File I/O

Why store data in files?

- Much larger data storage than RAM.
- Persist accross different executions of the program.
- Work with other programs.

Opening/Closing a file

```c
#include <stdlib.h>
#include <stdio.h>
/* File pointer to hold reference to our file */
FILE * fPtr;
// Open file in w (write) mode. "data/file1.txt"  is complete path to create file
fPtr = fopen("data/file1.txt", "w");
/* fopen() return NULL if last operation  was unsuccessful */
if(fPtr == NULL)
{
    /* File not created hence exit */
    printf("Unable to create file.\n");
    exit(0);
}
/* Done with this file, close file to release resource */
fclose(fPtr);
```

Reading from file

- `fgetc()` – Used to read single character from file.
- `fgets()` – Used to read string from file.
- `fscanf()` – Use this to read formatted input from file.
- `fread()` – Read block of raw bytes from file. Used to read binary files.

Reading

- Open a file using `fopen()` function and store its reference in a `FILE` pointer say `fPtr`.
- You must open file in `r` (read) mode or atleast mode that support read access.
- Read content from file using any of these functions `fgetc()`, `fgets()`, `fscanf()` or `fread()`.
  Finally, close the file using `fclose(fPtr)`.

Reading from file, char by char

```
do {
    /* Read single character from file */
    ch = fgetc(fPtr);
    /* Print character read on console */
    putchar(ch);
} while(ch != EOF); /* Repeat this
if last read character is not EOF */
```

Copying a file with source/destination as commandline arguments

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {

    if (argc != 3) {
        printf("Invalid arguments\n");
        return 0;
    }

    char ch;

    FILE* s = fopen(argv[1], "r");
    FILE* d = fopen(argv[2], "w");

    /* fopen() return NULL if last operation was unsuccessful */
    if(s == NULL || d == NULL)
    {
        /* Unable to open file hence exit */
```

Reading line by line using `fgets()`

```
char * fgets(char * str, int num, FILE * stream);
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFFER_SIZE 1000

int main() {
    /* File pointer to hold reference to our file */
    FILE * fPtr;

    char buffer[BUFFER_SIZE];
    int totalRead = 0;
    int total_chars = 0;

    /*
     * Open file in r (read) mode.
     * "data/file2.txt" is complete file path to read
     */
    fPtr = fopen("1.c", "r");

    /* fopen() return NULL if last operation was unsuccessful */
    if(fPtr == NULL)
    {
        /* Unable to open file hence exit */
        printf("Unable to open file.\n");
        printf("Please check whether file exists and you have read privilege.\n");
        return 0;
    }

    /* File open success message */
    printf("File opened successfully. Reading file contents line by line. \n\n");

    /* Repeat this until read line is not NULL */
    while(fgets(buffer, BUFFER_SIZE, fPtr) != NULL)
    {
        /* Total character read count */
        totalRead = strlen(buffer);
        total_chars += strlen(buffer);

        /* Print line read on cosole*/
        printf("%s", buffer);

    }

    printf("Total number of chars: %d", total_chars);

    /* Done with this file, close file to release resource */
    fclose(fPtr);

    return 0;
}
```

Reading/Writing Binary Data to/from a file

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Customer {
    char name[100];
    int phone_no;
} Customer;

int main() {
    Customer c = { "Ramu", 90034699 };
    FILE* cus_file = fopen("customer.bin","w");
    fwrite(&c, sizeof(Customer), 1, cus_file);
    fclose(cus_file);

    Customer d ;
    cus_file = fopen("customer.bin","r");
    fread(d, sizeof(Customer), 1, cus_file);
    printf("Customer Read Details: %s, %d", d.name, d.phone_no);
    fclose(cus_file);
    return 0;
}
```

Reading/Writing Array of structs

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Customer {
    char name[100];
    int phone_no;
} Customer;

int main() {
    Customer c[3] = {
        { "Ramu", 90034699 },
        { "Ammu", 900146939 },
        { "Thomas", 769834234 }
    };

    FILE* cus_file = fopen("customer.bin","w");
    fwrite(c, sizeof(Customer), 3, cus_file);
    fclose(cus_file);

    Customer d[3] ;
    cus_file = fopen("customer.bin","r");
    fread(&d, sizeof(Customer), 3, cus_file);
    for (int i = 0; i < 3; i++) {
        printf("Customer Read Details: %s, %d", d[i].name, d[i].phone_no);
    }
    fclose(cus_file);
    return 0;
}
```

Store Reciept Management System

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

typedef struct Customer {
    char name[100];
    int phone_no;
} Customer;

typedef enum PayMode {
    Cash,
    Card,
    UPI
} PayMode;

typedef struct Reciept {
    time_t time;
    float value;
    // Customer *customer; We cannot store this as pointer.
    int customer_index;
    PayMode mode;
} Reciept;

typedef struct Database {
    Customer customers[100];
    Reciept reciepts[1000];
    int customer_count;
    int reciept_count;
} Database;

int add_customer(char *name, int phone_no, Database *db) {
    Customer *c = &(db->customers[db->customer_count++]);
    c->phone_no = phone_no;
    strcpy(c->name, name);
    return db->customer_count-1;
}

Reciept* add_reciept(int value, int c, PayMode mode, Database *db) {
    time_t now = time(NULL);
    Reciept* r = &(db->reciepts[db->reciept_count++]);
    r->customer_index= c;
    r->value = value;
    r->time = now;
    r->mode = mode;
    return r;
}

int find_customer_by_phone_no(int phone, Database *db) {
    Customer *cust = NULL;
    for(int i =0; i< db->customer_count; i++) {
        if(phone == db->customers[i].phone_no) {
            cust = &(db->customers[i]);
            return i;
            break;
        }
    }
    return -1;
}

void print_db(Database* db) {
    printf("Customers_____________\n");
    for (int i = 0; i < db->customer_count; i++) {
        printf("%s\t%d\n", db->customers[i].name, db->customers[i].phone_no);
    }
    printf("Reciepts_____________\n");
    for (int i = 0; i < db->reciept_count; i++) {
        printf("%d\t%f\n", db->reciepts[i].customer_index, db->reciepts[i].value);
    }
}
```