File-Handling & Command-line Arguments

Course: Introduction to Programming and Data Structures

Laltu Sardar

Institute for Advancing Intelligence (IAI),
TCG Centres for Research and Education in Science and Technology (TCG Crest)

tcg crest
Inventing Harmonious Future

August 18, 2023

Basics of File Handling in C

tcg crest
Inventing Harmonious Future

fscanf and fprintf

- fscanf and fprintf works almost same as scanf and printf

```c
// Program to learn basic file operation
#include<stdio.h>

float average(float a, float b){
    return ((a+b)/2.0);
}

int main(){
    float a, b, avg;

    FILE * inp_file_ptr, * out_file_ptr; //File type pointer must be declared

    inp_file_ptr = fopen("input_file.txt","r"); // Opening input file for
        reading
    fscanf(inp_file_ptr, "%f %f", &a, &b);  // taking input from file
    fclose(inp_file_ptr);   // closing the input file

    avg = average(a, b);    //Compauting avarage

    out_file_ptr = fopen("output_file.txt","w");
    fprintf(out_file_ptr, "%f",avg); //writing on output file
    fclose(out_file_ptr); //closing the output file

    return 0;
}
```

File opening modes

• When you open a file, you need to specify the mode in which you want to open it. The following are the different file modes:

Mode	Meaning of Mode	During Inexistence of File
`r`	Reading.	If the file does not exist, `fopen()` returns `NULL`.
`w`	Writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
`a`	Append.	Data is added to the end of the file. If the file does not exist, it will be created.
`r+`	Reading and Writing.	If the file does not exist, `fopen()` returns `NULL`.
`w+`	Reading and Writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
`a+`	Reading and Appending.	If the file does not exist, it will be created.

Table: File opening modes in C

tcg crest
Inventing Harmonious Future

Reading from a file

Function	Description
`fscanf()`	Use formatted string and variable arguments list to take input from a file.
`fgets()`	Input the whole line from the file.
`fgetc()`	Reads a single character from the file.
`fgetw()`	Reads a number from a file.
`fread()`	Reads the specified bytes of data from a binary file.

Table: Some functions to Read from a file

tcg crest
Inventing Harmonious Future

Writing to a file

Function	Description
`fprintf()`	Similar to `printf()`, this function uses a formatted string and variable arguments list to print output to the file.
`fputs()`	Prints the whole line in the file and a newline at the end.
`fputc()`	Prints a single character into the file.
`fputw()`	Prints a number to the file.
`fwrite()`	This function writes the specified amount of bytes to the binary file.

Table: Some functions to Write from a file

tcg crest
Inventing Harmonious Future

Closing a file

1. The fclose() function is used to close the file
2. After successful file operations, you must always close a file **to remove it from the memory**.
3. Syntax of fclose()

```
fclose(file_pointer);
```

tcg crest
Inventing Harmonious Future

Command-line Arguments: Input from terminal before execution

tcg crest
Inventing Harmonious Future

Why inputs from command line

- Another form of input
- Useful when you want to control your program from outside.
- To override defaults and have more direct control over the application

Example:

```
1  int main(int argc, char *argv[]) {
2    /* ... */
3  }
```

or

```
1  int main(int argc, char **argv) {
2    /* ... */
3  }
```

tcg crest
Inventing Harmonious Future

```c
// Program to compute average of two float variables
#include<stdio.h>
#include<stdlib.h> //that contains atof

float average(float a, float b){
    return ((a+b)/2.0);
}
int main(int argc, char *argv[]){
    float a, b, avg;
    if (argc==3){
        a = atof(argv[1]); //converting string to float
        b = atof(argv[2]);
    }else{
        scanf("%f %f", &a, &b);  // taking input from terminal
    }
    avg = average(a, b);     //Compauting avarage
    printf("%.2f",avg); //writing on terminal
    return 0;
}
```

tcg crest
Inventing Harmonious Future

```c
// Program to compute average of two float variables
#include<stdio.h>
#include<stdlib.h> //that contains atof

float average(float a, float b){
    return ((a+b)/2.0);
}
int main(int argc, char *argv[]){
    float a, b, avg;
    if (argc==3){
        a = atof(argv[1]); //converting string to float
        b = atof(argv[2]);
    }else{
        scanf("%f %f", &a, &b);  // taking input from terminal
    }
    avg = average(a, b);     //Compauting avarage
    printf("%.2f",avg); //writing on terminal
    return 0;
}
```

- argc (ARGument Counter): is The number of command-line arguments passed. It includes the name of the program
- argv (ARGument Vector): An array of strings pointers listing all the arguments.
- argv[0] is the name of the program , After that till argv[argc-1] every element is command-line arguments.
- Only strings can be taken from command line.

tcg crest
Inventing Harmonious Future

Debugging a program in C

Debugger

Debugger

A program that runs other programs, allowing the user

- to exercise control over these programs,
- to examine variables when problems arise

GNU Debugger (GDB)

- The most popular debugger for UNIX systems to debug C and C++ programs.
- Helps you in getting information about the following:
  - If a core dump happened, then what statement or expression did the program crash on?
  - If an error occurs while executing a function, what line of the program contains the call to that function, and what are the parameters?
  - What are the values of program variables at a particular point during execution of the program?
  - What is the result of a particular expression in a program?

tcg crest
Inventing Harmonious Future

Begin with GDB

Requirement

- output executable file must be compiled with -g
  $ gcc -g -Wall demo-gdb.c -o prog.out

Starting GDB

- $ gdb ./prog.out
- $ gdb ./prog.out inp1 inp2 . . . //If it needs command line inputs

tcg crest
Inventing Harmonious Future

Most used commands in GDB

After selecting the program, we can use following commands.

- $ `start, s`
- $ `breakpoint, b`
- $ `next, n`
- $ `continue, c`
- $ `p`

Other Commands I

- `b` - Puts a breakpoint at the current line
- `b main` - Puts a breakpoint at the beginning of the program
- `b N` - Puts a breakpoint at line N
- `b +N` - Puts a breakpoint N lines down from the current line
- `b fn` - Puts a breakpoint at the beginning of function "fn"
- d N - Deletes breakpoint number N
- `info break` - list breakpoints
- `r` - Runs the program until a breakpoint or error
- `c` - Continues running the program until the next breakpoint or error
- `f` - Runs until the current function is finished
- `s` - Runs the next line of the program

tcg crest
Inventing Harmonious Future

Other Commands II

- `s N` - Runs the next N lines of the program
- `n` - Like s, but it does not step into functions
- `u N` - Runs until you get N lines in front of the current line
- `p var` - Prints the current value of the variable "var"
- `bt` - Prints a stack trace
- `u` - Goes up a level in the stack
- `d` - Goes down a level in the stack
- `q` - Quits gdb

tcg crest
Inventing Harmonious Future

Limitation of GDB

- Even though GDB can help you in finding out memory leakage related bugs, but it is not a tool to detect memory leakages.
- GDB cannot be used for programs that compile with errors and it does not help in fixing those errors.

tcg crest
Inventing Harmonious Future