

Files

File

What is a file?

- a named sequence of bytes stored on a file storage device
- The name of a file is also called its path

File Path

- Full path - unambiguous full name of a file
 - `/Users/kbhowmik/Desktop/cs110_materials/code-examples/24-fileio/files.c`
- Relative path - path relative to the current working directory
 - Working directory
 - `/Users/kbhowmik/Desktop/cs110_materials/code-examples/24-fileio/`
 - `file.c`
- Working directory
 - The current directory in the terminal
 - the directory you have open in Finder or Windows Explorer
 - Every running program has a working directory

Opening a file

```
#include <stdio.h>

int main() {
    FILE *fp = fopen("input.txt", "r");

    return 0;
}
```

Files must be opened by name before reading or writing

- `fopen(<filename>, <mode>)`
 - The filename and the mode are strings (type `const char *`)
- Modes
 - "r" – read
 - "w" – write
 - "a" – append
 - There are more, but we will just use these

Opening a file

```
#include <stdio.h>

int main() {
    FILE *fp = fopen("input.txt", "r");

    return 0;
}
```

- `fopen(<filename>, <mode>)`
 - Returns a pointer to a FILE structure
 - Returns NULL if the file could not be opened

Opening a file

```
#include <stdio.h>

int main() {
    FILE *fp = fopen("input.txt", "r");

    if (fp == NULL) {
        printf("Could not open input.txt\n");
        return 1;
    }

    return 0;
}
```

- `fopen(<filename>, <mode>)`
 - Returns a pointer to a FILE structure
 - Returns NULL if the file could not be opened

Reading from a file

```
#include <stdio.h>

int main() {
    FILE *fp = fopen("input.txt", "r");

    if (fp == NULL) {
        printf("Could not open input.txt\n");
        return 1;
    }

    int c;
    while ((c = fgetc(fp)) != EOF) {
        putchar(c);
    }

    return 0;
}
```

- Read and print each character in the input file
- `fgetc(FILE *file_pointer)` - reads a character from a file

```
int main() {  
    FILE *fp = fopen("input.txt", "r");  
  
    if (fp == NULL) {  
        printf("Could not open input.txt\n");  
        return 1;  
    }  
  
    int c;  
    while ((c = fgetc(fp)) != EOF) {  
        putchar(c);  
    }  
  
    fclose(fp);  
  
    return 0;  
}
```

- `fclose(FILE *file_pointer)` - closes a file

Writing to a file

```
#include <stdio.h>

int main() {
    FILE *fp_to_write = fopen("output.txt", "w");

    if (fp_to_write == NULL) {
        printf("Error opening output.txt\n");
        return 1;
    }

    fprintf(fp_to_write, "Hello file! How are you?\n");

    fclose(fp_to_write);

    return 0;
}
```

- If the file does not exist, `fopen()` will create the file and write the content

Writing to a file

```
#include <stdio.h>

int main() {
    FILE *fp_to_write = fopen("output.txt", "w");

    if (fp_to_write == NULL) {
        printf("Error opening output.txt\n");
        return 1;
    }

    fprintf(fp_to_write, "I hope you are well\n");

    fclose(fp_to_write);

    return 0;
}
```

- If the file already exists, its contents will be overwritten

Writing to a file

```
#include <stdio.h>

int main() {
    FILE *fp_to_write = fopen("output.txt", "w");

    if (fp_to_write == NULL) {
        printf("Error opening output.txt\n");
        return 1;
    }

    fprintf(fp_to_write, "Hello file! How are you?\n");

    fclose(fp_to_write);

    return 0;
}
```

- While a file is open in write mode, subsequent calls to `fprintf()` will keep writing additional data to the file
- Both of these lines will be written to the file.

Writing to a file

```
int main() {  
    FILE *fp_to_write = fopen("output.txt", "w");  
  
    if (fp_to_write == NULL) {  
        printf("Error opening output.txt\n");  
        return 1;  
    }  
  
    fprintf(fp_to_write, "Hello file! How are you?\n");  
    fprintf(fp_to_write, "I hope you are well\n");  
  
    fclose(fp_to_write);  
  
    return 0;  
}
```

- While a file is open in write mode, subsequent calls to `fprintf()` will keep writing additional data to the file
- Both of these lines will be written to the file.

Appending to a file

```
#include <stdio.h>

int main() {
    FILE *fp_to_append = fopen("output.txt", "a");

    if (fp_to_append == NULL) {
        printf("Error opening append.txt\n");
        return 1;
    }

    fprintf(fp_to_append, "Here is a line to append!\n");

    fclose(fp_to_append);
}
```

- The contents of the file will not be overwritten
- Writing using this file pointer will append data to the file

Reading integers from a file

```
#include <stdio.h>

int main() {
    FILE *numbers_fp = fopen("numbers.txt", "r");

    if (numbers_fp == NULL) {
        printf("Error opening numbers.txt\n");
        return 1;
    }

    int input;
    while (fscanf(numbers_fp, "%i", &input) == 1) {
        printf("%i\n", input);
    }

    fclose(numbers_fp);

    return 0;
}
```

- fscanf() and fprintf() work like scanf() and printf()
- also take a FILE pointer as the first argument

Assignment: File Integer Sum

- Your program will
 - read integers from an input file
 - write the sum of the integers to an output file

Assignment: File Integer Sum

- input.txt
 - 1 2 3 4 5 6 7
- ./file_integer_sum input.txt output.txt
- output.txt
 - 28

Exit Codes

- Incorrect number of arguments, exit code 1
- Error opening the input file, exit code 2
- Error opening the output file, exit code 3

Error messages

- Do not hardcode file names to print error messages
- Use the arguments received from command line.