20/20

# FILE IO IN C

# APPEND MODE TRIALS

GROUP 10


**Member's Name and Roll no.**

1. Labiba Binte Tasin 1805070
2. Anup Bhowmik 1805082
3. Sheikh Hasanul Banna 1805094
4. Sk. Sabit Bin Mosaddek 1805106
5. Arif Faisal 1805118

**1. Open in append mode and try to read from the file using:**

**a) fscanf()**

Read fails. Trying to read from the file where two integers (10, 20) is written separated by a space, it reads garbage value from the file (in my case, 8 & 52) and fscanf returns -1.

**Code:**

```
int n,m;
FILE *fp=fopen("appendtest.txt","a");
fscanf(fp,"%d %d",&n,&m);
if(ferror(fp))
        printf("Error!\n");
printf("%d %d\n", n,m);
```

**Input file:**

```
10 20
```

**Output:**

```
Error!
 8 52
```

**b) fgets()**

Read fails. Trying to read from the file where a string ("aaaa") is written, it reads garbage value from the file (in my case, "@T?v▄■`") and fgets() returns null pointer.

**Code:**

```
char s[100];
FILE *fp=fopen("appendtest.txt","a");
fgets(s,100,fp);
if(ferror(fp))
        printf("Error!\n");
printf("%s\n", s);
```

**Input file:**

```
aaaa
```

**Output:**

Error!

@T?v▄▪`

**c) fgetc()**

Read fails. But in this case, no garbage value is shown. Instead, fgetc returned -1.

**Code:**

```
char c;
FILE *fp=fopen("appendtest.txt","a");
c=fgetc(fp);
if(ferror(fp))
        printf("Error!\n");
printf("%d\n", c);
```

**Input file:**

h

**Output:**

Error!

-1

**2. Open in append mode and try to write on the file using:**

**a) fprintf()**

Writes the intended character or any other values without any space separation right after where the original file had ended. It takes the entire string as input and write the whole thing on the file (does not stop if found any whitespace).

**Code:**

```
FILE *fp=fopen("appendtest.txt","a");
fprintf(fp," world !");
if(ferror(fp))    printf("Error!\n");
```

**INITIAL FILE:**

HELLO

**Output in file:**

HELLOWORLD !

**b) fputs()**

Writes the intended string without any space separation right after where the original file had ended.

**Code:**

```
FILE *fp=fopen(“appendtest.txt”,”a”);
fputs(“\nNEW LINE\n”,fp);
if(ferror(fp))
        printf(“Error!\n”);
```

**INITIAL FILE:**

HELLOWORLD !

**Output in file:**

HELLOWORLD !

NEW LINE

**c) fputc()**

Writes the intended character without any space separation right after where the original file had ended.

**Code:**

```
FILE *fp=fopen(“appendtest.txt”,”a”);
fputc(‘A’,fp);
if(ferror(fp))
        printf(“Error!\n”);
```

**INITIAL FILE:**

HELLOWORLD !

**Output in file:**

HELLOWORLD !A

**3. Open a file in append mode that is not on the disc or current directory and try to:**

**a) Read from the file**

Creates the file in current directory and Reads garbage value.

**b) Write to the file**

Creates the file in current directory and puts the value in the file like a boss.

**4. Try using fseek() in append mode to insert some words in between some words already written on the file:**

fseek() doesn't seem to have any impact here. fseek() may change the file pointer to a certain point of the file. But the append mode tends to take the file pointer to the last position no matter what. So, this cannot insert some words in between some words.

Code:

```
FILE *fp=fopen("appendtest.txt","a");
fseek(fp, -3, SEEK_END);
fputc('A',fp);
if(ferror(fp))
        printf("Error!\n");
```

INITIAL FILE:

HELLOWORLD!

Output in file:

HELLOWORLD!A

**5. Try opening a binary file and write something using append mode:**

**a) Using fwrite()**

It successfully write in the file which was opened in append mode. Perhaps the function work irrespective of any mode.

**b) Using fread()**

It could not read from the file using fread(). Reads garbage as usual.

**6. Try opening a particular file in both read and write mode and check:**

**a) First opened for reading and then opened in append mode**

It could read from file using read mode and append as well. But whatever it is written from append mode, it could not be read using read mode. There is an exception, after writing something in append mode, if it gets flushed using fflush(), then it can be read using read mode. Append mode stores the data in buffer before writing it which brings failure to read the

data. Whenever it gets flushed, read mode can read data from file correctly. However, if a file is opened in append mode ftell() returns 0 at first, but it starts writing from the end.

**CODE:**

```
char s[100];
FILE *fp=fopen("appendtest.txt","r");
FILE *fp1=fopen("appendtest.txt","a");
fgets(s,100,fp);
printf("POSITIONS : %d %d\n",ftell(fp),ftell(fp1));
printf("%s\n",s);
if(fgets(s,100,fp))                    //reading everything from the file
        printf("%s\n",s);
else printf("END OF FILE REACHED\n");
fputs("NEW LINE\n",fp1);               //writing in append mode
if(fgets(s,100,fp))                    //trying to read it
        printf("%s\n",s);
else printf("END OF FILE REACHED\n");
printf("POSITIONS : %d %d\n",ftell(fp),ftell(fp1));
fflush(fp1);                           //flushed the buffer.
if(fgets(s,100,fp))                    //trying to read it again
        printf("%s\n",s);
else printf("END OF FILE REACHED\n");
printf("POSITIONS : %d %d\n",ftell(fp),ftell(fp1));
```

**INITIAL FILE:**

```
HELLOWORLD!
```

**OUTPUT:**

```
POSITIONS : 11 0
HELLOWORLD!
END OF FILE REACHED
END OF FILE REACHED
POSITIONS : 11 21
```

NEW LINE

POSITIONS : 21 21

**OUTPUT in file:**

HELLOWORLD!NEW LINE

**b) First opened for writing and then opened in append mode**

It could write and append correctly. But it writes everything in write mode first and then writes in append mode. If something written on append mode gets flushed, write mode will erase everything in front of it before writing.

**Code:**

```
char s[100];
FILE *fp=fopen("appendtest.txt","w");
FILE *fp1=fopen("appendtest.txt","a");
fputs("NEW LINE\n",fp1);
fputs("HELLOWORLD\n",fp);
fflush(fp1);
fputs("ANOTHER\n",fp1);
fputs("THIS SHOULD BE LAST\n",fp);
```

**Output in file:**

HELLOWORLD

THIS SHOULD BE LAST

ANOTHER

**c) First opened for reading and then opened for writing. Afterwards opened for Append**

All the previous entries in the file are removed and the new values are written from the beginning of the file. The reason is: it cleans the file first when it is opened in “w” mode. Then the append mode starts working.

**d) First opened for writing and then opened for reading. Afterwards opened for Append**

The outcome is the same.

**7. Try redirecting a file in append mode:**

**a) using freopen() to redirect a file to STDIN in append mode**

Fails to read. Scanf returns error.

**Code:**

```
freopen("appendtesst.txt","a",stdin);
char s[100];
scanf("%s",s);
if(ferror(stdin))printf("ERROR!\n");
printf("%s\n",s);
```

**OUTPUT:**

ERROR!

n■a

**b)using freopen() to redirect a file to STDOUT in append mode**

Works fine. Everything written by printf and puts, gets appended to the file. Creates the file if it does not exist.

**Code:**

```
freopen(“appendtest.txt”,”a”,stdout);
printf(“This works\n);
```

**Initial File:**

Hello world

**Output in file:**

Hello world

This works

# SUMMARY:

The append mode cannot read from the file but it can handle writing even in binary mode keeping its consistency in writing at the end of the file even if we try to seek back.