CS111，C Programming

Lab / IO

黄嘉炜

huangjw3@mail.sustech.edu.cn

Outline

- Review
- I/O - Overview
- I/O - FILE
- Assignment

Review：分数 + 学号排序?

Output

You need to sort the N students by their total score, and finally output Top-K the students' in descending order.

And the each output line contains: 1 student ID, name, Math score, Physics score, English score, Physical education score, and **total score**.

Note that:

- when $(K+1)$th students is same total score with the Kth student, also output $(K+1)$th student, until the total score not the same.
- The output order of students in the same total score, order by the student ID by ascending.

Require

Use the `struct` to solve this problem.

Review：search keywords

write a function that: search keyword case insensitive, and return top-N occurrences.

Hint:

- Function define: int search_keyword_case_insensitive(const char* str, const char* keyword, int top_n, int* positions)
  where: positions is a pointer to an integer array to store the starting positions of {top_n} occurrences
- The return value (int) is the length of valid value in {positions} array after searching. And the return value <= {top_n}.

Coding Together?

基本思路

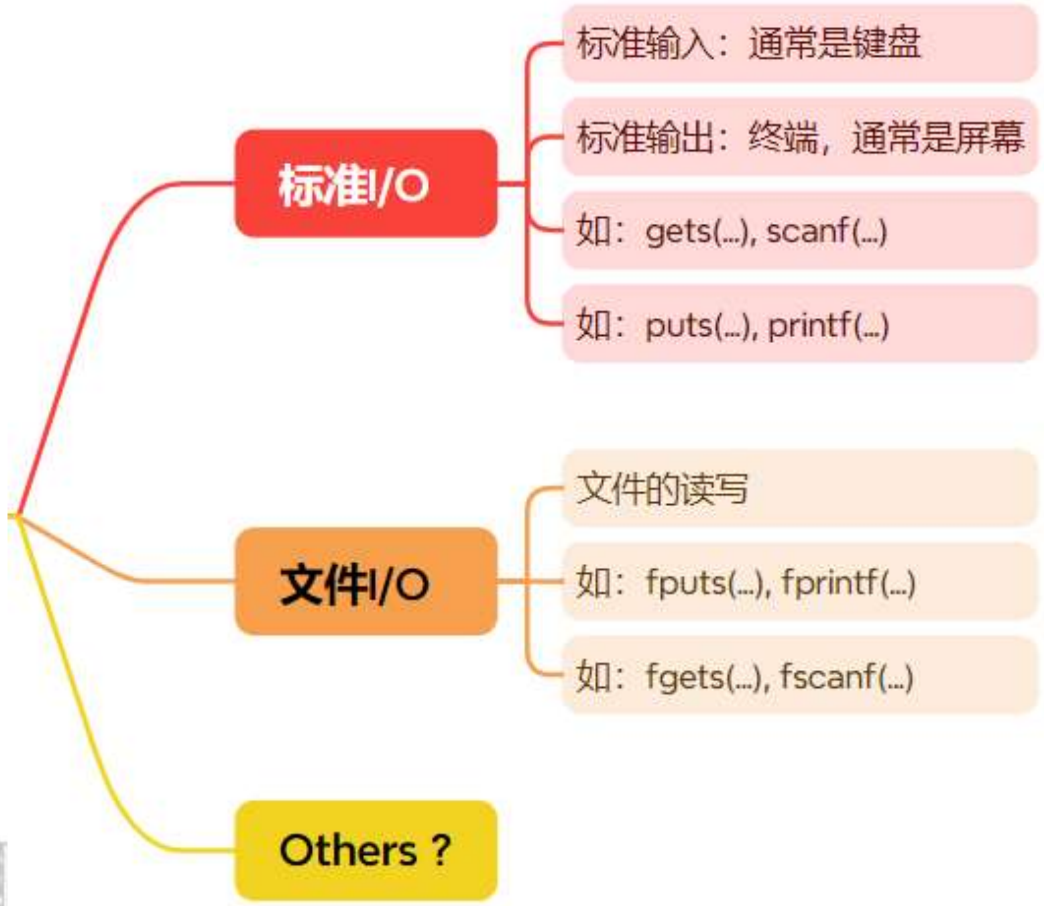
- 遍历长字符串str
- 每次遍历： (str+i) 作为字符串起点，跟 keyword 比较
  - 统一转为小写，再比较
  - 如果一致，记录 i 作为 position，count+1

Outline

- Review
- **I/O - Overview**
- I/O - FILE
- Assignment

I/O: Overview

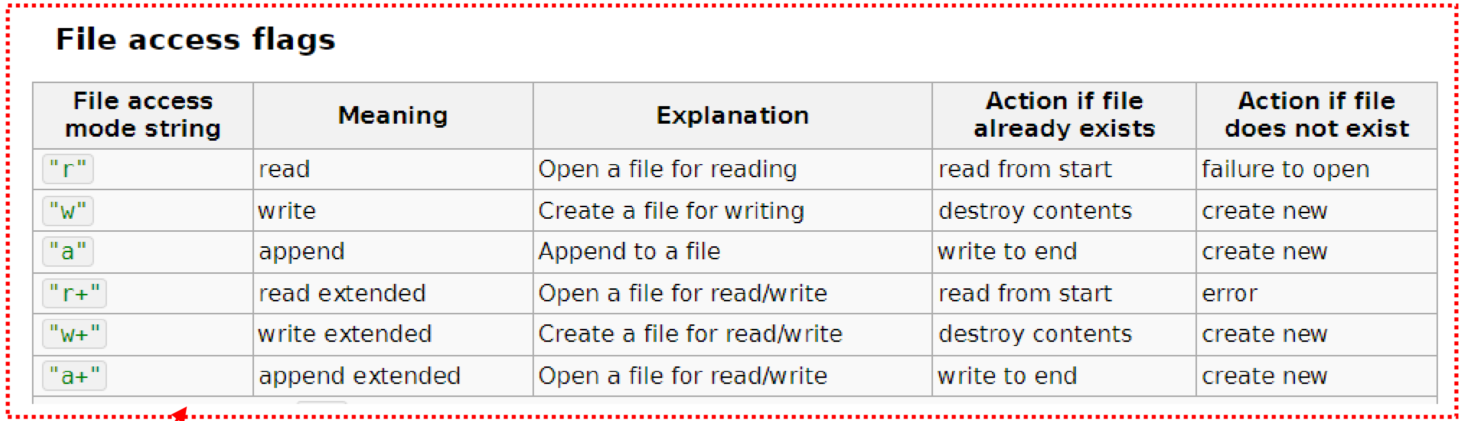
I/O (Input/Output) – 程序与外界（如用户、文件、设备等）进行交互的一种方式



File open & close

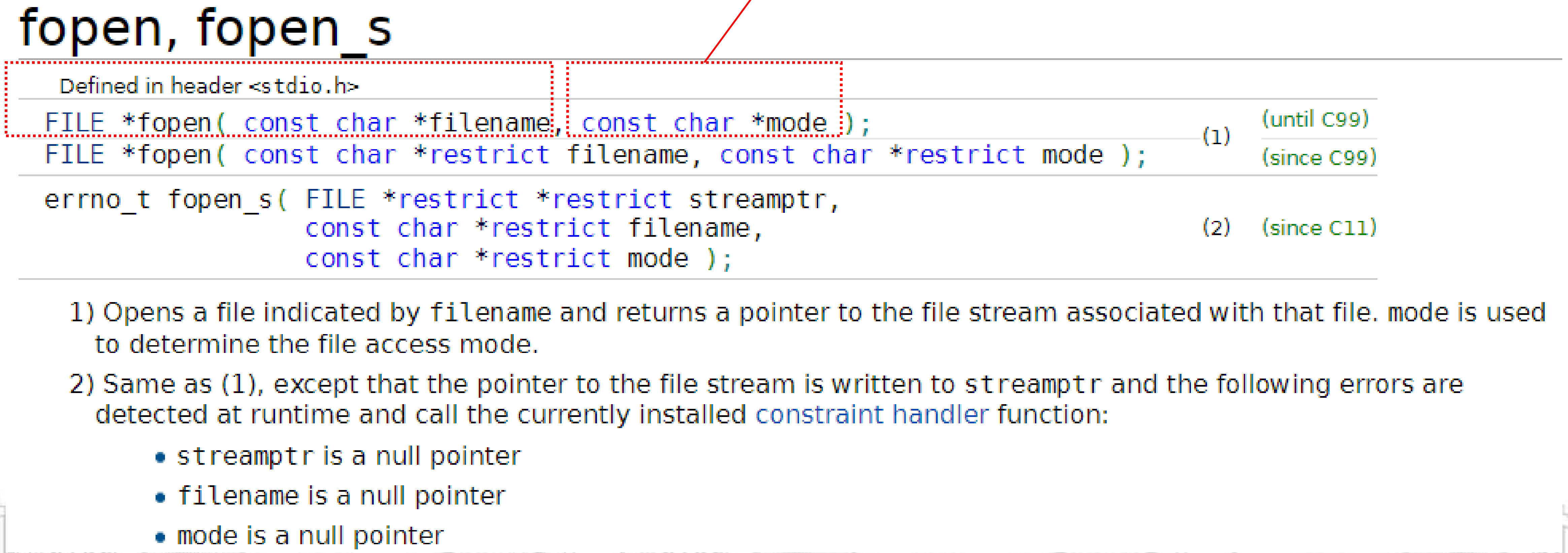
fopen(..)

https://en.cppreference.com/w/c/io/fopen



File access flags

File access mode string	Meaning	Explanation	Action if file already exists	Action if file does not exist
"r"	read	Open a file for reading	read from start	failure to open
"w"	write	Create a file for writing	destroy contents	create new
"a"	append	Append to a file	write to end	create new
"r+"	read extended	Open a file for read/write	read from start	error
"w+"	write extended	Create a file for read/write	destroy contents	create new
"a+"	append extended	Open a file for read/write	write to end	create new



fopen, fopen_s

Defined in header <stdio.h>

```
FILE *fopen( const char *filename, const char *mode );                              (1) (until C99)
FILE *fopen( const char *restrict filename, const char *restrict mode );                (since C99)
errno_t fopen_s( FILE *restrict *restrict streamptr,
                 const char *restrict filename,                                      (2) (since C11)
                 const char *restrict mode );
```

1) Opens a file indicated by filename and returns a pointer to the file stream associated with that file. mode is used to determine the file access mode.

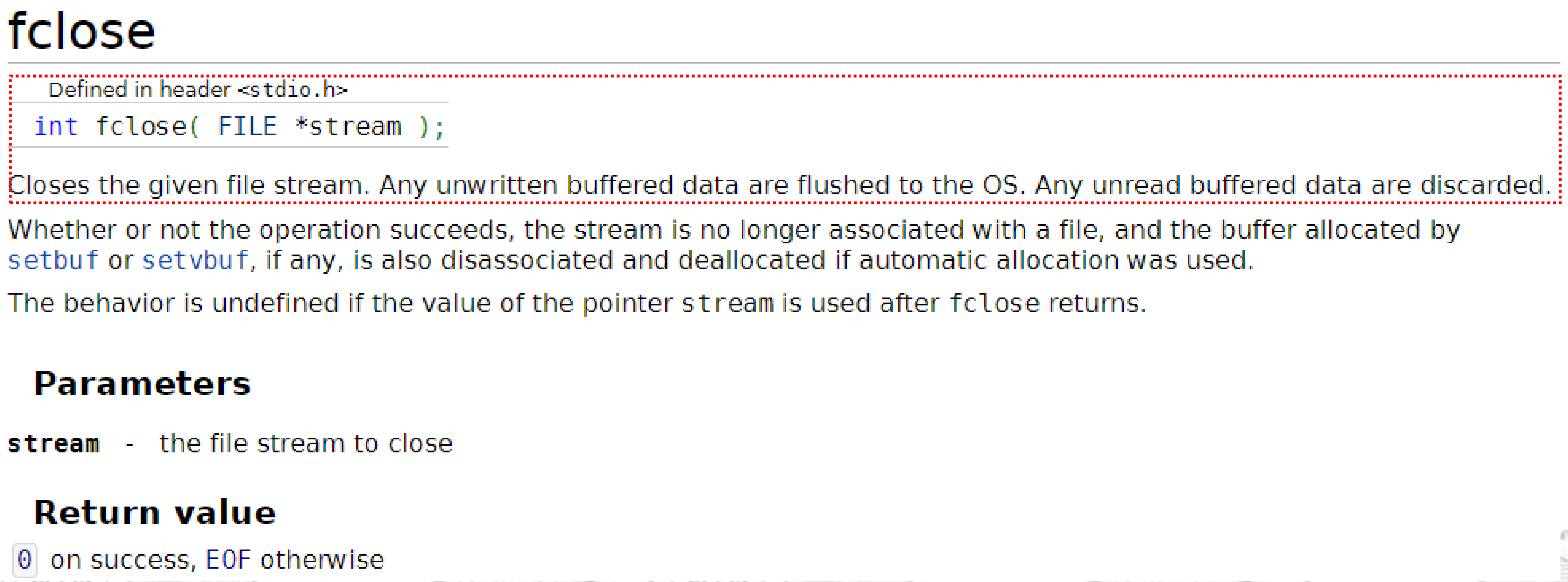
2) Same as (1), except that the pointer to the file stream is written to streamptr and the following errors are detected at runtime and call the currently installed constraint handler function:

- streamptr is a null pointer
- filename is a null pointer
- mode is a null pointer

File open & close

fclose(..)

https://en.cppreference.com/w/c/io/fclose



fclose

Defined in header <stdio.h>

```
int fclose( FILE *stream );
```

Closes the given file stream. Any unwritten buffered data are flushed to the OS. Any unread buffered data are discarded.

Whether or not the operation succeeds, the stream is no longer associated with a file, and the buffer allocated by `setbuf` or `setvbuf`, if any, is also disassociated and deallocated if automatic allocation was used.

The behavior is undefined if the value of the pointer `stream` is used after `fclose` returns.

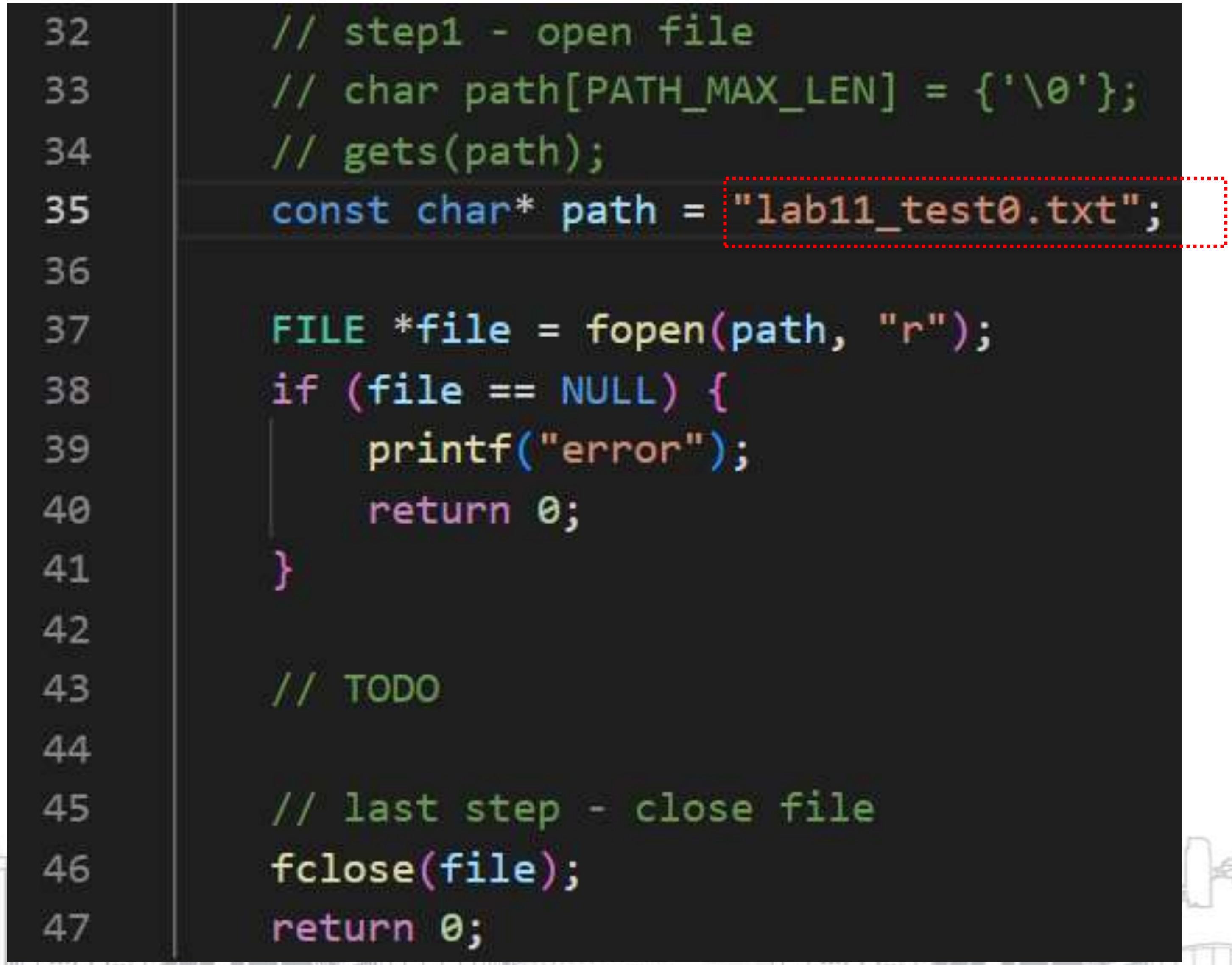
Parameters

stream - the file stream to close

Return value

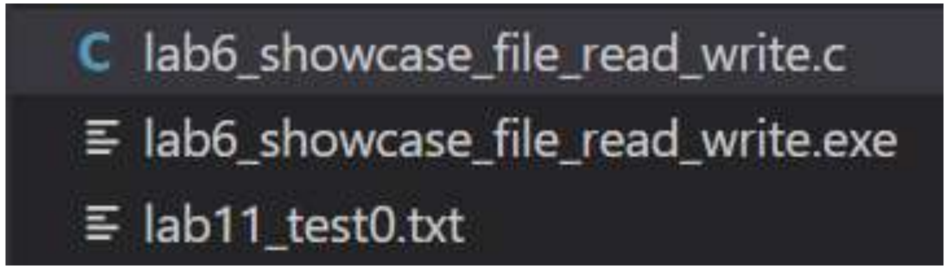
`0` on success, `EOF` otherwise

File open & close



```c
    // step1 - open file
    // char path[PATH_MAX_LEN] = {'\0'};
    // gets(path);
    const char* path = "lab11_test0.txt";

    FILE *file = fopen(path, "r");
    if (file == NULL) {
        printf("error");
        return 0;
    }

    // TODO

    // last step - close file
    fclose(file);
    return 0;
```

当前目录的文件



```
lab6_showcase_file_read_write.c
lab6_showcase_file_read_write.exe
lab11_test0.txt
```

File Read

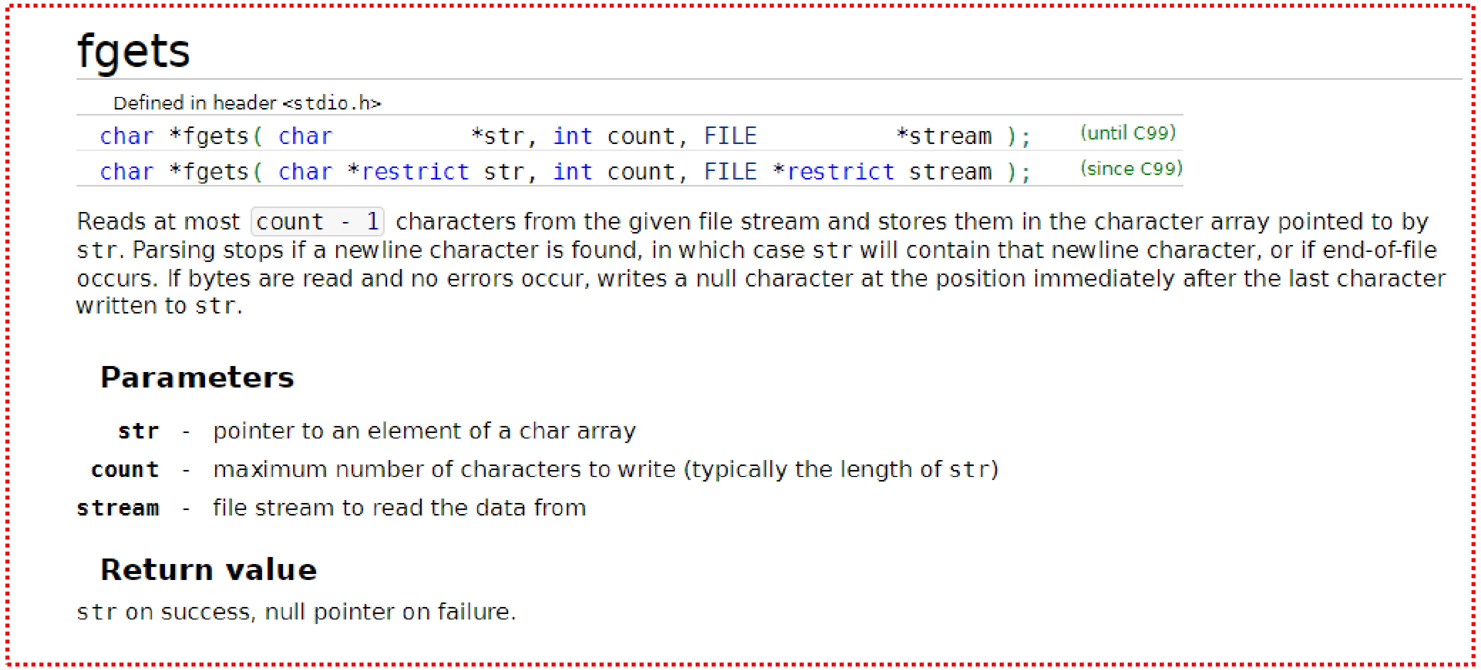
Key Functions:

- fgets(…) - read a string
- fscanf(…) - read a formatted string, …
- fread(…) – read raw binary data, …

File Read

Key Functions:

- **fgets(…) - read a string**
- fscanf(…) - read a formatted string, …
- fread(…) – read raw binary data, …



fgets

Defined in header `<stdio.h>`

```
char *fgets( char          *str, int count, FILE          *stream );   (until C99)
char *fgets( char *restrict str, int count, FILE *restrict stream );   (since C99)
```

Reads at most `count - 1` characters from the given file stream and stores them in the character array pointed to by `str`. Parsing stops if a newline character is found, in which case `str` will contain that newline character, or if end-of-file occurs. If bytes are read and no errors occur, writes a null character at the position immediately after the last character written to `str`.

Parameters

str	-	pointer to an element of a char array
count	-	maximum number of characters to write (typically the length of `str`)
stream	-	file stream to read the data from

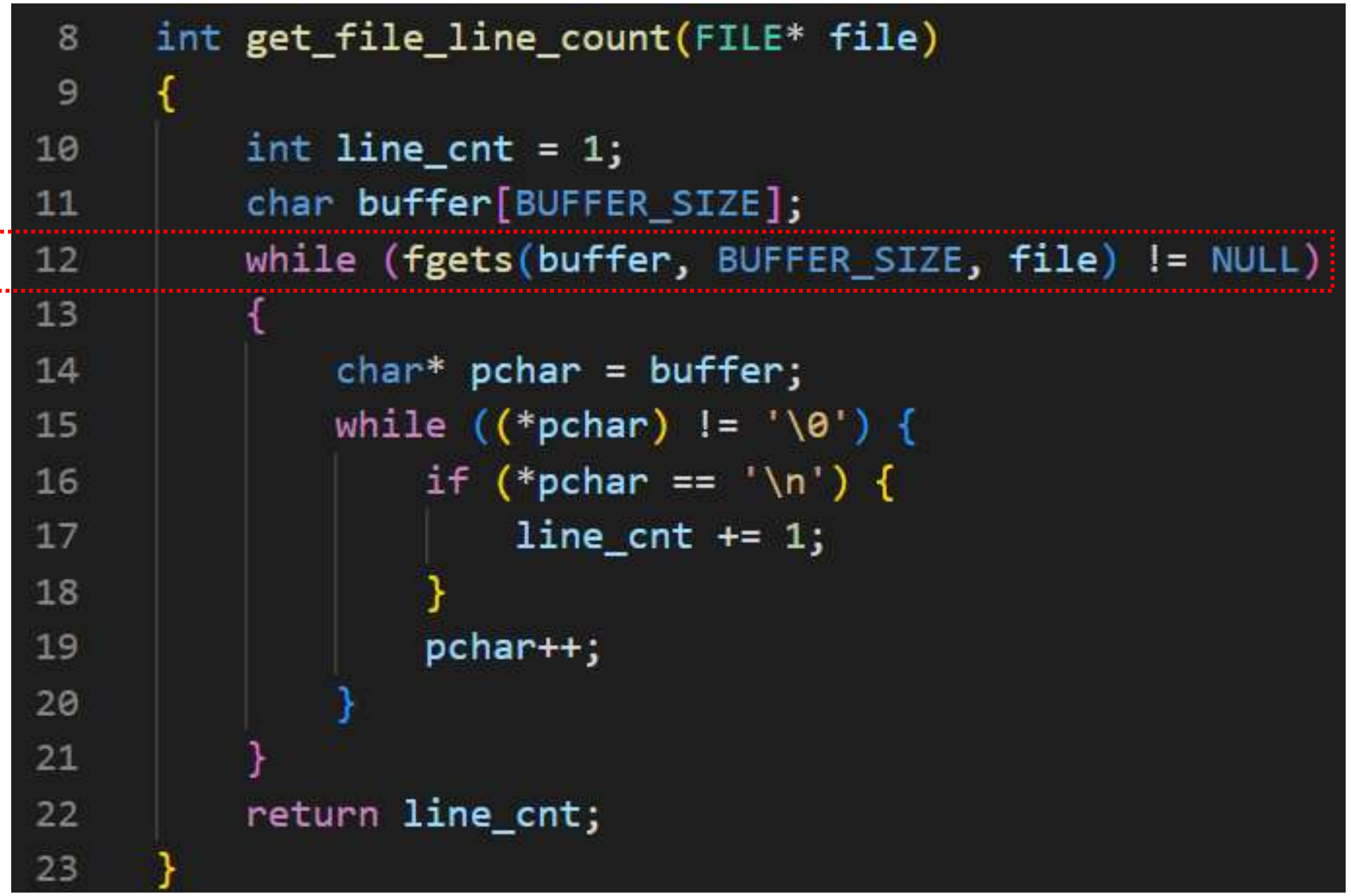
Return value

`str` on success, null pointer on failure.

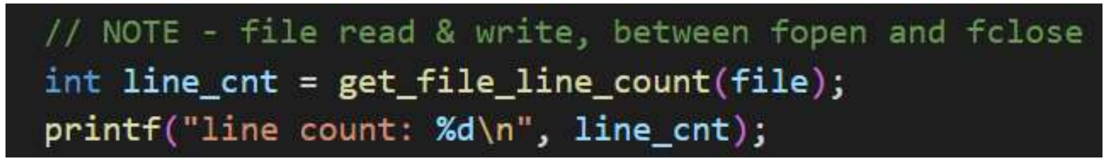
File Read

Key Functions:

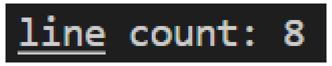
- **fgets(…) - read a string**
- fscanf(…) - read a formatted string, …
- fread(…) – read raw binary data, …



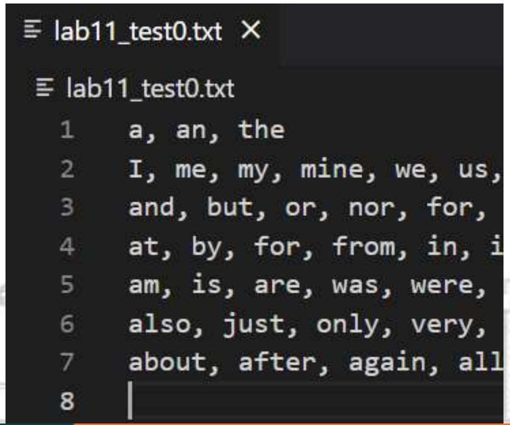
```c
int get_file_line_count(FILE* file)
{
    int line_cnt = 1;
    char buffer[BUFFER_SIZE];
    while (fgets(buffer, BUFFER_SIZE, file) != NULL)
    {
        char* pchar = buffer;
        while ((*pchar) != '\0') {
            if (*pchar == '\n') {
                line_cnt += 1;
            }
            pchar++;
        }
    }
    return line_cnt;
}
```



```c
// NOTE - file read & write, between fopen and fclose
int line_cnt = get_file_line_count(file);
printf("line count: %d\n", line_cnt);
```



```
line count: 8
```



lab11_test0.txt ×

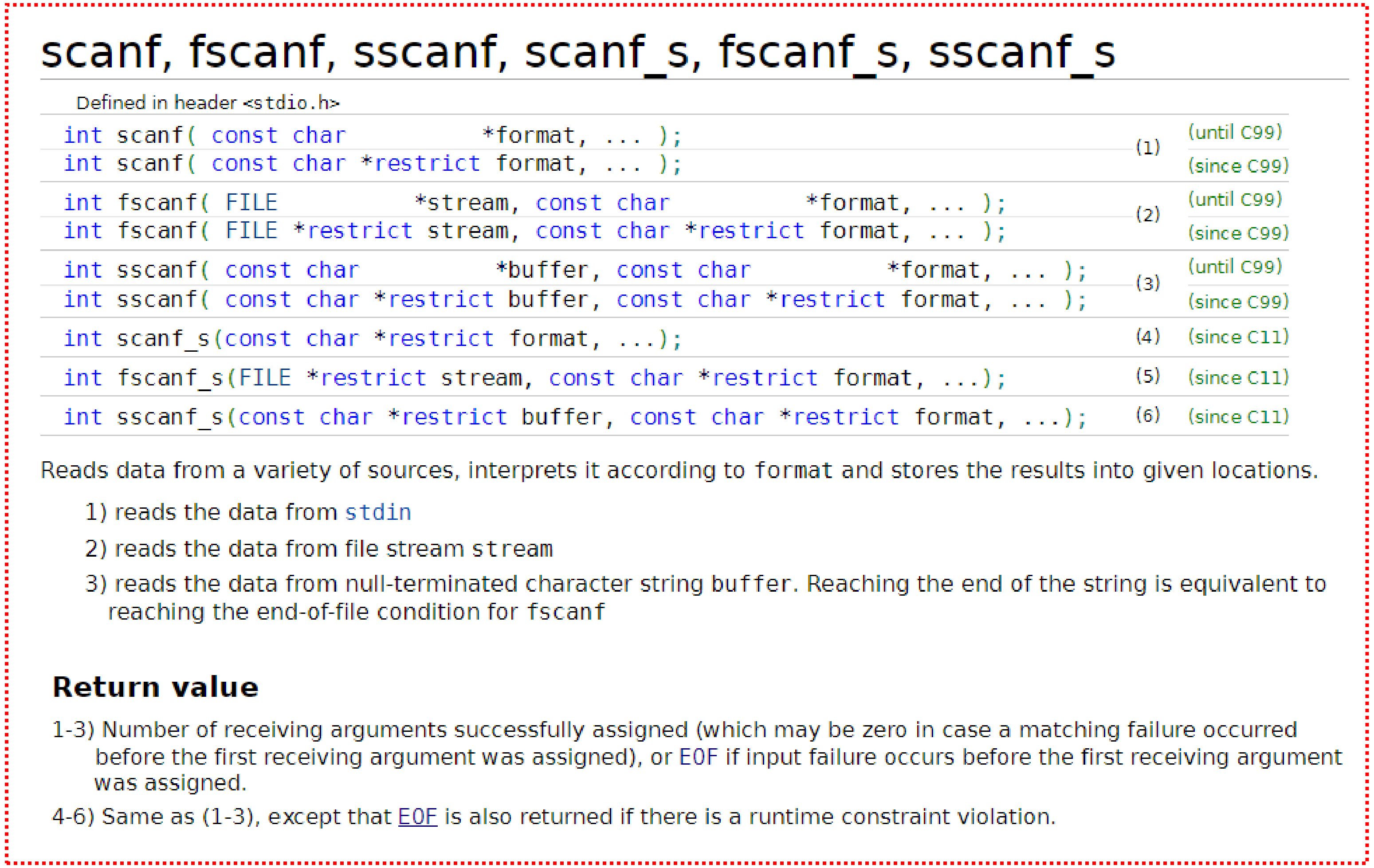
lab11_test0.txt

```
a, an, the
I, me, my, mine, we, us,
and, but, or, nor, for,
at, by, for, from, in, i
am, is, are, was, were,
also, just, only, very,
about, after, again, all

```

File Read

Key Functions:

- fgets(…) - read a string
- **fscanf(…) - read a formatted string, …**
- fread(…) – read raw binary data, …



scanf, fscanf, sscanf, scanf_s, fscanf_s, sscanf_s

Defined in header <stdio.h>

```
int scanf( const char          *format, ... );                                          (1) (until C99)
int scanf( const char *restrict format, ... );                                              (since C99)
int fscanf( FILE          *stream, const char          *format, ... );                  (2) (until C99)
int fscanf( FILE *restrict stream, const char *restrict format, ... );                      (since C99)
int sscanf( const char          *buffer, const char          *format, ... );            (3) (until C99)
int sscanf( const char *restrict buffer, const char *restrict format, ... );                (since C99)
int scanf_s(const char *restrict format, ...);                                          (4) (since C11)
int fscanf_s(FILE *restrict stream, const char *restrict format, ...);                  (5) (since C11)
int sscanf_s(const char *restrict buffer, const char *restrict format, ...);            (6) (since C11)
```

Reads data from a variety of sources, interprets it according to `format` and stores the results into given locations.

1) reads the data from `stdin`

2) reads the data from file stream `stream`

3) reads the data from null-terminated character string `buffer`. Reaching the end of the string is equivalent to reaching the end-of-file condition for `fscanf`

Return value

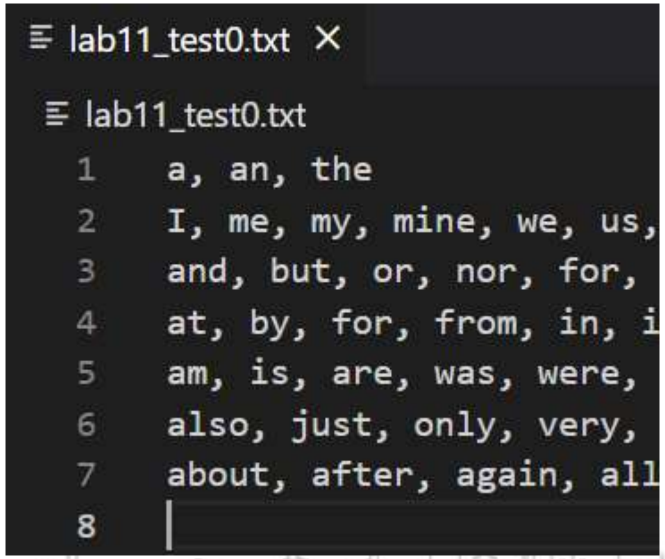
1-3) Number of receiving arguments successfully assigned (which may be zero in case a matching failure occurred before the first receiving argument was assigned), or EOF if input failure occurs before the first receiving argument was assigned.

4-6) Same as (1-3), except that EOF is also returned if there is a runtime constraint violation.

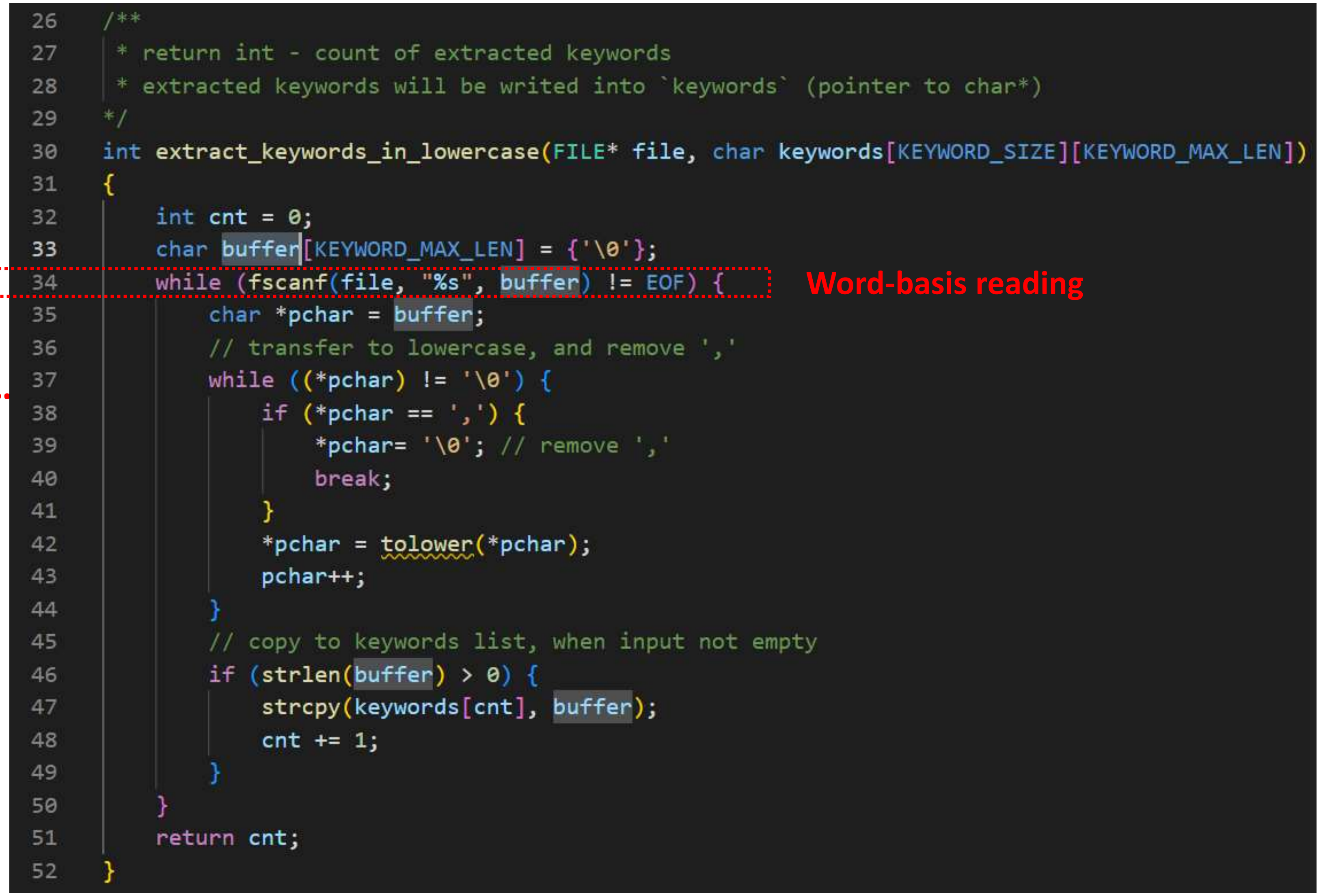
File Read

Key Functions:

- fgets(…) - read a string
- **fscanf(…) - read a formatted string, …**
- fread(…) – read raw binary data, …



```
lab11_test0.txt
a, an, the
I, me, my, mine, we, us,
and, but, or, nor, for,
at, by, for, from, in, i
am, is, are, was, were,
also, just, only, very,
about, after, again, all
```



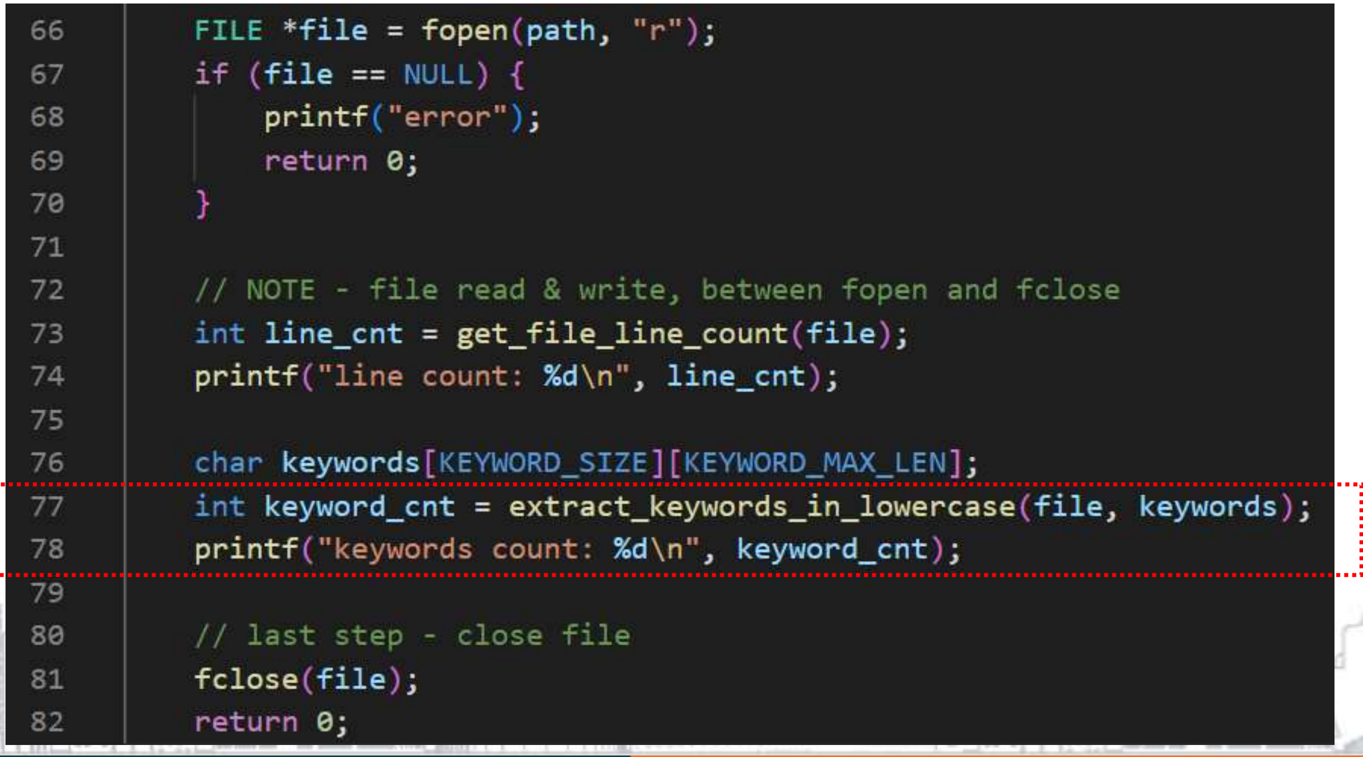
```c
/**
 * return int - count of extracted keywords
 * extracted keywords will be writed into `keywords` (pointer to char*)
*/
int extract_keywords_in_lowercase(FILE* file, char keywords[KEYWORD_SIZE][KEYWORD_MAX_LEN])
{
    int cnt = 0;
    char buffer[KEYWORD_MAX_LEN] = {'\0'};
    while (fscanf(file, "%s", buffer) != EOF) {
        char *pchar = buffer;
        // transfer to lowercase, and remove ','
        while ((*pchar) != '\0') {
            if (*pchar == ',') {
                *pchar= '\0'; // remove ','
                break;
            }
            *pchar = tolower(*pchar);
            pchar++;
        }
        // copy to keywords list, when input not empty
        if (strlen(buffer) > 0) {
            strcpy(keywords[cnt], buffer);
            cnt += 1;
        }
    }
    return cnt;
}
```

Word-basis reading

File Read

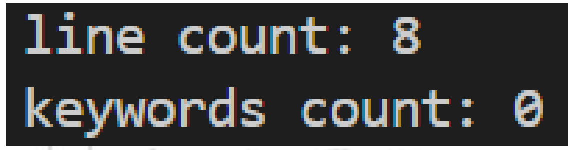
Key Functions:

- fgets(…) - read a string
- **fscanf(…) - read a formatted string, …**
- fread(…) – read raw binary data, …



```c
    FILE *file = fopen(path, "r");
    if (file == NULL) {
        printf("error");
        return 0;
    }

    // NOTE - file read & write, between fopen and fclose
    int line_cnt = get_file_line_count(file);
    printf("line count: %d\n", line_cnt);

    char keywords[KEYWORD_SIZE][KEYWORD_MAX_LEN];
    int keyword_cnt = extract_keywords_in_lowercase(file, keywords);
    printf("keywords count: %d\n", keyword_cnt);

    // last step - close file
    fclose(file);
    return 0;
```

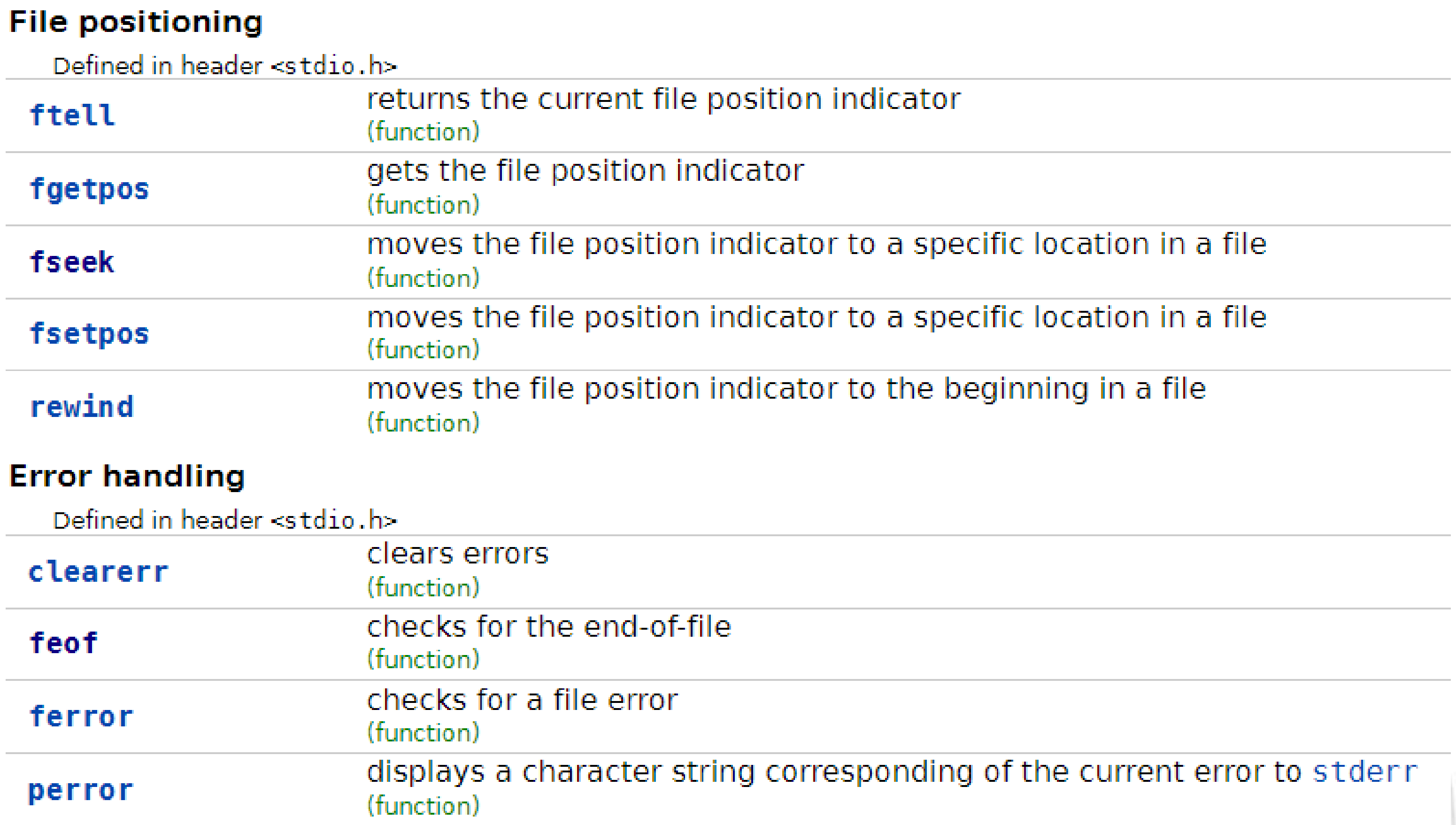
Why? No keywords…



```
line count: 8
keywords count: 0
```

More: File Status

Ref, https://en.cppreference.com/w/c/io



File positioning

Defined in header <stdio.h>

`ftell`	returns the current file position indicator (function)
`fgetpos`	gets the file position indicator (function)
`fseek`	moves the file position indicator to a specific location in a file (function)
`fsetpos`	moves the file position indicator to a specific location in a file (function)
`rewind`	moves the file position indicator to the beginning in a file (function)

Error handling

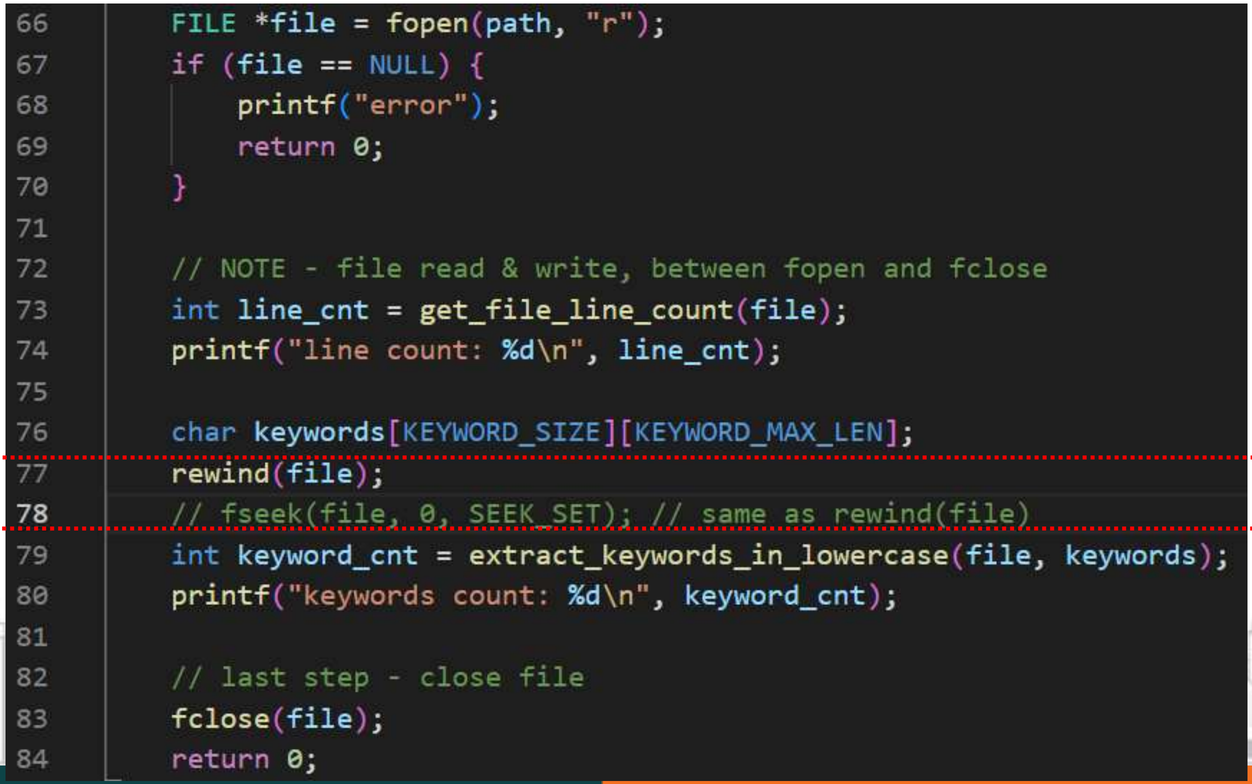
Defined in header <stdio.h>

`clearerr`	clears errors (function)
`feof`	checks for the end-of-file (function)
`ferror`	checks for a file error (function)
`perror`	displays a character string corresponding of the current error to `stderr` (function)

File Read

Key Functions:

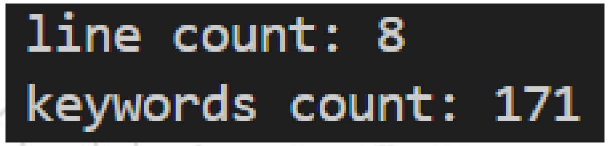
- fgets(…) - read a string
- **fscanf(…) - read a formatted string, …**
- fread(…) – read raw binary data, …



```c
    FILE *file = fopen(path, "r");
    if (file == NULL) {
        printf("error");
        return 0;
    }

    // NOTE - file read & write, between fopen and fclose
    int line_cnt = get_file_line_count(file);
    printf("line count: %d\n", line_cnt);

    char keywords[KEYWORD_SIZE][KEYWORD_MAX_LEN];
    rewind(file);
    // fseek(file, 0, SEEK_SET); // same as rewind(file)
    int keyword_cnt = extract_keywords_in_lowercase(file, keywords);
    printf("keywords count: %d\n", keyword_cnt);

    // last step - close file
    fclose(file);
    return 0;
```

When file reread all over again,

Need to rewind / fseek

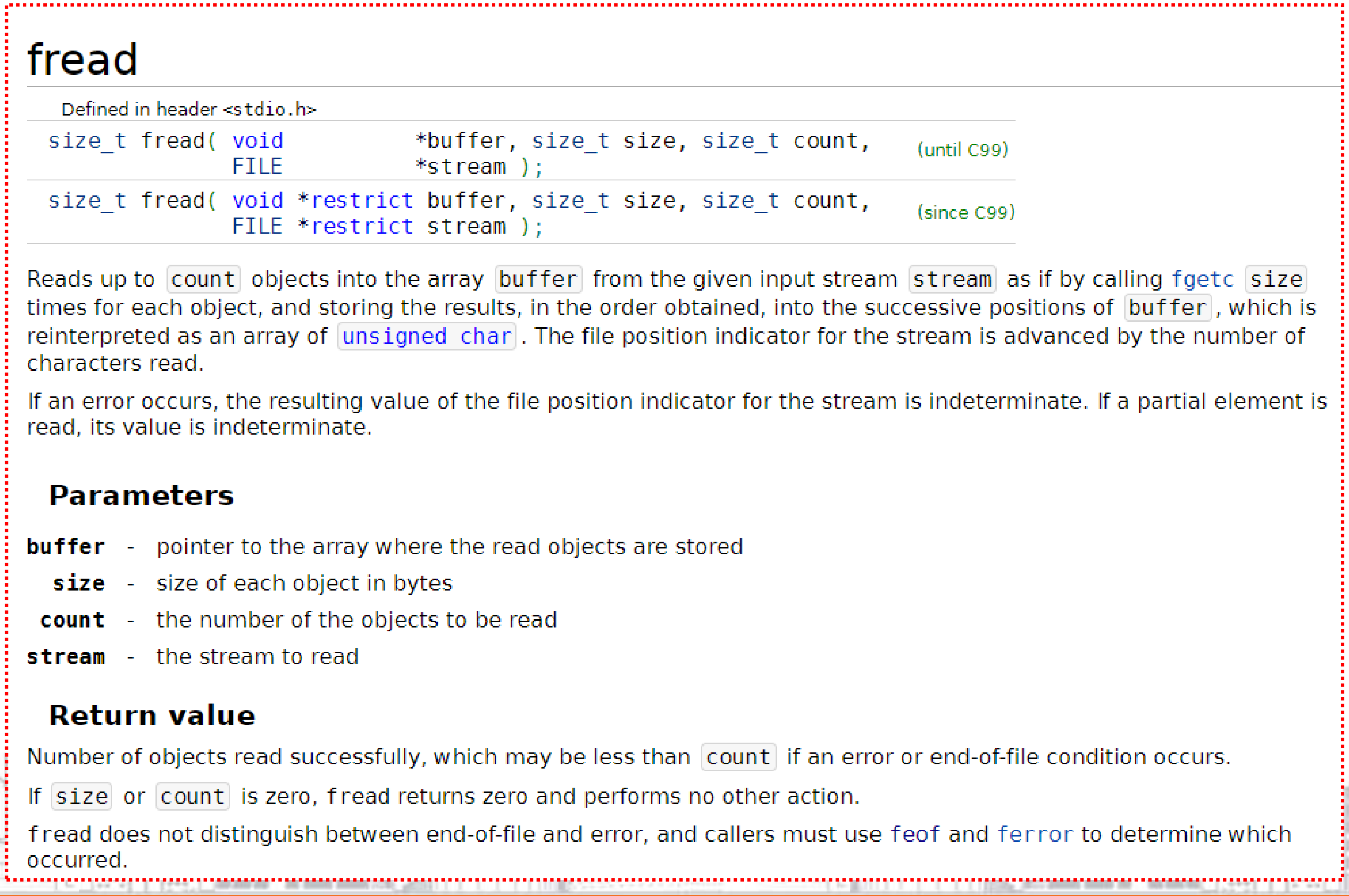


```
line count: 8
keywords count: 171
```

File Read

Key Functions:

- fgets(…) - read a string
- fscanf(…) - read a formatted string, …
- **fread(…) – read raw binary data, …**



fread

Defined in header <stdio.h>

```
size_t fread( void          *buffer, size_t size, size_t count,    (until C99)
              FILE          *stream );
size_t fread( void *restrict buffer, size_t size, size_t count,    (since C99)
              FILE *restrict stream );
```

Reads up to `count` objects into the array `buffer` from the given input stream `stream` as if by calling fgetc `size` times for each object, and storing the results, in the order obtained, into the successive positions of `buffer`, which is reinterpreted as an array of `unsigned char`. The file position indicator for the stream is advanced by the number of characters read.

If an error occurs, the resulting value of the file position indicator for the stream is indeterminate. If a partial element is read, its value is indeterminate.

Parameters

buffer	- pointer to the array where the read objects are stored
size	- size of each object in bytes
count	- the number of the objects to be read
stream	- the stream to read

Return value

Number of objects read successfully, which may be less than `count` if an error or end-of-file condition occurs.

If `size` or `count` is zero, `fread` returns zero and performs no other action.

`fread` does not distinguish between end-of-file and error, and callers must use feof and ferror to determine which occurred.

File Write

Key Functions:

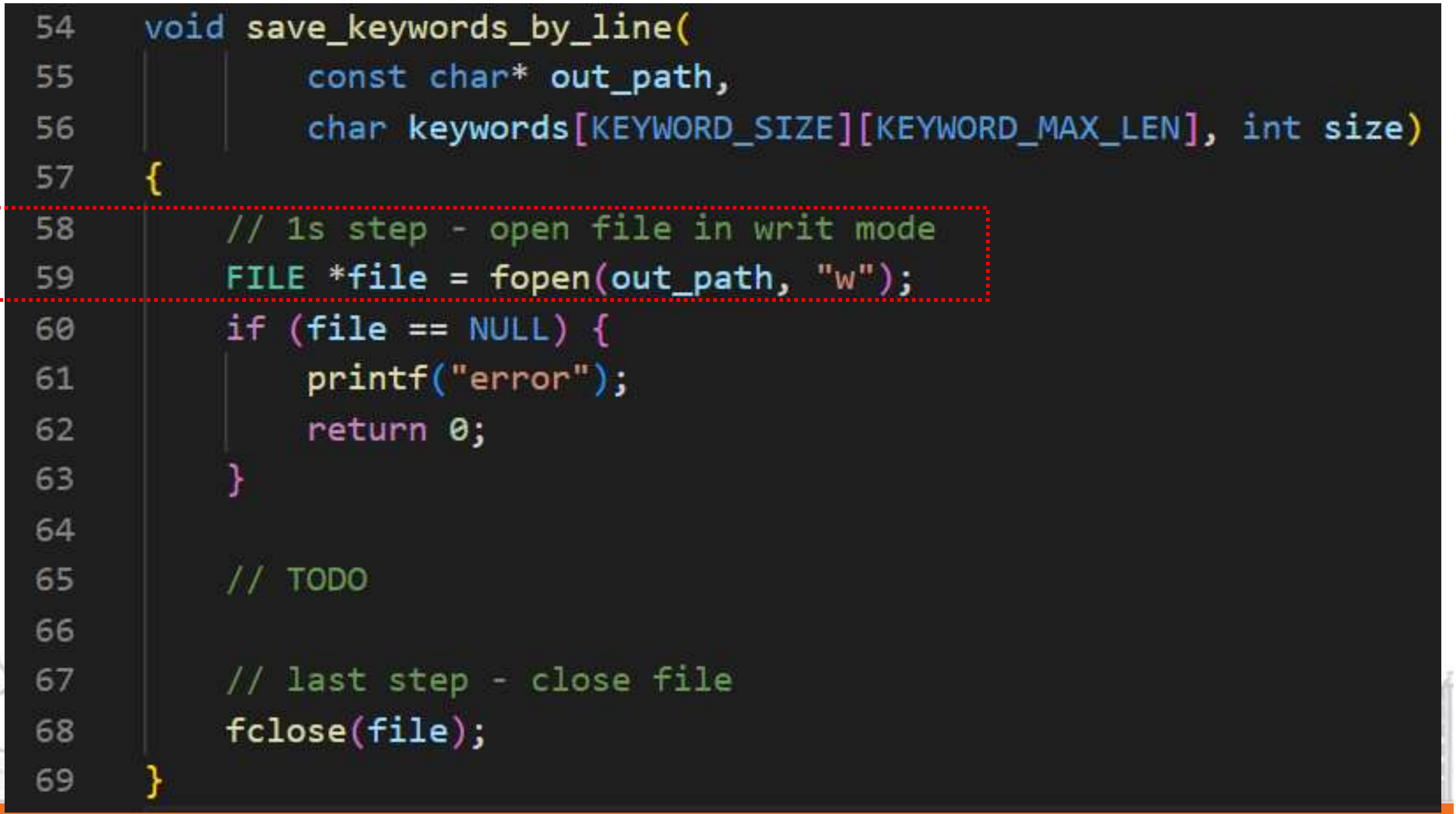
- fputs(…) – write a string
- fprintf(…) – write a formatted string, …
- fwrite(…) – write raw binary data, …

File Write

Key Functions:

- fputs(…) – write a string
- fprintf(…) – write a formatted string, …
- fwrite(…) – write raw binary data, …

Note:
Before file writing,
ensure file open in writeable mode

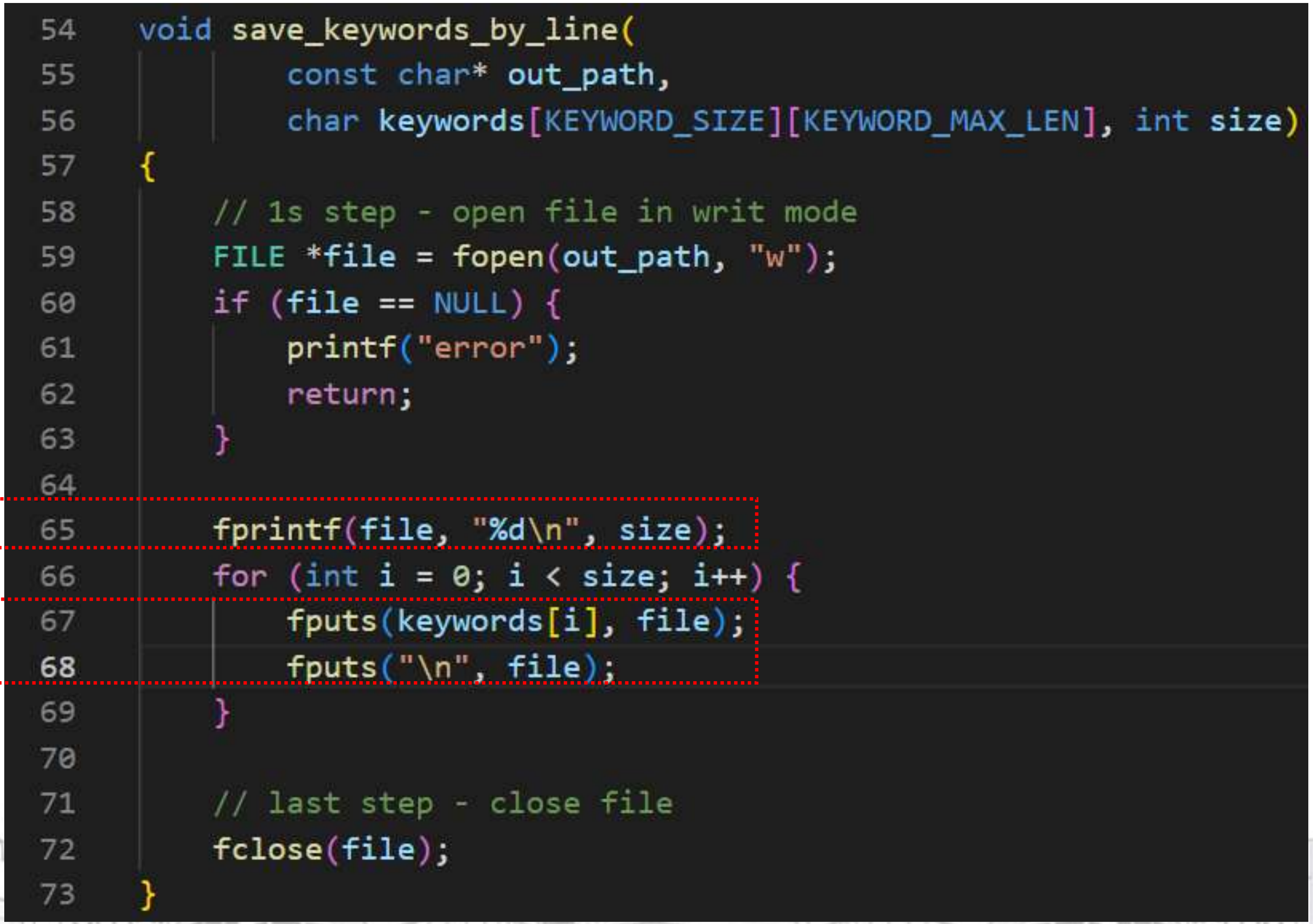


```c
void save_keywords_by_line(
        const char* out_path,
        char keywords[KEYWORD_SIZE][KEYWORD_MAX_LEN], int size)
{
    // 1s step - open file in writ mode
    FILE *file = fopen(out_path, "w");
    if (file == NULL) {
        printf("error");
        return 0;
    }

    // TODO

    // last step - close file
    fclose(file);
}
```

File Write

Key Functions:

- **fputs(…) – write a string**
- **fprintf(…) – write a formatted string, …**
- fwrite(…) – write raw binary data, …



```
void save_keywords_by_line(
        const char* out_path,
        char keywords[KEYWORD_SIZE][KEYWORD_MAX_LEN], int size)
{
    // 1s step - open file in writ mode
    FILE *file = fopen(out_path, "w");
    if (file == NULL) {
        printf("error");
        return;
    }

    fprintf(file, "%d\n", size);
    for (int i = 0; i < size; i++) {
        fputs(keywords[i], file);
        fputs("\n", file);
    }

    // last step - close file
    fclose(file);
}
```

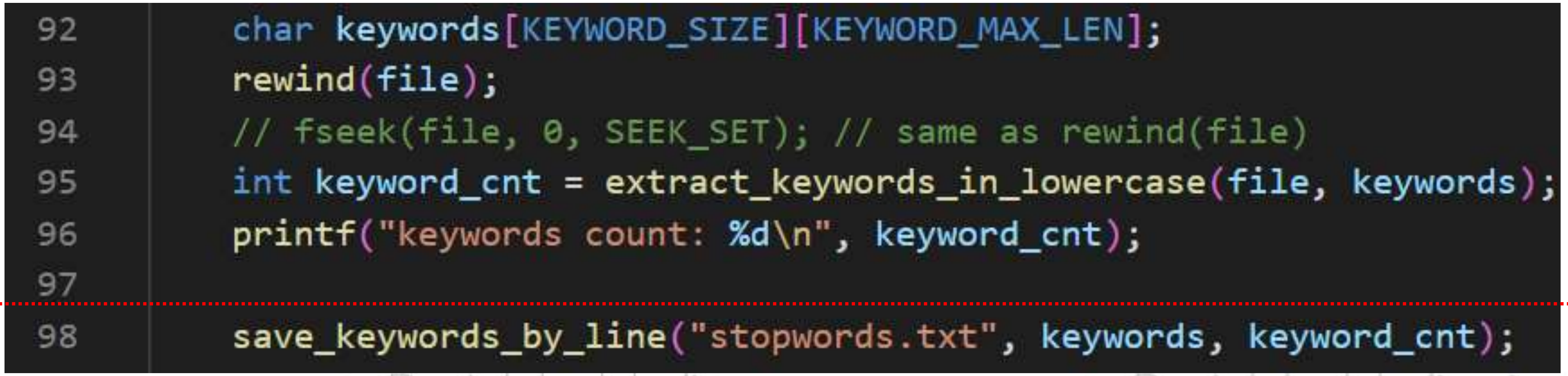
Note:

- **fputs – 作为文件输出，不会自动添加回车(“\n”)**
- puts – 作为标准输出，会在字符串末尾添加回车（“\n”）

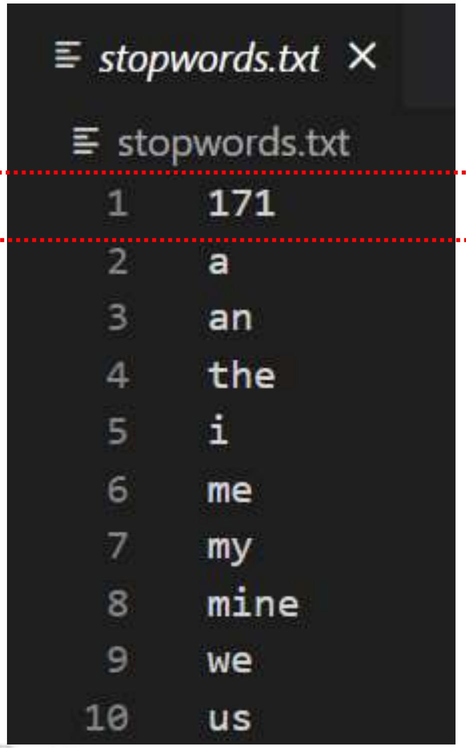
File Write

Key Functions:

- **fputs(…) – write a string**
- **fprintf(…) – write a formatted string, …**
- fwrite(…) – write raw binary data, …



```
    char keywords[KEYWORD_SIZE][KEYWORD_MAX_LEN];
    rewind(file);
    // fseek(file, 0, SEEK_SET); // same as rewind(file)
    int keyword_cnt = extract_keywords_in_lowercase(file, keywords);
    printf("keywords count: %d\n", keyword_cnt);

    save_keywords_by_line("stopwords.txt", keywords, keyword_cnt);
```



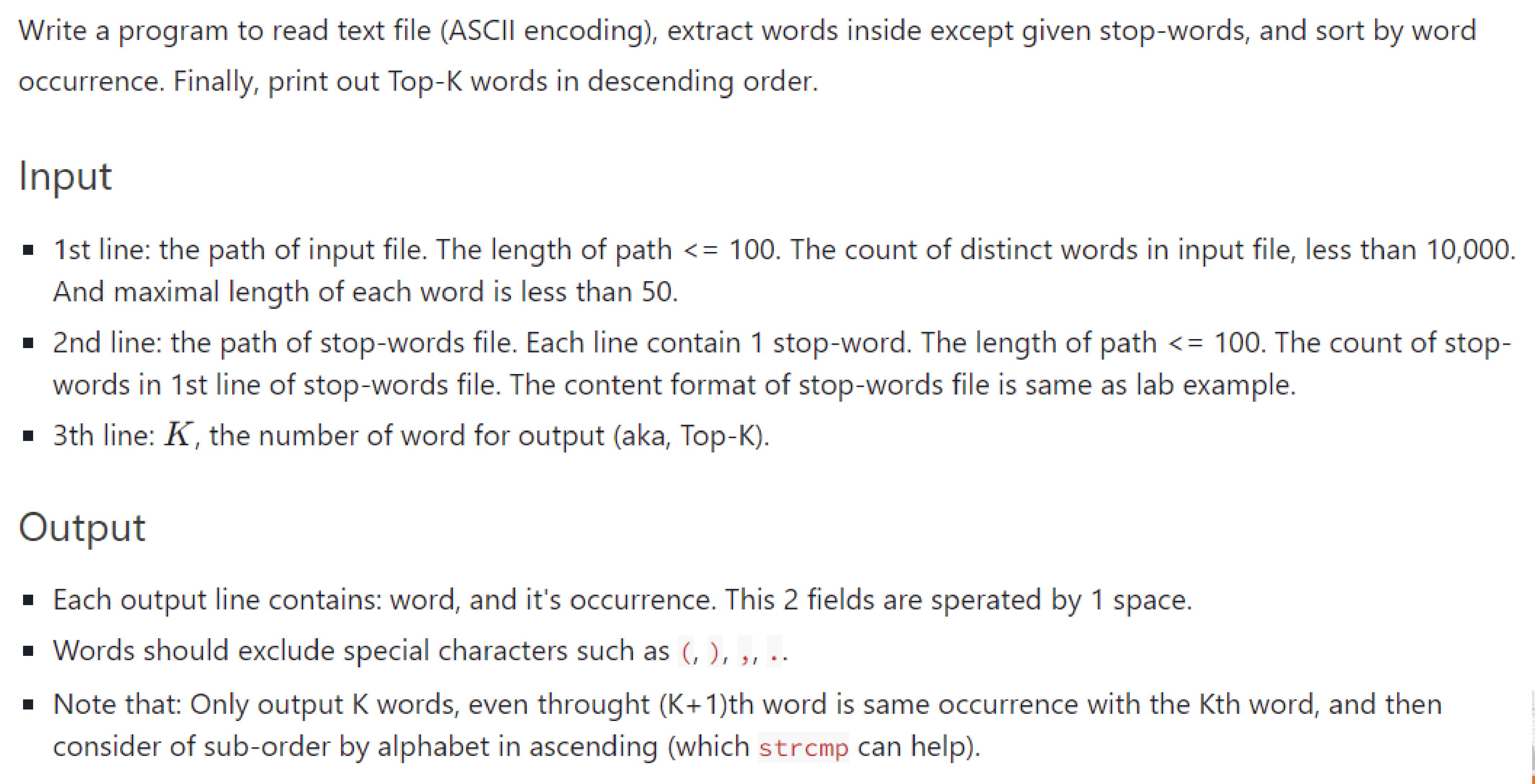
```
stopwords.txt ×
stopwords.txt
171
a
an
the
i
me
my
mine
we
us
```

Outline

- Review
- I/O - Overview
- I/O - FILE
- **Assignment**

Assignment) 词语统计

OJ 初步测试
+
现场 Code Review



Write a program to read text file (ASCII encoding), extract words inside except given stop-words, and sort by word occurrence. Finally, print out Top-K words in descending order.

Input

- 1st line: the path of input file. The length of path <= 100. The count of distinct words in input file, less than 10,000. And maximal length of each word is less than 50.
- 2nd line: the path of stop-words file. Each line contain 1 stop-word. The length of path <= 100. The count of stop-words in 1st line of stop-words file. The content format of stop-words file is same as lab example.
- 3th line: K, the number of word for output (aka, Top-K).

Output

- Each output line contains: word, and it's occurrence. This 2 fields are sperated by 1 space.
- Words should exclude special characters such as `(`, `)`, `,`, `.`.
- Note that: Only output K words, even throught (K+1)th word is same occurrence with the Kth word, and then consider of sub-order by alphabet in ascending (which `strcmp` can help).

Assignment) 词语统计

OJ 初步测试
+
现场 Code Review

Format

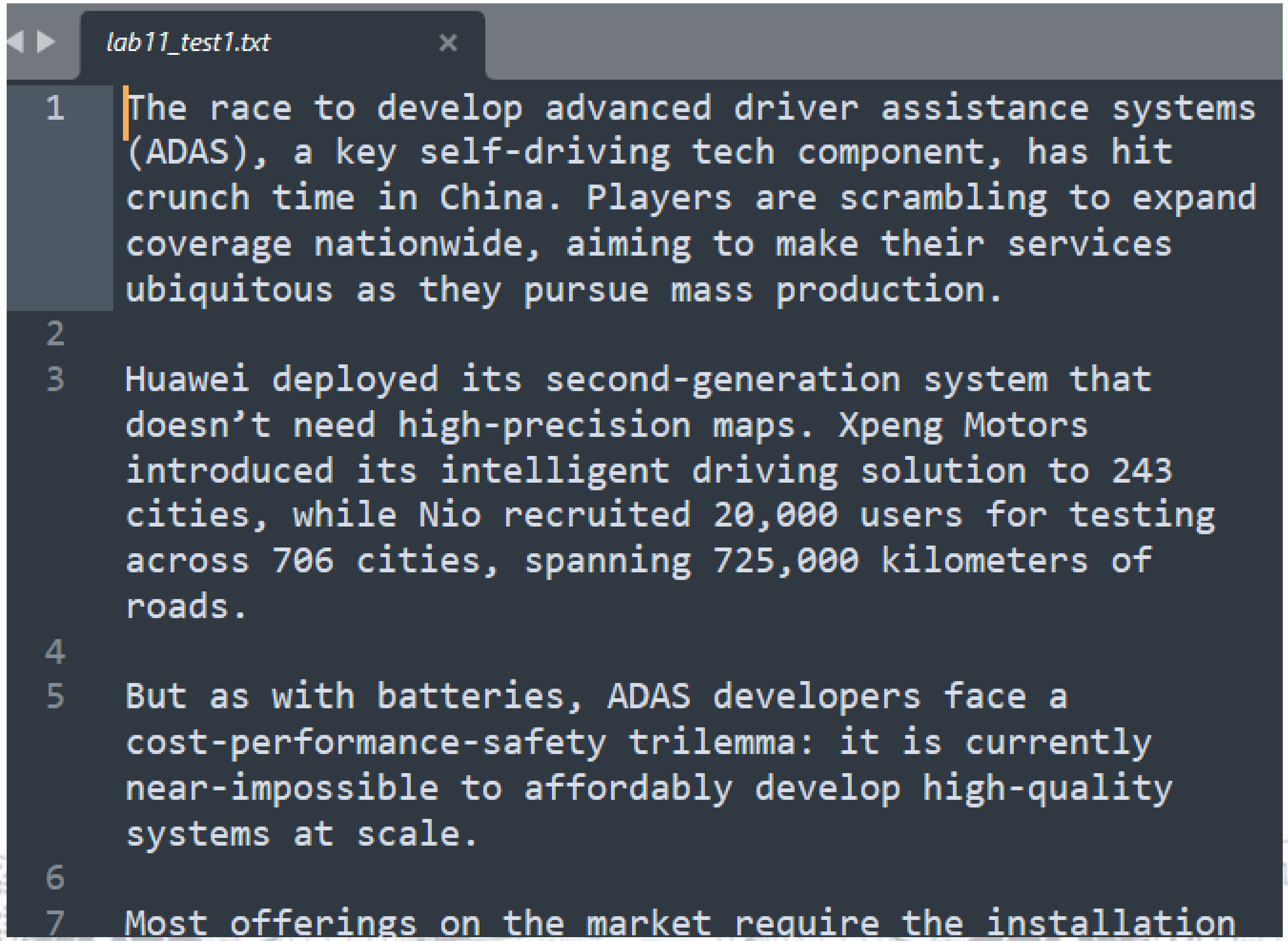
测试文件，
可在 blackboard下载，
并放到代码所在目录

Input1

```
lab11_test1.txt
stopwords.txt
5
```

Output1

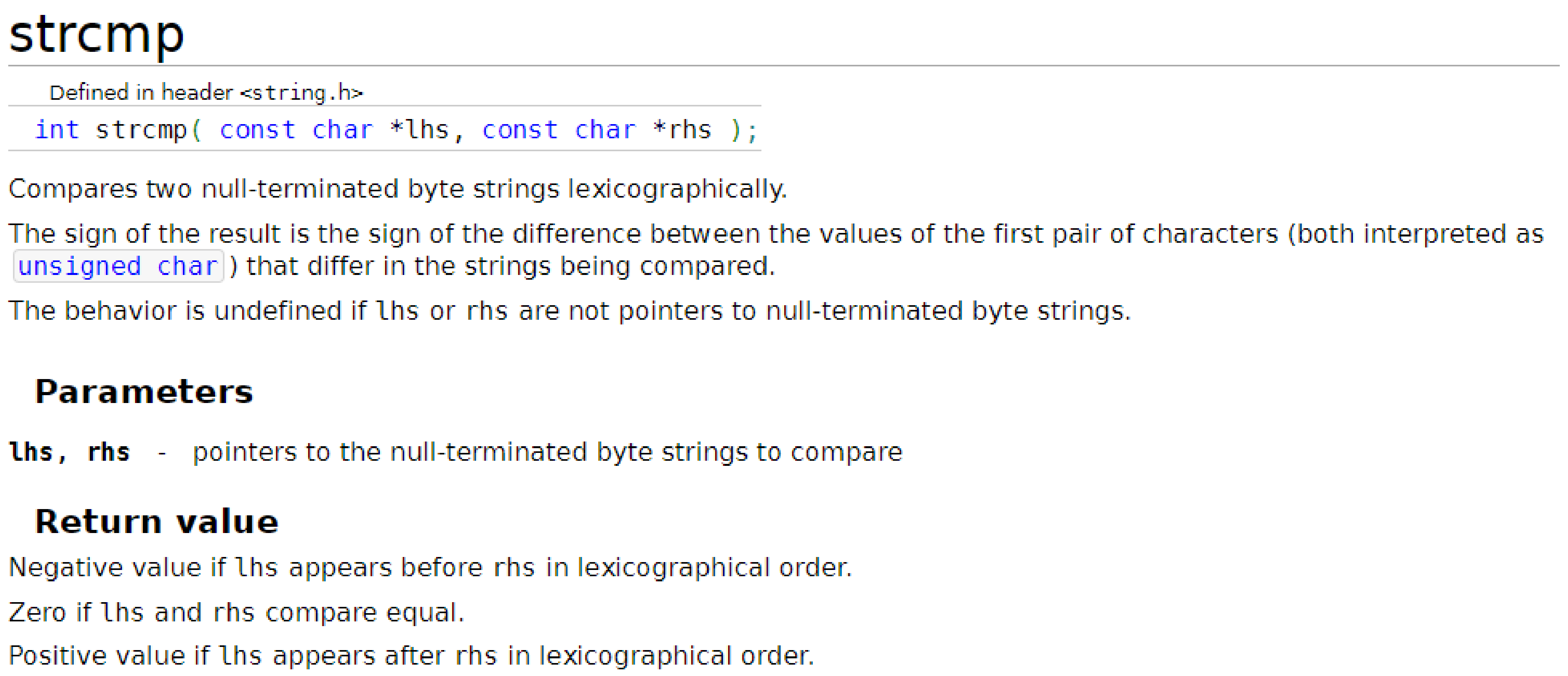
```
dji 16
lidar 10
rmb 10
cameras 7
road 7
```



lab11_test1.txt

```
The race to develop advanced driver assistance systems
(ADAS), a key self-driving tech component, has hit
crunch time in China. Players are scrambling to expand
coverage nationwide, aiming to make their services
ubiquitous as they pursue mass production.

Huawei deployed its second-generation system that
doesn’t need high-precision maps. Xpeng Motors
introduced its intelligent driving solution to 243
cities, while Nio recruited 20,000 users for testing
across 706 cities, spanning 725,000 kilometers of
roads.

But as with batteries, ADAS developers face a
cost-performance-safety trilemma: it is currently
near-impossible to affordably develop high-quality
systems at scale.

Most offerings on the market require the installation
```

Appendix, strcmp



strcmp

Defined in header `<string.h>`

```c
int strcmp( const char *lhs, const char *rhs );
```

Compares two null-terminated byte strings lexicographically.

The sign of the result is the sign of the difference between the values of the first pair of characters (both interpreted as `unsigned char`) that differ in the strings being compared.

The behavior is undefined if `lhs` or `rhs` are not pointers to null-terminated byte strings.

Parameters

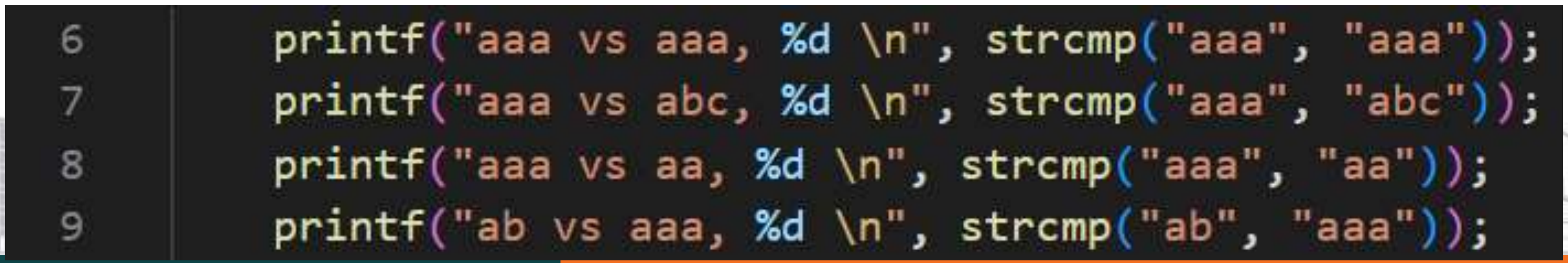
lhs, rhs - pointers to the null-terminated byte strings to compare

Return value

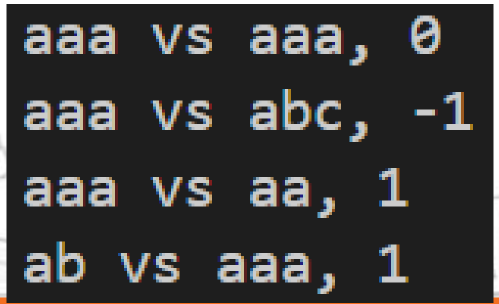
Negative value if `lhs` appears before `rhs` in lexicographical order.

Zero if `lhs` and `rhs` compare equal.

Positive value if `lhs` appears after `rhs` in lexicographical order.



```c
    printf("aaa vs aaa, %d \n", strcmp("aaa", "aaa"));
    printf("aaa vs abc, %d \n", strcmp("aaa", "abc"));
    printf("aaa vs aa, %d \n", strcmp("aaa", "aa"));
    printf("ab vs aaa, %d \n", strcmp("ab", "aaa"));
```



```
aaa vs aaa, 0
aaa vs abc, -1
aaa vs aa, 1
ab vs aaa, 1
```

SUSTech Southern University of Science and Technology

THANK YOU