Lessons Learned from Maintenance of Scientific File Formats

Arnaud Desitter

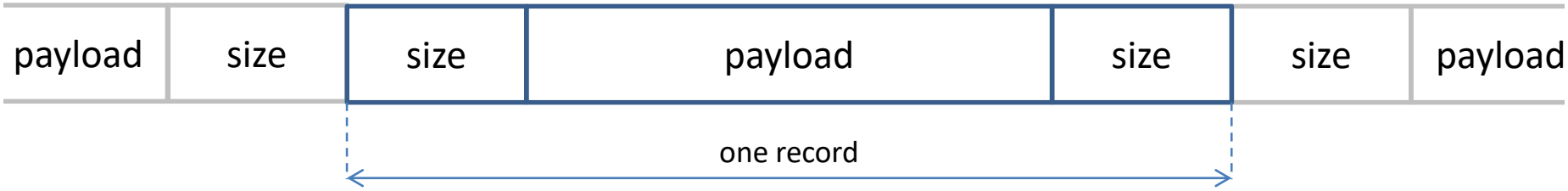
June 2014

Do not over-specify

- Fortran “unformatted” data
- Used in many major scientific systems
- Format invented circa 1978 by Stuart Feldman (author of “make”) when he implemented the first Fortran 77 compiler.
- The format has “scaled” ever since

How?

- A succession of “binary” records prefixed and suffixed by the payload size as a 32bit signed integer
- Little-endian vs Big-endian
- **32bit** signed integer
  => cannot handle more than 2 GiB per record



Ideally, a good format extension must:

1. Be backward compatible
2. Likely to fail in a clear way with software that predate the extension

Not much to play with:

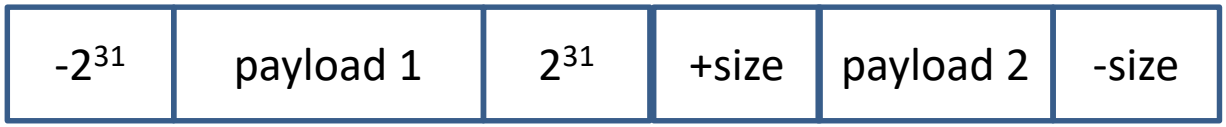
A succession of “binary” records prefixed and suffixed by the payload size as a 32bit signed integer

Not much to play with:

A succession of “binary” records prefixed and suffixed by the payload size as a 32bit **signed** integer

The bit sign is unused!

Long records are split in sub-records. The sign bit of the **prefix** indicates whether the record is continued or not. The sign bit of the **suffix** indicates the presence of a preceding record.



Lessons learned

- Do not over-specify
  - Unused bits are your scope for extensions
  - Make them reserved
    - x64 architecture made possible thanks to reserved unused bits in x86 architecture
  - Think of compounding information
    - UTF8, x86 instructions

Do not under-specify

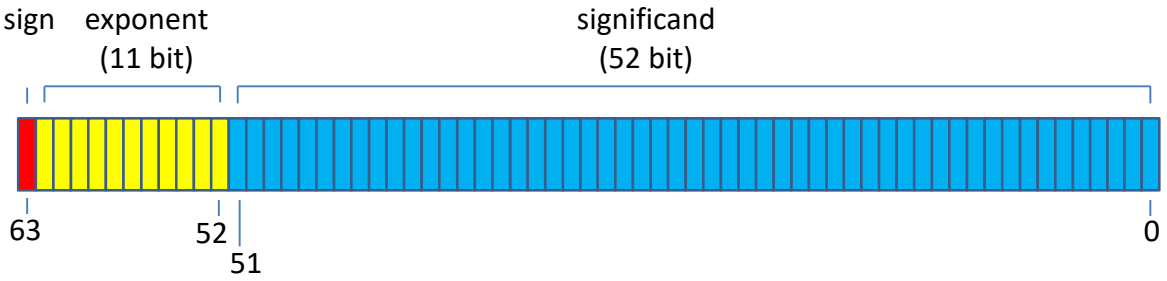
IEEE 754 – 1985 “**IEEE Standard for Binary Floating-Point Arithmetic**”

This standard is arguably the most important in the computer industry, the result of an unprecedented cooperation between academic computer scientists and the cutting edge of the industry

-- *Michael L. Overton*, Numerical Computing with IEEE Floating Point Arithmetic,
SIAM, 2004

NaN encoding

binary64:



- Sign: 0 or 1- this is irrelevant to NaN.
  Exponent: set to 1
  Significand: at least one non 0 bit. This may be used for a payload.
  (If the significand is zero, this represents infinities)

- IEEE 754-1985 did not specify how Quiet NaN and Signalling NaN are tagged
- For most processors bit 51 is “is_quiet”
- For PA-RISC and MIPS, bit 51 is “is_signalling”
- IEEE 754-2008 recommends that the most significant bit is “is_quiet” for binary formats (section 6.2.1)

MIPS architecture has been altered to support the “is_quiet” NaN.

(Revision 5.03, Sept. 9, 2013 - https://sourceware.org/ml/newlib/2014/msg00086.html)

Maintenance of two build chains

Unhappy developers

Look for “**[RFC, MIPS] Relax NaN rules”** in gcc mailing list

Lessons learned

- Do not under-specify where there is a source of incompatibility

  Flexibility does not help in this case

Lessons learned

- **Do not over-specify**
  - Unused bits are your scope for extensions
  - Make them reserved
    - x64 architecture made possible thanks to reserved unused bits in x86 architecture
  - Think of compounding information
    - UTF8, x86 instructions
- **Do not under-specify** where there is a source of incompatibility
  - Flexibility does not help in this case