Pros and Cons of Storing Binary Data

Advantages of storing/transmitting binary data
- **avoid converting** to/from ASCII, and
- **use less space** on disk.

Disadvantages of storing binary data
- **not human-readable**,
- **not portable** (endianness, floating-point variations, and so forth),
- **more difficult to** manage **upgrades**.

Note: **need to flatten** data structures **regardless**.

Use `fread` to Read Binary Input from a Stream

To read binary input from a stream:

```
size_t fread (void* ptr,
    size_t size, size_t n_elt,
    FILE* stream);
```

- `ptr` is **address to which data are stored**
- `size` is the **size of one "thing"**
- `n_elt` is the **number of "things"**
- `stream` is the **stream from which to read**
- **returns number of "things" read** or **0 on failure** (or 0 requested)

Use `fwrite` to Write Binary Output to a Stream

To write binary output to a stream:

```
size_t fwrite (const void* ptr,
    size_t size, size_t n_elt,
    FILE* stream);
```

- `ptr` is **address from which data are written**
- `size` is the **size of one "thing"**
- `n_elt` is the **number of "things"**
- `stream` is the **stream to which to write**
- **returns number of "things" written** or **0 on failure** (or 0 requested)

Use `fgets` & String "I/O" to Parse Human-Readable Files

Humans are error prone.

Writing **error-handling and variation-handling code**
- for human-readable files
- is **challenging with `fscanf`**.

Instead, one can
- **use `fgets`** to read each line into a string, then
- **use string "I/O"** to parse the string.
- Failures can be re-parsed in different ways,
- and failed lines can be echoed to the human.

