CS232L Operating Systems Lab
Lab 07: Introduction to C Programming

CS Program
Habib University

Fall 2019

1 Introduction

In this lab you will learn how to:

1. do File I/O in C on Linux

2 File I/O

The basic flow to do file i/o is quite simple. You:

- open a file
- manipulate the file
- close the file

Don't forget the last step as not closing a file would result in resource leaks.

In Linux opened file are represented as streams represented by `FILE *` objects.
The `fopen()` function would take a path to a file and a mode and return a pointer to a stream object which we can then pass to other functions. The mode argument tells the `fopen` whether we want open the file in read mode, write mode, or both.

```c
/*hello_fopen.c*/
#include <stdio.h>
#include <stdlib.h>

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

  FILE * stream = fopen("helloworld.txt", "w");

  fprintf(stream, "Hello World!\n");

  fclose(stream);
}
```

Listing 1: hello_fopen.c

In the above listing, the `fprintf()` function uses the stream pointer returned by the `fopen()` to write to the opened file. `fclose()` then close the file stream that had been opened by the `fopen()`.
When a process is created, the OS will open three stream by default: stdin, stdout, and stderr.

`fprintf()` and `fscanf()` can be used to do formatted output and input on opened file streams. The function `printf()` is a wrapper for `fprintf` with the first argument pointing to stdout. Similarly `fscanf()` for `scanf()` with the input stream set to stdin.

```c
/*file_input.c*/
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char * argv){

  FILE * stream = fopen("students.dat", "r");

  char iD[1024],lname[1024];
  int a;
  float b;
  char c;

  while ( fscanf(stream,
                 "%s %s %d %f %c",
                 iD, lname, &a, &b, &c) != EOF){

    printf("ID: %s\n",iD);
    printf( "Name: %s\n",lname);
    printf("marks: %d\n",a);
    printf("gpa: %f\n",b);
    printf("grade: %c\n",c);
    printf("\n");
  }

  fclose(stream);
  return 0;
}
```

Listing 2: file_input.c

The EOF character is a special character that signifies the end of file. `fscanf()` would return EOF when there's no more data in the file.

2.1 Todo: copy

1. read about the different values that could be passed to the `mode` parameter of `fopen()` and experiment with them.
2. modify the above listing so that every time it runs, it should append to the file `helloworld.txt`.
3. write a program that imitates the linux `head` command i.e., display the first 10 lines of the file passed on its command line argument.
4. write a program that imitates the linux `cp` command i.e., copies the contents of one file to the another by overwriting the destination file. It should create the destination file if it doesn't exist.

2.2 Todo: fseek

For every opened stream, the GNU C library keeps track of the position where you currenltly are in that stream.

1. read about the `fseek(), ftell()` functions and use them to writ at the start, middle, and end of an opened stream. Observe the effects by checking the contents of the written file.
2. using `fseek()` try writing beyond the end of a file; like a million bytes after the end of the file. Observe the resulting file size with the `ls -l` command.

3 File Descriptors

Linux also provides a lower level set of functions that work with file descriptors instead of file streams. The counter part to above used functions are: `open, close, read, write, lseek`. Read the man pages of these functions for further information.

You'll see that these functions do not do formatted I/O. You can take a look at the `sprintf()` function to format a string before outputting it.

3.1 todo: descriptors

Repeat the above exercises, this time using the file descriptors.