CSC209 Week 5 Notes

Hyungmo Gu

May 13, 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

```c
#include <stdio.h>

int main() {
    FILE *sample_file;

    sample_file = fopen("example_sources/sample.txt", "r");
    if (sample_file == NULL) {
        fprintf(stderr, "Error opening file \n");
        return 1;
    }

    ...

    return 0;
}

```

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

```c
#include <stdio.h>

int main() {
```

```
        FILE *sample_file;

        ...

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

        return 0;
    }

```

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>

#define LINE_LENGTH 80

int main() {
    FILE *sample_file;
    int error;
    char line[LINE_LENGTH + 1];

    sample_file = fopen("example_sources/sample.txt", "r");

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

    ...
    return 0;
}

```

- Reading from Input

  - **Syntax:** fgets(line, LINE_LENGTH + 1, stdin)
  - Notice *stdin* is the standard input, like input in Python

```c
#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)

```c
#include <stdio.h>

#define LINE_LENGTH 80

int main() {
    FILE *sample_file;
    int error, score, total;

    sample_file = fopen("example_sources/sample.txt", "r");
    if (sample_file == NULL) {
        perror("Error opening file\n");
        return 1;
    }

    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);
    }

    error = fclose(sample_file);
    if (error != 0) {
```

```
21              perror("fclose failed on input file\n");
22              return 1;
23          }
24
25          return 0;
26      }
27
```