Systems Programming

Assignment-3

Haehyun Cho

School of Software
Soongsil University

Cyber Security
Research Center

Goal

In this programming assignment, we will design our own core input and output functions of the C standard I/O library.

The goal is to implement C I/O functions by using UNIX I/O functions

I/O Functions we need to implement

1. **fopen:** opens a file
2. **fread:** reads from a file
3. **fwrite:** writes to a file
4. **fflush:** synchronizes a stream with an actual file
5. **fseek:** moves the file position to a specific location in a file
6. **feof:** check for the end-of-file
7. **fclose:** closes a file

fopen()

FILE *fopen(const char *pathname, const char *mode);

r	Open text file for reading. The stream is positioned at the beginning of the file.
r+	Open for reading and writing. The stream is positioned at the beginning of the file.
w	Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.
w+	Open for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.
a	Open for appending (writing at end of file). The file is created if it does not exist. The stream is positioned at the end of the file.
a+	Open for reading and appending (writing at end of file). The file is created if it does not exist. Output is always appended to the end of the file. POSIX is silent on what the initial read position is when using this mode, but, the stream must be positioned at the end of the file in this work.

fopen("/bin/sh", "r+");

fread()/fwrite()

int fread(void *ptr, int size, int nmemb, FILE *stream);

int fwrite(const void *ptr, int size, int nmemb, FILE *stream);

The function **fread**() reads *nmemb* items of data, each *size* bytes long, from the stream pointed to by *stream*, storing them at the location given by *ptr*.

The function **fwrite**() writes *nmemb* items of data, each *size* bytes long, to the stream pointed to by *stream*, obtaining them from the location given by *ptr*.

fread()/fwrite()

On success, **fread**() and **fwrite**() return the number of items read or written. This number equals the number of bytes transferred only when *size* is 1. If an error occurs, or the end of the file is reached, the return value is a short item count (or zero). The file position indicator for the stream is advanced by the number of bytes successfully read or written.

fflush()

int fflush(FILE *stream);

For output streams, **fflush**() forces a write of all user-space buffered data for the given output or update *stream* via the stream's underlying write function.

We don’t use fflush() for read-only filestreams .

Return value:

Upon successful completion **0** is returned. Otherwise, **EOF** is returned.

fseek()

int fseek(FILE **stream*, int *offset*, int *whence*);

The *fseek*() function shall set the file-position indicator for the stream pointed to by *stream*. The new position, measured in bytes from the beginning of the file, shall be obtained by adding *offset* to the position specified by *whence*. The specified point is the beginning of the file for SEEK_SET, the current value of the file-position indicator for SEEK_CUR, or end-of-file for SEEK_END.

The *fseek*() and *fseeko*() functions shall return 0 if they succeed. Otherwise, they shall return -1.

feof()

int feof(FILE **stream*);

The *feof*() function shall test the end-of-file indicator for the stream pointed to by *stream*.

The *feof*() function shall return -1 if and only if the end- of-file indicator is set for *stream*.

fclose()

int fclose(FILE **stream*);

The *fclose*() function shall cause the stream pointed to by *stream* to be flushed and the associated file to be closed. Any unwritten buffered data for the stream shall be written to the file; any unread buffered data shall be discarded.

Upon successful completion, *fclose*() shall return 0; otherwise, it shall return EOF.

example of “mystdio.h”

```
#define BUFSIZE 1024
#define EOF -1
#define stdin  1
#define stdout 2
#define stderr 3
#define SEEK_SET 0
#define SEEK_CUR 1
#define SEEK_END 2
```

```
typedef struct myFile {
        int fd;
        int pos;
        int size;
        int mode;
        int flag;
        char lastop;
        bool eof;
        char *buffer;
}FILE;
```

```
typedef struct myFile
{
	int fd;
	int pos;
	int size;
	int mode;
	int flag;
	char lastop;
	bool eof;
	char *buffer;
}FILE;
```

myFile 구조체는 예시이며,
반드시 이대로 따라서 할 필요는 없음.

본인이 필요한 대로 Buffering을 사용한
File IO에 필요한 구조체를 정의하면 됨.

!!! What you need to submit !!!

Just the header file:

“mystdio.h”

But, compress the header file in a zipfile with the other files.

Your *zipfile* name must be your student ID#.

e.g., 20219876.zip

If you did not follow the rule, your point will get a penalty point (-2).