# CSC209 Week 5 Notes

Hyungmo Gu

May 14, 2020

### Files in C 1 of 5

- Opening file
  - Syntax: \*fopen(const char \*filename, const char \*mode)
  - the import file should be in the same folder as 'a.out' (default)
  - Mode Strings
    - 1. r File opened for reading
    - 2. w File opened for writing
    - 3. a File opened for appending

```
#include <stdio.h>
2
      int main() {
3
          FILE *sample_file;
          sample_file = fopen("example_sources/sample.txt", "r");
          if (sample_file == NULL) {
               fprintf(stderr, "Error opening file \n");
               return 1;
          }
11
12
13
          return 0;
14
      }
15
16
```

- Closing file
  - Syntax: fclose(FILE \*filename)
  - returns 0 if close successful

```
#include <stdio.h>

int main() {
```

```
FILE *sample_file;

file *sample_file;

file *sample_file;

if (fclose(sample_file) != 0) {
    fprintf(stderr, "fclose failed\n");
    return 1;
}

return 0;

return 0;

file *sample_file;

return 0;

file *sample_file;

file *sample_file;
```

### Files in C 2 of 5

- Reading from Files
  - Syntax: char \*fgets(char \*s, int n, FILE \*stream)
  - Reads data line by line
    - 1. char \*s is a pointer to memory where text can be stored
      - \* Note new var can be created here, like for loop (i.e. for(i=0; i; 1; i++)).
      - \* On success, fgets returns s
      - \* On failure, fgets returns NULL
    - 2. int n is the maximum upper number of characters fgets allowed to put in s

```
#include <stdio.h>
1
2
      #define LINE_LENGTH 80
3
      int main() {
          FILE *sample_file;
6
          int error;
          char line[LINE_LENGTH + 1];
           sample_file = fopen("example_sources/sample.txt", "r");
10
11
           while (fgets(line, LINE_LENGTH + 1, sample_file) != NULL) {
               printf("%s", line);
13
          }
14
15
16
           return 0;
17
      }
18
```

• Reading from Input

- Syntax: fgets(line, LINE\_LENGTH + 1, stdin)
- Notice stdin is the standard input, like input in Python

```
#include <stdio.h>

#define LINE_LENGTH 80

int main() {
    char line[LINE_LENGTH + 1];

while (fgets(line, LINE_LENGTH + 1, stdin) != NULL) {
    printf("%s", line);
}

return 0;
}
```

## Files in C 3 of 5

- The scanf function
  - returns successfully read items
  - number of read items depends on format
  - Syntax: int fscanf(FILE \*stream, const char \*format, type \*s, type \*n)

```
#include <stdio.h>
1
2
      #define LINE_LENGTH 80
3
      int main() {
5
          FILE *sample_file;
6
          int error, score, total;
8
          sample_file = fopen("example_sources/sample.txt", "r");
9
          if (sample_file == NULL) {
10
               perror("Error opening file\n");
11
               return 1;
          }
13
14
          while (fscanf(sample_file, "%d %d", &score, &total) == 2) { //
     <- ==2 means each fscan must return 2 values, one for each col.
               printf("Score: %d, Total: %d.\n", score, total);
          }
17
          error = fclose(sample_file);
19
          if (error != 0) {
```

```
perror("fclose failed on input file\n");
return 1;
}

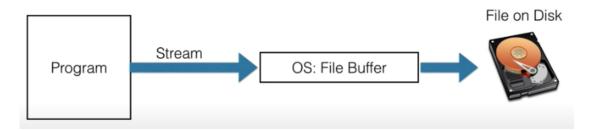
return 0;
}
```

# Files in C 4 of 5

- Writing to Files
  - Syntax: output\_file = fopen(const char \*filename, "w")
    - \* w overwrites existing file
    - \* a appends output to the end of file

```
#include <stdio.h>
          int main() {
               FILE *output_file;
4
               int error;
               int total = 50;
6
               float small_number = 0.125;
               output_file = fopen("example_sources/output.txt", "w")
               if (output_file == NULL) {
10
                   perror("Error opening file\n");
                   return 1;
12
               }
13
14
               fprintf(output_file, "The first line in the file\n");
               fprintf(output_file, "The integer is %d\n", total);
16
               fprintf(output_file, "The small float is %f\n",
17
     small_number);
18
19
               . . .
               return 0;
21
          }
22
23
```

 NOTE: the output stream is first stored in memory controlled by OS before to disk



- \* OS periodically writes content from memory to disk
- \* Abnormal computer shutdown  $\rightarrow$  some data may be lost

1 2

### Files in C 5 of 5

• Bringing Everything Together

```
#include <stdio.h>
      int main() {
3
          FILE *sample_file, *output_file;
          int error_open, error_closed, score;
          int total = 50;
6
          sample_file = fopen("example_sources/sample.txt", "r");
          if (sample_file == NULL) {
               fprintf(stderr, "Error opening file \n");
               return 1;
11
          }
12
          output_file = fopen("example_sources/output.txt", "w");
14
          if (output_file == NULL) {
               perror("Error opening file\n");
16
               return 1;
          }
18
19
          while (fscanf(sample_file, "%d %d", &score, &total) == 2) {
20
               printf("Score: %d\n", score);
21
               fprintf(output_file, "Score: %d\n", score);
22
23
          }
          error_open = fclose(sample_file);
25
          if (error_open != 0) {
26
               perror("fclose failed\n");
27
28
               return 1;
          }
29
30
          error_closed = fclose(output_file);
31
```

Listing 1: files\_example\_5.c

# Strings in C 1 of 6

- Introduction to Strings
  - String is an array of chars with 0 at the end
    - \* i.e. 'hello' =  $['h', 'e', 'l', 'l', 'o', '\setminus 0']$
    - \* without '\0', undesired output is included, i.e. hello?[BT