FILE I/O IN C

Experiments on file opened in “w+” mode

Roll no :

1805061

1805073

1805085

1805097

1805109

1805121

18/20

1. **Output operation:**

   a) **fputs():**

   File output using fputs() works as expected. A text file named “out.txt” is opened in “w+” mode.

   Code:

   ```
   FILE *fp=fopen(“out.txt”, “w+”);
   char str[]= “Hello, world!”;
   fputs(str, fp);
   fclose(fp);
   ```

   Output:

   ```
   Hello, world!
   ```

   b) **fputc():**

   File output using fputc() works fine. A text file named “out.txt” is opened in “w+” mode.

   Code:

   ```
   FILE *fp=fopen(“out.txt”, “w+”);
   char ch= ‘H’;
   fputc(ch, fp);
   fclose(fp);
   ```

   Output:

   ```
   H
   ```

   c) **fprintf():**

   File output using fprintf() works as expected. A text file named “out.txt” is opened in “w+” mode.

   Code:

   ```
   FILE *fp=fopen(“out.txt”, “w+”);
   fprintf( fp, “Hello, world!”);
   fclose(fp);
   ```

   Output:

   ```
   Hello, world!
   ```

   d) **fwrite():**

   File output using fwrite() works fine as long as the “number of objects” passed as the parameter of fwrite() is less than or equal to the actual byte size of the string that is to be input. Otherwise, the output is gibberish (viewed in text mode with a text editor). If the parameter passed is less than the string length, the output is truncated. Outputting arrays other than the character ones with fwrite() gave gibberish results. A text file named “out.txt” is opened in “w+” mode.

   Code:

   ```
   FILE *fp=fopen(“out.txt”, “w+”);
   char str[]= “Hello, world!”;
   fwrite(str,sizeof(char),10,fp);
   fclose(fp);
   ```

   Output:

```
Hello, wor
```

Code:

```
FILE *fp=fopen(“out.txt”, “w+”);
char str[]= “Hello, world!”;
fwrite(str,sizeof(char),13,fp);
fclose(fp);
```

Output:

```
Hello, world!
```

Code:

```
FILE *fp=fopen(“out.txt”, “w+”);
char str[]= “Hello, world!”;
fwrite(str,sizeof(char),20,fp);
fclose(fp);
```

Output:

```
效汬∀眠牯摬!襹異1
```

Code:

```
FILE *fp=fopen(“out.txt”, “w+”);
float arr[]= {0,1,2,3};
fwrite(arr,sizeof(float),4,fp);
fclose(fp);
```

Output:

```
€?  @  @@
```

2. Input operations:

Opening a file in “w+” erases all the data contained within that file. So, the following experiments are done after opening a file “in.txt” in “w+” using FILE pointer fp and then writing “Hello, world!” in it.

a) fgetc():

File input using fgetc() works fine.

Code:

```
char ch=fgetc(fp);
putchar(ch);
fclose(fp);
```

Output: *(on console)*

```
H
```

b) **fgets():**

File input using fgets() works as expected.

Code:

```
fgets(str, 80, fp);
printf(“%s”,str);
fclose(fp);
```

Output: *(on console)*

```
Hello, world!
```

c) **fscanf():**

File input using fscanf() works fine as well.

Code:

```
fscanf(fp, “%s”, str);
printf(“%s”,str);
fclose(fp);
```

Output: *(on console)*

```
Hello,
```

d) **fread():**

File input using fread() crashes the program.

Code:

```
fread(str, sizeof(char), 13, fp);
printf(“%s”, str);
fclose(fp);
```

Output: *(on console)*

```
[program crashed]
```

3. **Random access:**

Random access works perfectly. A file is opened in “w+” mode using the FILE pointer fp and the string “Hello, world!” is written using FILE output functions. Then the following experiment is conducted.

Code:

```
fseek(fp,4,SEEK_SET);
char ch;
fgetc(ch,fp);
putchar(ch);
fclose(fp);
```

Output:

```
o
```

Summary: All the text mode input-output works fine on the file opened in “w+”. However, opening a file in “w+” erases the data contained within. Some binary I/O operations causes unexpected outcomes.

[Experiments are done using gcc on a 64-bit Windows operating system]