("%d ",ftell(fp));

    fclose(fp);
    //fclose(fp2);
}
```



```
8 18 12
Process returned 0 (0x0)   execution time : 0.029 s
Press any key to continue.
```

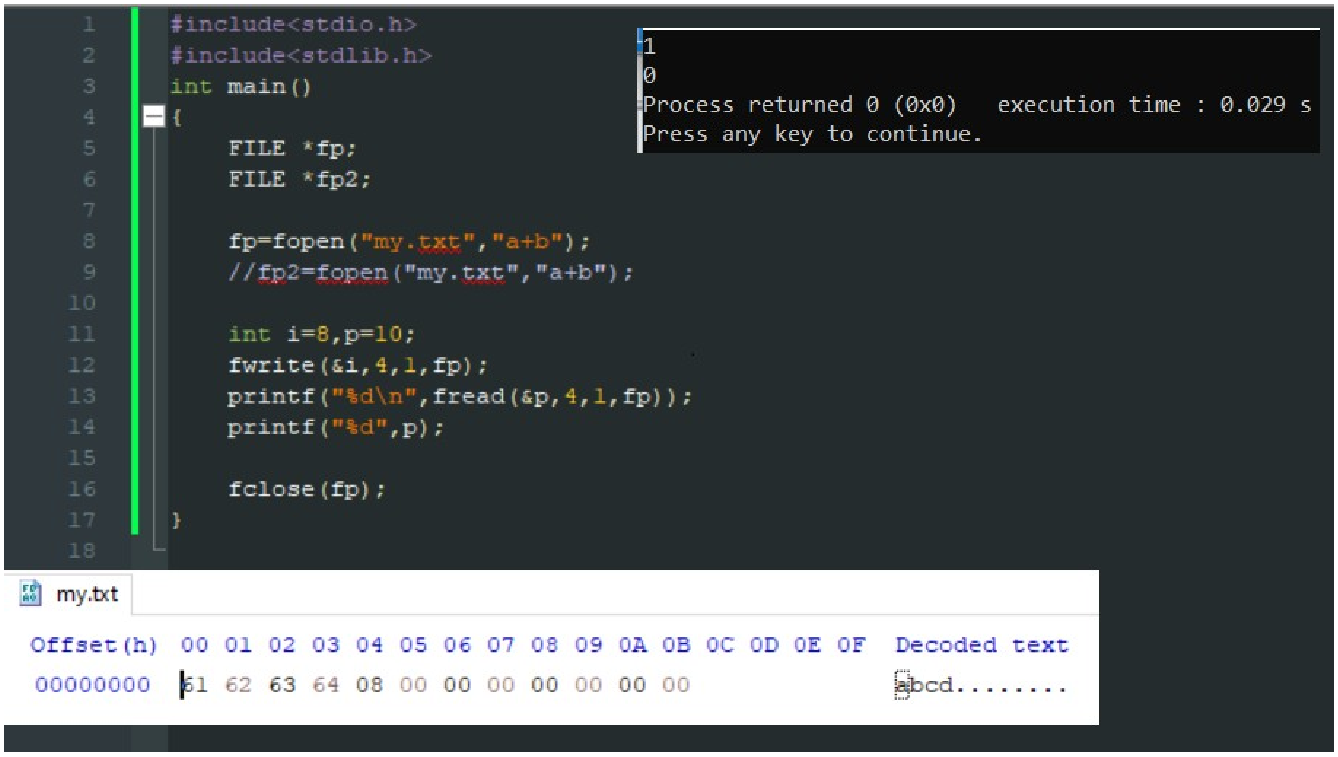


```
my.txt

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  61 62 63 64 08 00 00 00 02 00 00 00              abcd........
```

We found out that append writes in the end. fseek may move the file pointer but always moves to the end of the text file when called to write.

1. fread and fwrite

We first tried to do fread just after writing a number(8) in the file after “abcd”. The return value of fread was 1, meaning even though the file bytes were not there some bytes were read. The value was 0 (assigned to p). Interestingly, the text file started showing 4 NULL bytes in the end. It seems that fread just increased the file size.



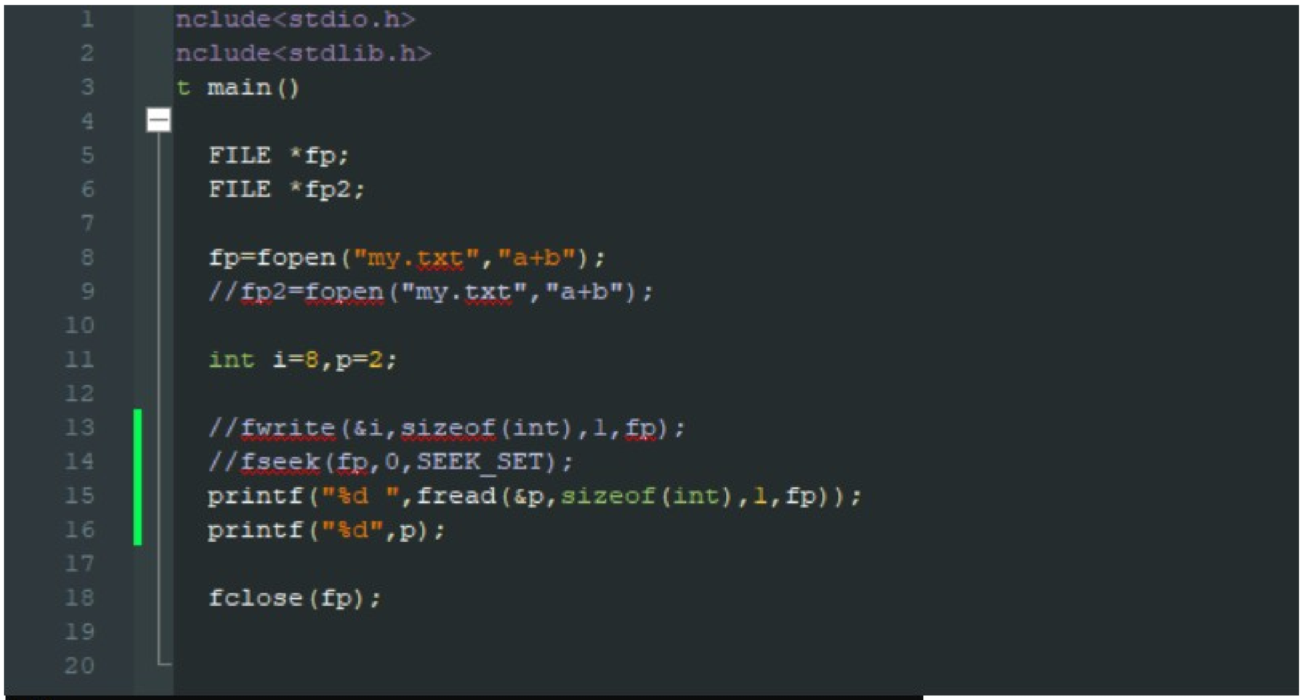
```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    FILE *fp;
    FILE *fp2;

    fp=fopen("my.txt","a+b");
    //fp2=fopen("my.txt","a+b");

    int i=8,p=10;
    fwrite(&i,4,1,fp);
    printf("%d\n",fread(&p,4,1,fp));
    printf("%d",p);

    fclose(fp);
}
```

```
1
0
Process returned 0 (0x0)   execution time : 0.029 s
Press any key to continue.
```

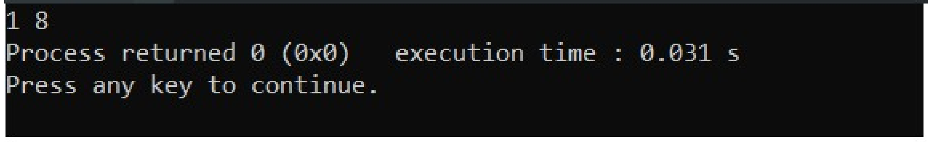
my.txt

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  61 62 63 64 08 00 00 00 00 00 00 00              abcd........
```

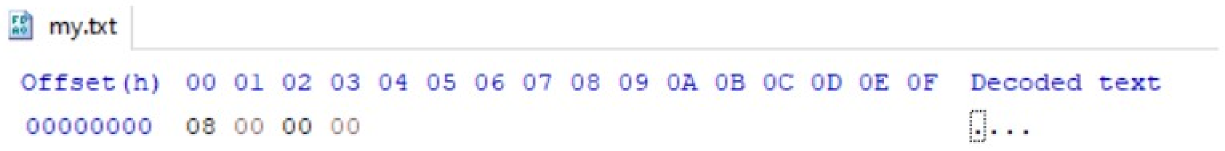
Then, we tried to do just fread on a file which only contained 3 NULL bytes and a byte with value 8.



```
nclude<stdio.h>
nclude<stdlib.h>
t main()

    FILE *fp;
    FILE *fp2;

    fp=fopen("my.txt","a+b");
    //fp2=fopen("my.txt","a+b");

    int i=8,p=2;

    //fwrite(&i,sizeof(int),1,fp);
    //fseek(fp,0,SEEK_SET);
    printf("%d ",fread(&p,sizeof(int),1,fp));
    printf("%d",p);

    fclose(fp);
```



```
1 8
Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```



```
my.txt
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  08 00 00 00                                      ....
```

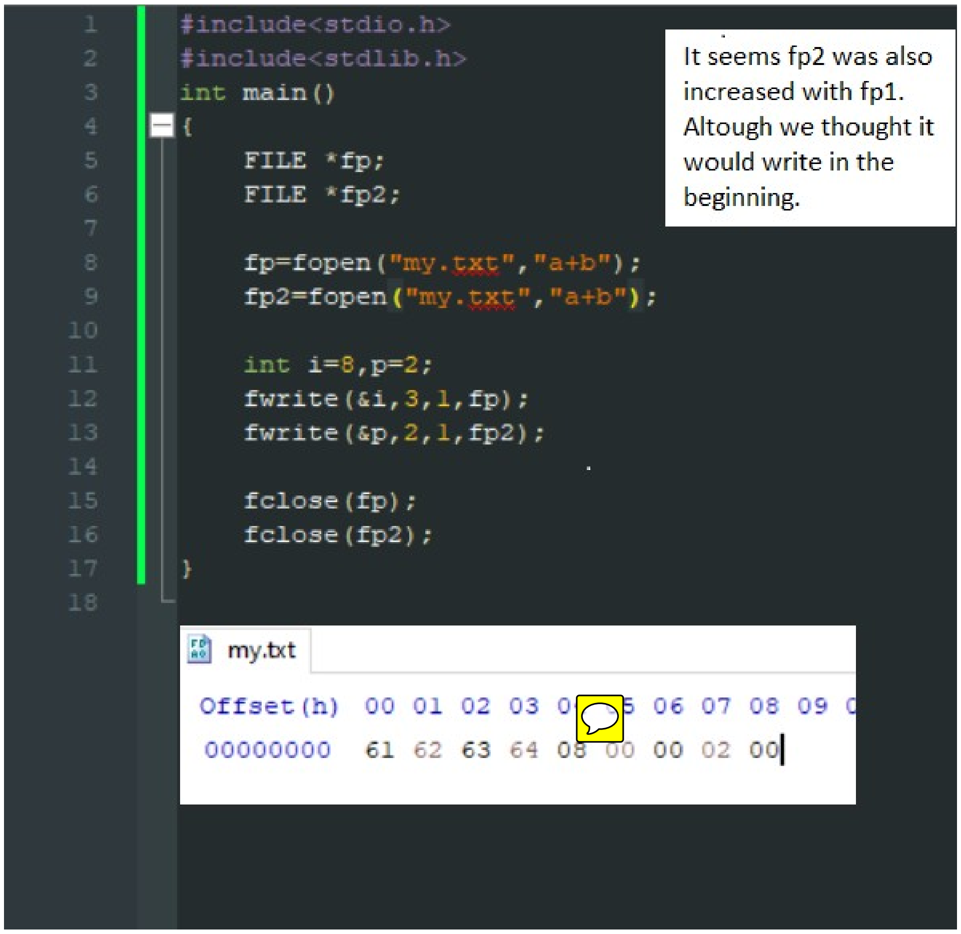
We found that in this case no 0 bytes were written. It reads the first 4 bytes. But we didn’t do fseek to bring fp to the beginning.

It seems that fp (even in a+) starts from the beginning. This is how it can read from the file. Then as we write something the pointer moves to the end.

To read the data we should always fseek to the required field. FILE doesn’t hold two pointers for fread and fwrite separately.

2. Two file pointers

Opened the same file twice using two file pointers. The file contained “abcd”. First fwrite wrote 8. Then using the fp2 pointer and writing 2 doesn’t replace the 2 bytes rather writes them at the end.



```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    FILE *fp;
    FILE *fp2;

    fp=fopen("my.txt","a+b");
    fp2=fopen("my.txt","a+b");

    int i=8,p=2;
    fwrite(&i,3,1,fp);
    fwrite(&p,2,1,fp2);

    fclose(fp);
    fclose(fp2);
}
```

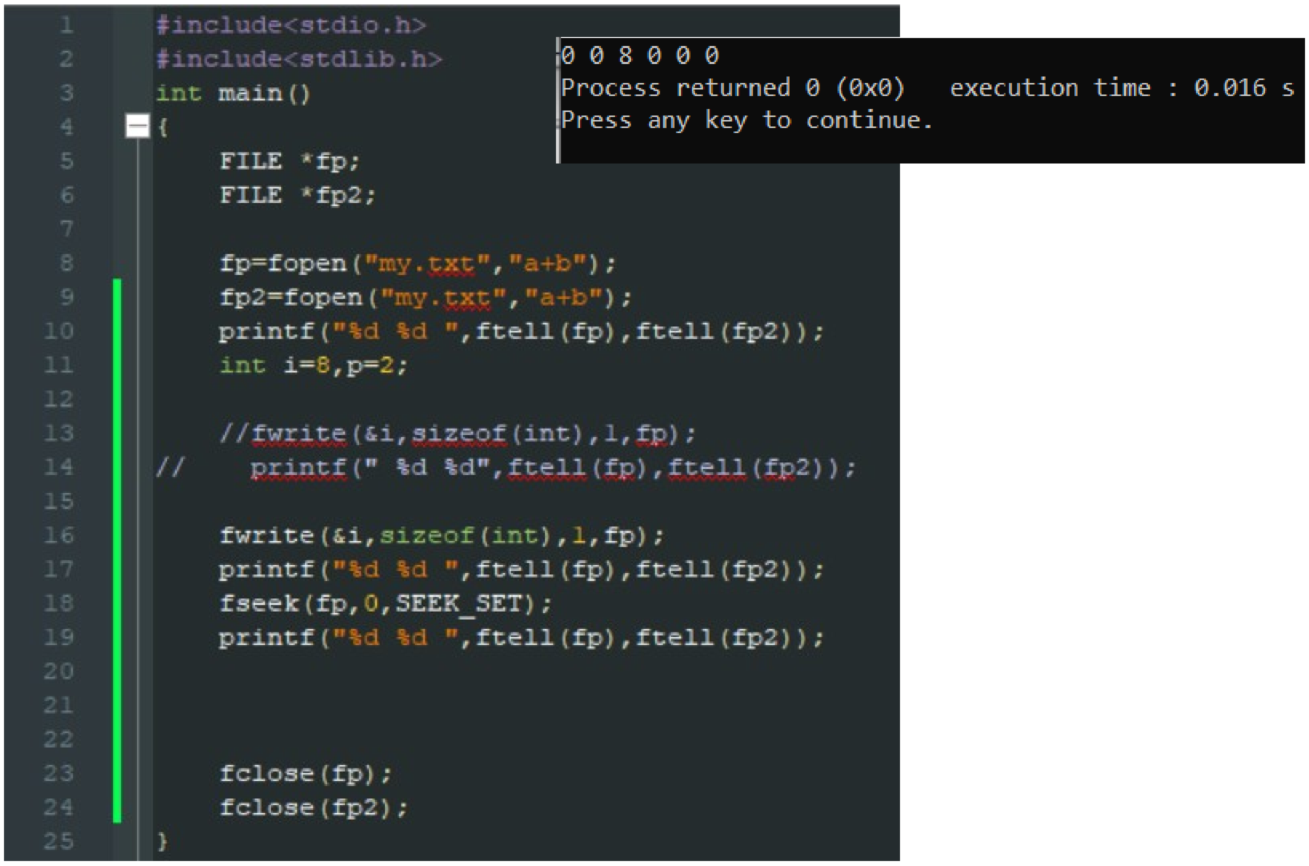
It seems fp2 was also increased with fp1. Altough we thought it would write in the beginning.

my.txt

```
Offset(h)  00 01 02 03 0  5 06 07 08 09 0
00000000   61 62 63 64 08 00 00 02 00
```

It seems that fp2 was increased with fp1. Also it might be possible that fp2 was in the beginning at first (as conjectured in 1) and then moved to the end position as it was called to write.

To check which one is correct we used ftell in the next program.



```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    FILE *fp;
    FILE *fp2;

    fp=fopen("my.txt","a+b");
    fp2=fopen("my.txt","a+b");
    printf("%d %d ",ftell(fp),ftell(fp2));
    int i=8,p=2;

    //fwrite(&i,sizeof(int),1,fp);
//    printf(" %d %d",ftell(fp),ftell(fp2));

    fwrite(&i,sizeof(int),1,fp);
    printf("%d %d ",ftell(fp),ftell(fp2));
    fseek(fp,0,SEEK_SET);
    printf("%d %d ",ftell(fp),ftell(fp2));



    fclose(fp);
    fclose(fp2);
}
```

```
0 0 8 0 0 0
Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

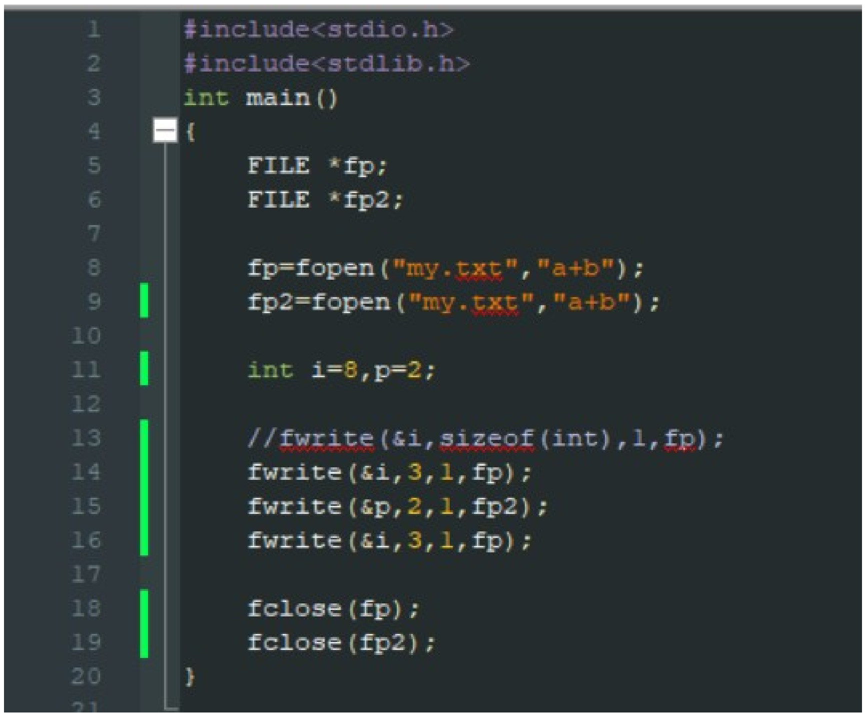
Both fp and fp2 begins at the beginning. Then when calling fwrite fp moves to 4. Then writes 4 bytes and ftell returns 8. fp2 remains at the beginning.

From this test, we can conclude that opening in ‘a+b’ mode the pointer points at the beginning. But calling fwrite makes it to go to the end.

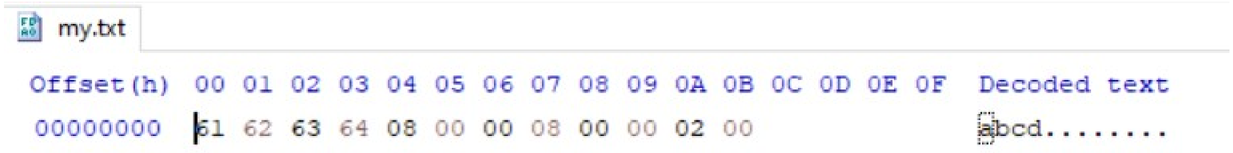
Though this thought can hardly explain the next program.

3. fwrite and two file pointers

(From 2) We used fp to write at the end. But somewhat weirdly fp doesn't overwrite fp2's data nor does it write after f2's written data.
Rather it inserts(!) the data.

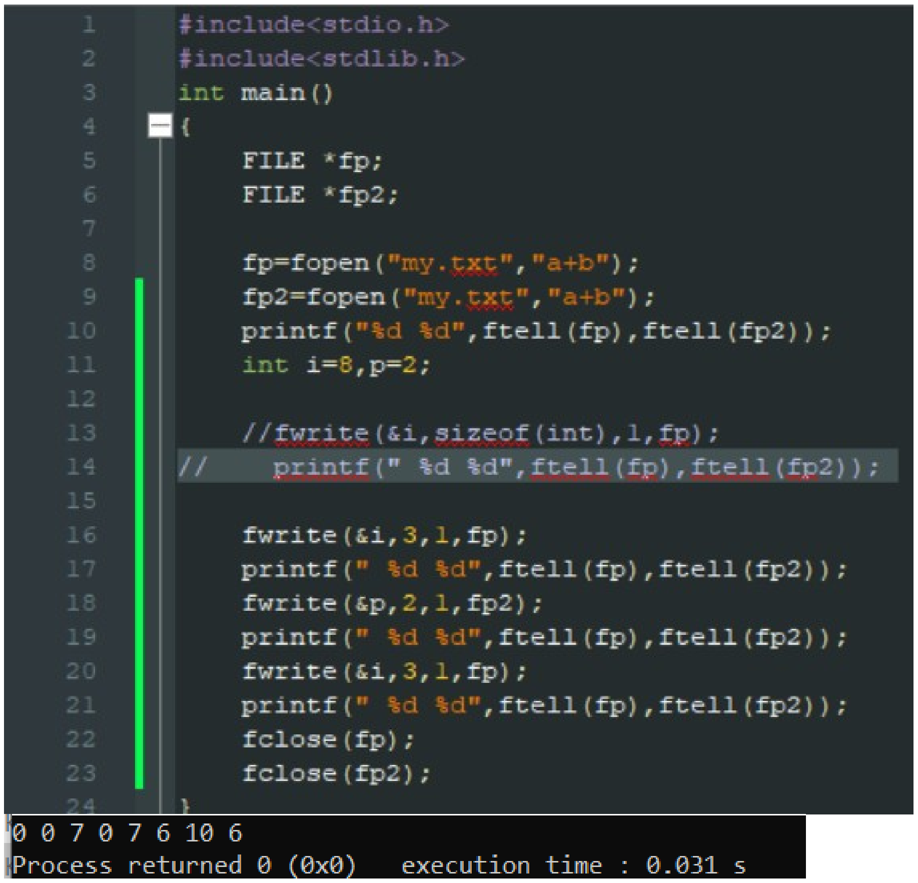


```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    FILE *fp;
    FILE *fp2;

    fp=fopen("my.txt","a+b");
    fp2=fopen("my.txt","a+b");

    int i=8,p=2;

    //fwrite(&i,sizeof(int),1,fp);
    fwrite(&i,3,1,fp);
    fwrite(&p,2,1,fp2);
    fwrite(&i,3,1,fp);

    fclose(fp);
    fclose(fp2);
}
```



```
my.txt

Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   61 62 63 64 08 00 00 08 00 00 02 00              abcd........
```

To check what's actually happening in the file we used ftell function and got output like this in the console.



```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    FILE *fp;
    FILE *fp2;

    fp=fopen("my.txt","a+b");
    fp2=fopen("my.txt","a+b");
    printf("%d %d",ftell(fp),ftell(fp2));
    int i=8,p=2;

    //fwrite(&i,sizeof(int),1,fp);
//    printf(" %d %d",ftell(fp),ftell(fp2));

    fwrite(&i,3,1,fp);
    printf(" %d %d",ftell(fp),ftell(fp2));
    fwrite(&p,2,1,fp2);
    printf(" %d %d",ftell(fp),ftell(fp2));
    fwrite(&i,3,1,fp);
    printf(" %d %d",ftell(fp),ftell(fp2));
    fclose(fp);
    fclose(fp2);
}
```

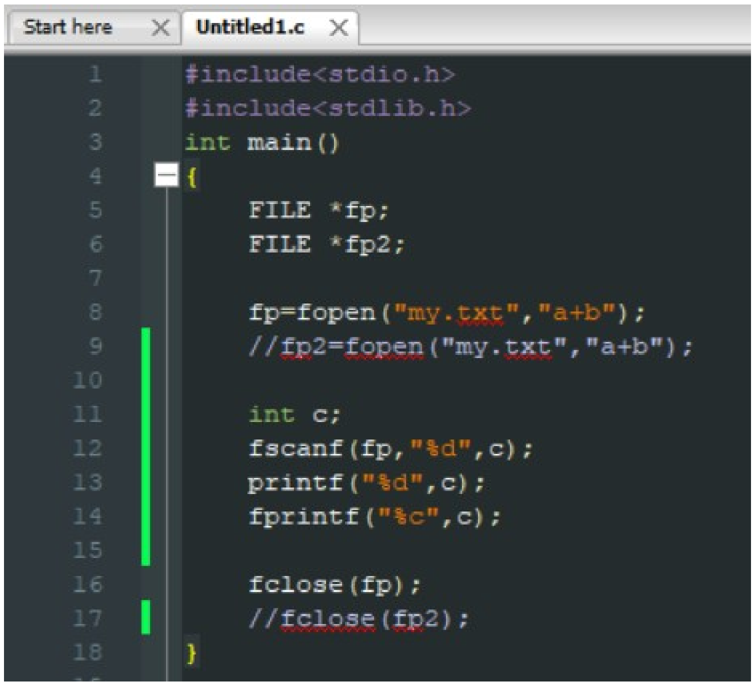
```
0 0 7 0 7 6 10 6
Process returned 0 (0x0)   execution time : 0.031 s
```

fp2 points at 6 even in the end. But writes data in the 10th position somehow.

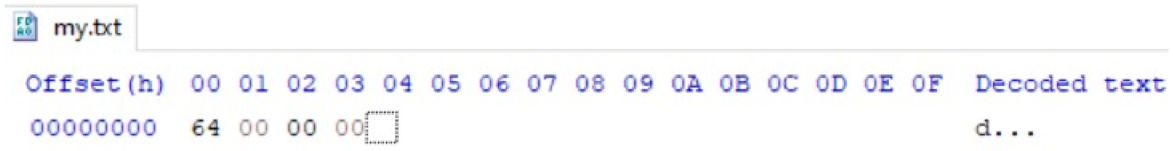
We couldn't find how this might work.

4. Using other function

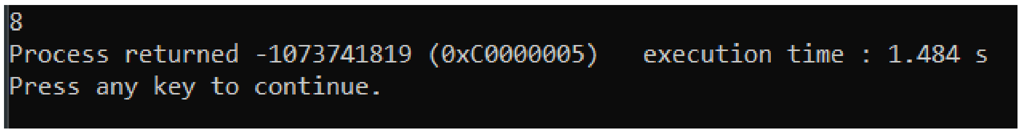
Next, we checked if **fscanf, fprintf, fgetc, fputc** functions work properly.



```
Start here    Untitled1.c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    FILE *fp;
    FILE *fp2;

    fp=fopen("my.txt","a+b");
    //fp2=fopen("my.txt","a+b");

    int c;
    fscanf(fp,"%d",c);
    printf("%d",c);
    fprintf("%c",c);

    fclose(fp);
    //fclose(fp2);
}
```



```
my.txt
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   64 00 00 00                                      d...
```

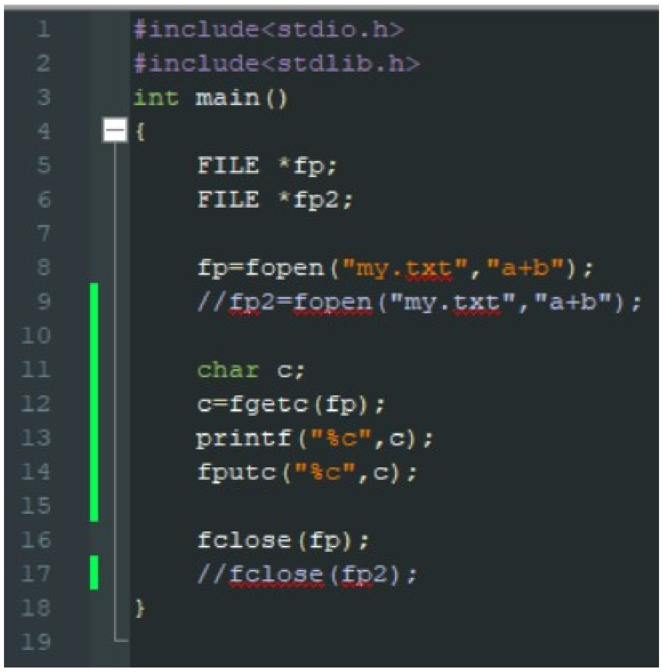


```
8
Process returned -1073741819 (0xC0000005)   execution time : 1.484 s
Press any key to continue.
```

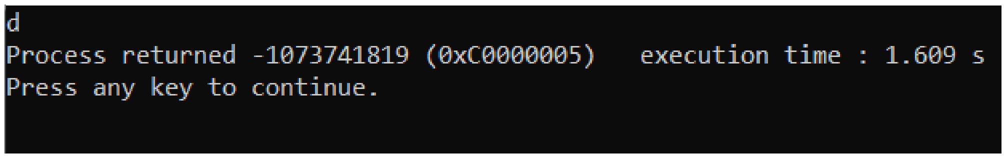
Input: As shown in the hex editor.

fscanf read the 4 bytes successfully. But fprintf couldn't write any data and the program didn't execute properly.

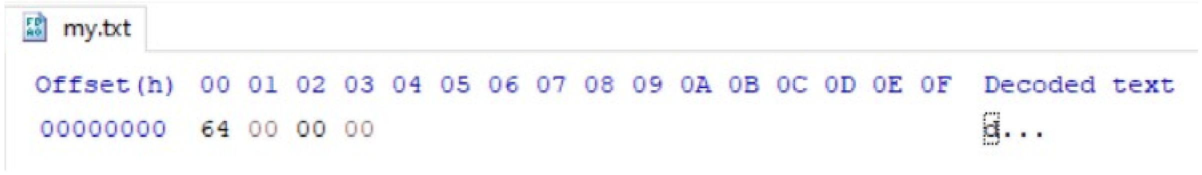
Similar result was found using fgetc and fputc.



```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    FILE *fp;
    FILE *fp2;

    fp=fopen("my.txt","a+b");
    //fp2=fopen("my.txt","a+b");

    char c;
    c=fgetc(fp);
    printf("%c",c);
    fputc("%c",c);

    fclose(fp);
    //fclose(fp2);
}

```



```
d
Process returned -1073741819 (0xC0000005)   execution time : 1.609 s
Press any key to continue.
```



```
my.txt

Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   64 00 00 00                                      d...
```

We are not certain why fgetc and fscanf was working while fprintf and fputc wasn’t.

Conclusion:

“a+b” mode opens file and the file pointer stays in the beginning. If program tries to read it reads from the start. And when tries to write the file pointer goes to the end. To read again we have to do fseek. Otherwise, fread will write null bytes and read them.

Number 3 program startled us and we are unable to explain it. Don’t know how data was being inserted.

Reading data with fgetc, fscanf works quite right. But fprintf, fputc doesn’t work and program stops.