File Handling in C

Storage seen so far

- All variables stored in memory
- Problem: the contents of memory are wiped out when the computer is powered off
- Example: Consider keeping students' records
  - 100 students records are added in array of structures
  - Machine is then powered off after sometime
  - When the machine is powered on, the 100 records entered earlier are all gone!
  - Have to enter again if they are needed

Solution: Files

- A named collection of data, stored in secondary storage like disk, CD-ROM, USB drives etc.
- Persistent storage, not lost when machine is powered off
- Save data in memory to files if needed (file write)
- Read data from file later whenever needed (file read)

Organization of a file

- Stored as sequence of bytes, logically contiguous
  - May not be physically contiguous on disk, but you do not need to worry about that
- The last byte of a file contains the end-of-file character (**`EOF`**), with ASCII code 1A (hex).
  - While reading a text file, the EOF character can be checked to know the end
- Two kinds of files:
  - Text : contains ASCII codes only
  - Binary : can contain non-ASCII characters
    - Example: Image, audio, video, executable, etc.
    - EOF cannot be used to check end of file

Basic operations on a file

- Open
- Read
- Write
- Close
- Mainly we want to do read or write, but a file has to be opened before read/write, and should be closed after all read/write is over

Opening a File: fopen()

- You must include <stdio.h>
- Prototype Form:
  - **FILE * fopen (const char * filename, const char * mode)**
- FILE is a structure type declared in stdio.h.
  - You don't need to worry about the details of the structure.
    - In fact it may vary from system to system.
  - fopen returns a pointer to the FILE structure type.
  - You must declare a pointer of type FILE to receive that value when it is returned.
  - Use the returned pointer in all subsequent references to that file.
  - If fopen fails, NULL is returned.
- The argument filename is the name of the file to be opened.

Example: opening file.dat for write

```
FILE *fptr;
char filename[ ]= "file2.dat";
fptr = fopen (filename,"w");
if (fptr == NULL) {
    printf (“ERROR IN FILE CREATION”);
    /* DO SOMETHING */
}
```

Opening a File

Values of mode

- Enclose in <u>double</u> quotes or pass as a string variable
- Modes:

r: open the file for reading (NULL if it doesn’t exist)

w: create for writing. destroy old if file exists

a: open for writing. create if not there. start at the end-of-file

r+: open for update (r/w). create if not there. start at the beginning.

w+: create for r/w. destroy old if there

a+: open for r/w. create if not there. start at the end-of-file

- In the text book, there are other binary modes with the letter b. They have no effect in today’s C compilers.

stdin, stdout, and stderr

- Every C program has three files opened for them at start-up: stdin, stdout, and stderr
- stdin is opened for reading, while stdout and stderr are opened for writing
- They can be used wherever a FILE * can be used.
- Examples:
  - fprintf(stdout, "Hello there!\n");
    - This is the same as printf("Hello there!\n");
  - fscanf(stdin, "%d", &int_var);
    - This is the same as scanf("%d", &int_var);
  - fprintf(stderr, "An error has occurred!\n");
    - This is useful to report errors to standard error - it flushes output as well, so this is really good for debugging!

The exit() function

void exit(int status);

- Sometimes error checking means we want an emergency exit from a program
- Can be done by the exit() function
- The exit() function, called from anywhere in your C program, will terminate the program at once

Usage of exit( )

Example:

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

……
if( (fp=fopen("a.txt","r")) == NULL){
    fprintf(stderr, "Cannot open file a.txt!\n");
    exit(1);
}
```

Closing a file

- int fclose(FILE *stream)
- This method returns zero if the stream is successfully closed. On failure, EOF is returned
- Should close a file when no more read/write to a file is needed in the rest of the program
- File is closed using fclose() and the file pointer

```
FILE *fptr;
char filename[]= "myfile.dat";
fptr = fopen (filename,"w");
fprintf (fptr,"Hello World of filing!\n");
…. Any more read/write to myfile.dat….
fclose (fptr);
```

An example:

```
#include <stdio.h>
int main()
{
   FILE *fp;
  fp= fopen("tmp.txt", "w");
  if (fp == NULL)
  {
    printf (“ERROR IN FILE CREATION”);
   /* Do something */
   exit(1);
  }
  fprintf(fp,"This is a test\n");
  fclose(fp);
  return 0;
}
```