

# Binary File IO

CS2263 – Systems Software Development

*Real Gone, Sheryl Crow (Theme from the movie "Cars") <https://www.youtube.com/watch?v=6zVjmoUGfv8>*

1

## Learning Outcomes

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

- Recognize ASCII text in its binary format
- Write and read simple data types' binary data to/from files
- Write and read struct data types' binary data to/from files
- Discuss to advantages of using binary files over text files.

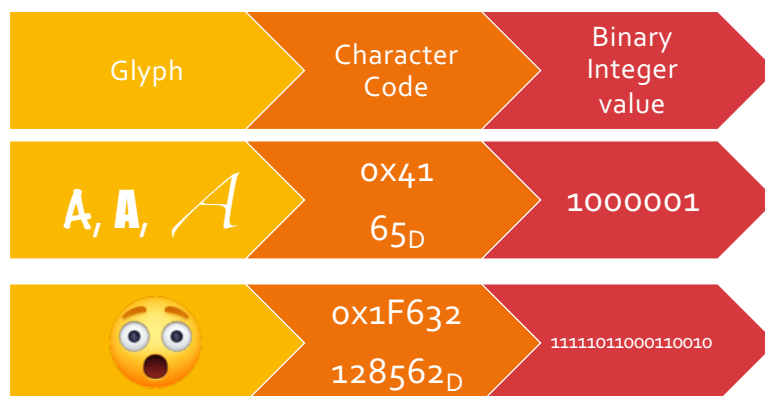
2

## References

- Lu, Yung-Hsiang. 2015. Intermediate C Programming. CRC Press. New York.
  - Chapter 10.
- Kernighan, BW and DM Ritchie. 1988. The C Programming Language. Prentice-Hall.
  - Appendix B

3

## What is a Character?



4

## What is a File?

- A collection of bytes on a storage device, terminated by an EOF value
- What are in the bytes? If we looked closely, what would we see?



5

## ASCII versus Binary

- Not really the correct terminology, but...
- Most common tools for viewing files (that you're used to) interpret the bits as text (ASCII) characters and display the associated glyph
  - Text editor
 

```
$ more Strings.c
```
  - `getchar()`, `fgetc()`, `scanf("%c", &c)`
- Binary files on the other-hand require an understanding of the specific data type for each byte
  - `int`, `double`, `char`, `float`
  - `unsigned int`, `unsigned long int` ...
  - Why not `int*`, `double*`, etcetera?
  - Why not structs?
- Each binary file will require an in-depth understanding of it's sequence of data types.

6

So.  
Much.  
Work.  
  
One.

- Yes. Or at least maybe.
- ASCII (text) files: luxury.
- Consider int values
  - A three-digit value, plus the space to end the value (in characters) takes up the same space as an `int` (4 bytes)
  - Any text representation of a value larger than 999 (or smaller than -99) take more than a binary representation of the value

7

So.  
Much.  
Work.  
  
Two.

- Consider the conversion process
  - WAT?
- Computer Science assignment
  - "Read in an ASCII representation of an integer value as a character string (e.g. `getchar()`). Convert it to a 2s complement binary value and store it in an `int` variable, and print it out (e.g. `printf("%d", ...)`)"
  - What's the computing effort to do this?
- This is what `scanf()` does under a `%d` format specifier

8

## Binary I/O

- Identify the filetype as binary:
  - `fopen("filename", "rb")`
  - `fopen("filename", "wb")`
- Use binary read and write functions:
  - `size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)`
  - `size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)`

9

## "So What Are We Going to do Today Brain?"

<https://www.youtube.com/watch?v=mYvAYwpUDv8>

- Or maybe, "What we try and do every night, Pinky. Understand some C". Except that it's not peculiar to C.
- What are the speed/storage differences between storing values as text or in binary?
- We will:
  - Generate large quantities of data
  - Examine the resulting file sizes
  - Examine the differences in execution time.

10

## Today's Code Tour

- Point.h, Point.c
- genPointsTxt – generates  $n$  points and writes them to stdout as text, one at a time
- genPointsBin - generates  $n$  points and writes them to stdout as binary, one at a time
- genPointsBlob - generates  $n$  points and writes them to stdout as binary, as a single write
- pointsTxt2Bin – reads text points from a file, writes binary to a file
- pointsBin2Txt – reads binary points from a file, writes text to stdout
- pointsSeekBin – reads specific binary values from a file (random access)

```
Still questioning the value of make?  
$ unzip L19src.zip  
$ cd !*  
$ make install  
Your executables are in ./bin
```