



The Ins and Outs of  
**Files**

CS2263 – Systems Software Development



## References

- Lu, Yung-Hsiang. 2015. Intermediate C Programming. CRC Press. New York. (Chapter 10, 11)



# Lecture Learning Outcomes

At the conclusion of this presentation students should be able to:

- Explain the general characteristics of files
- List two general types of files
- Explain the similarities of using file functions of `<stdio.h>` and OOP objects
- Program using character-based (ASCII) file I/O



# What is a File?

<https://old-hit.lib.unb.ca/welcome/img/memories/o35.jpg>



<https://old-hit.lib.unb.ca/welcome/img/memories/o33.jpg>



# Files: Data at Rest

Files are just arrays on an external device

- Numeric, character data that belong together
- Have a similarity to arrays
- Can be found on storage devices
  - Disk
  - Memory stick
- Transported over a network
- Can also come from a keyboard, or to a display/terminal.
- Data in motion is a *stream*



# Before We Start

- Every process when constructed has three streams automatically initialized:
  1. Standard in (keyboard): `stdin`
  2. Standard out (terminal): `stdout`
  3. Standard error (terminal): `stderr`
- You know this intuitively from Java



# File Types

- Text/ASCII
  - All data is stored as characters, even numbers
- Binary
  - All data is stored in its native format
    - ASCII codes (integers)
    - 2s complement integers
    - IEEE754 floating point values
- There are (of course) specific file formats for how data is ordered, represented within each of these format types



# File Anatomy

- Similar to a string
  - Data with and end character
  - EOF, defined in C within `<stdio.h>`
  - Not definable within ASCII – Why?
- `arrayPosition.c`, `filePosition.c`





# Files, in C

- In C, a file is represented via the data type `FILE*`
- Every function that operates on a file has a `FILE*` parameter
- File Level
  - `fopen()`
  - `fclose()`
- Character I/O
  - `fgetc()`
  - `fputc()`
  - `ungetc()`
  - `fgets()`
  - `fputs()`
- Formatted I/O
  - `fprintf()`
  - `fscanf()`
- File Positioning
  - `fseek()`
  - `ftell()`
  - `rewind()`
- `backwards.c`



## File Patterns: File-Level

- Open the file or die
- Use it
- Close the file

```
/* open the file or die */  
pFIn = fopen(argv[1], "r");  
if(pFIn == (FILE*)NULL){  
    fprintf(stderr, "Unable to open file (%s)\n", argv[1]);  
    return EXIT_FAILURE;  
}
```

```
/*  
 * Done - close up!  
 */  
fclose(pFIn);
```



# File Patterns: Processing

- Read
- While not done
  - Do stuff
  - Read again
- readPoints.c

```
iErr = fscanf(pFIn, "%f %f", &x2, &y2);
while(iErr == 2){
    //Do stuff here

    iNRead = fscanf(pFIn, "%f %f", &x2, &y2);
}
```

