

# File I/O

---

## Lecture Notes

### Overview

- A file is a block of (usually contiguous) data used for storing information
- Usually stored on some sort of durable storage
- In Unix: everything is a file (buffers, directories, files, sockets, etc.)
- Files may be plain text (ASCII), binary, formatted, etc.
- In general, file processing involves:
  - Opening (or creating) a file
  - Reading or writing to a file
  - Closing the file
- Binary vs Plaintext

### File I/O in C

1. Introduction
  - a. EOF: End of File flag (every file has one)
  - b. `stdout`, `stdin`, `stderr` are all files
2. Basics
  - a. `FILE`: built-in type for files
  - b. Best to use pointers:

```
FILE *inFile = NULL;
FILE *outFile = NULL;
```
  - c. All file functions are done with file pointers
  - d. Files are buffers or streams: a file pointer points to a part of a file
3. Opening Files
  - a. File Open

```
FILE *fopen(const char *path, const char *mode)
```
  - b. First argument: file to open
  - c. Second argument: mode
    - `r` – reading
    - `r+` - reading and writing
    - `w` – writing (truncates if already exists)
    - `w+` - reading and writing (truncates)
    - `a` - appending
  - d. Returns a pointer to the file, initially points to the beginning, as we read/write, pointer “scans” through the file

- i. May return NULL upon failure
  - e. Misc
    - i. Permissions
    - ii. Absolute paths
    - iii. Relative paths
- 4. Input
  - a. File Scan Formatted:
 

```
int fscanf(FILE *stream, const char *format, ... );
```

    - i. Returns an int (number of matched placeholders) or EOF
  - b. File Get String:
 

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

    - Reads in at most (num - 1) from stream into str
    - Stops after the EOF or endline character is read
    - If endline is read, it is *preserved* in str
    - Null terminating character is automatically inserted
    - Returns str on success, NULL on error or EOF
  - c. Binary input:
 

```
fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
```
- 5. Output
  - a. File Print Formatted:
 

```
int fprintf(FILE *stream, const char *format, ... );
```
  - b. Binary output:
 

```
fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)
```
- 6. File Closing:
 

```
fclose(FILE*)
```

## File I/O in Java

1. Inputs/Outputs: BufferedReader, BufferedWriter, InputStream, OutputStream, etc.
2. Better solution: Scanner
  - a. Standard Input: Scanner s = new Scanner(System.in);
  - b. File input: Scanner s = new Scanner(new File("filename"));
  - c. s.next() (String), s.nextInt(), s.nextDouble()
  - d. By default: token is whitespace (can set using s.useDelimiter(":"); accepts any regular expression)
3. File Output
  - a. Buffered Text Data:
 

```
File f = new File("name");
BufferedWriter bw = new BufferedWriter(new FileWriter(f));
bw.write(..); //string or character data
```
  - b. Raw Data:
 

```
FileOutputStream fos = new FileOutputStream(new
```

```
File("name"));
fos.write(..); //must be pure bytes
```

- c. Non-buffered Text (easier, but fails silently):

```
PrintWriter pw = new PrintWriter(new File("name"));
can use print (overloaded) or printf
```

#### 4. Formatting

- a. String.format: varargs
- b. java.util.Formatter

## Exercises

1. Demonstrate the difference between binary and plaintext files by saving the same data in both formats to different files. Observe the contents and their sizes.
2. Read in a file containing album information and modify it in some manner. Output the data formatted in the following formats:
  - a. XML
  - b. JSON (see <http://json.org/>)
  - c. HTML Table
3. Open the unix system's `/etc/passwd` file, which contains data about users, parse each line and find a specific user (login); then parse the rest of the line and print that user's data.
4. Open a file containing Cartesian coordinates, one (x, y) pair to a line and determine which pair of points is the closest to each other.
5. Open an HTML file and extract all paragraph (`<p>...</p>`) elements and print them to the standard output.
6. Open a file containing data for a bridge hand (13 cards to 4 players) and determine if it's a legitimate deal