2. Data Formats

Fundamentals of Data Management

Dr Rob Baxter
Software Development Group Manager, EPCC
r.baxter@epcc.ed.ac.uk
+44 131 651 3579 | +44 7971 437749

Lecture outline

- What do we mean by “data formats”?
- What’s the difference between “encoding” and “format”?
- Files and file systems
- Some commonly-used scientific file formats
- After completing this lesson, you should be able to:
  - Grasp the basics of how digital data are stored
  - Name the most common standard file formats in use in HPC
  - Understand why using standard file formats is A Good Thing
  - Understand a little about data recovery

What does digital data look like?

100101010010101001010101010101010001011101010110101010101010101010001010111010
101011111010110101101011011101010101011010101011010101010101101010101010101011
010111010110111101010101010110101010101110101011101001011101011101011110101
0010101001010100101010101010100010111010101101010101010101010100010101110101
0101111010111010110101101110101010110101010110101010101011010101010101010110
101110101011111101010101011010101010111010101110100101110101110101111101010
0101010010101001010101001010101110101011010101010101010101000101011101010
1011110101110101101011010111010101010101011010101010101101010101010101101
01110101011111101010101010111010101011010101110100101110101110101111010100
101010010101001010101010101000101110101010010101010101010001010111010101
0111101011101011010110111010101010110101010110101010101101010101011010
1110101011111101010101010110101010101110101010101010111010111101011110101001
010100101010010101010101010100010111010101101010101010101010100010101110101010
111101011101011010110111010101010110101010110101010101010110101010101110101
11010101111110101010101011010101010111010101110100101110101110101111101010010
1010010101001010101010101000101110101011010101010101010101000101011101010101
111010111010110101101110101010101101010101101010101011010101010101010111011
1010101111110101010101011010101010111010101110100101110101110101111010111100

You get the picture

Meaning from bits

- The meaning of digital data is captured at (at least) three levels:
  - How to record numbers and characters as groups of bits (*encoding*)
  - How to interpret the arrangement of the numbers or characters (*file format*)
  - How to collect bits into groups with meaning (*files, records*)

Meaning from bits: encoding

- Interpreting groups of bits = encoding
- Standard encoding formats for fixed bit lengths
  - e.g. 8 bits = 1 byte = 1 ASCII character
- Custom-specified “binary formats”
  - e.g. “the next 8,192 bits are 128 64-bit IEEE floating-point numbers”
- Encoding is at a character or number level
- Computer systems typically work with bytes as a base unit
  - e.g. “4-byte integer” rather than “32-bit integer”
  - Traditionally: 2 bytes = 1 word; 4 bytes = 1 long-word

Endianness

- "Endianness" refers to two different ways of ordering the bytes that make up a word:
  - little-endian: store the least significant byte first
  - big-endian: store the most significant byte first
    - (where "first" = "in the smallest memory address")
- e.g. the decimal integer 8,762,359
  - In hexadecimal: 85B3F7 or 00 85 B3 F7 in 4 1-byte groups
  - Layout in memory (or on disk!) will be

memory address	101	102	103	104
big-endian	00	85	B3	F7
little-endian	F7	B3	85	00

Endianness

- Big-endian is more intuitive
  - cf. decimal positional notation: thousands-hundreds-tens-units
- Alas, the world's most prevalent microprocessor architecture (Intel x86/x64) is little-endian
- Most networking protocols (esp. IPv6) are big-endian
- Endianness can be a problem for binary file portability across architectures
  - e.g. Intel x86-created files on IBM z/series mainframes
  - although modern compilers provide good support
  - and standard i/o libraries for standard file formats deal with it for you

Unicode

- "*Unicode is a computing industry standard for the consistent encoding, representation and handling of text expressed in most of the world's writing systems*." (Wikipedia)
- Defines character encodings (Unicode Transformation Formats, UTFs) and more
- Most text (certainly on the Web) is encoded using Unicode
  - UTF-8, 8-bit variable-width encoding = ASCII = most HTML
    - used by default in most Unix-like systems
  - UTF-16, 16-bit variable-width encoding
    - used by default in modern MS Windows systems
    - has a Byte Order Mark (BOM), U+FEFF, to help with endianness

Floating point formats

$$\boxed{a}\ \ 0\cdot\boxed{m}\ \ \boxed{m}\ \ \boxed{m}\ \ \boxed{m}\ \times 10^{\boxed{e}}$$

- Computer storage of real numbers is directly analogous to scientific notation
  - (sign *a*) 0.(mantissa *m*) × 10 $^{(\text{exponent } e)}$
  - except using binary representation not decimal
  - ... with a few subtleties regarding sign of *m* and *e*
- IEEE 754 floating point standard is most widely used:
  - 32bit single precision: 1bit sign, 8bit exponent, 23bit mantissa
  - 64bit double precision: 1bit sign, 11bit exponent, 52bit mantissa

1	1001010	10101000101000000101
sign	exponent	mantissa

More complex encodings

- EBCDIC: Extended Binary Coded Decimal Interchange Code
  - an 8-bit character encoding, used mainly on IBM mainframe operating systems, with no clean mapping to ASCII/UTF-8…

- Compression formats (algorithm specific):
  - zip, jpeg, mpeg…
  - Typically a combination of file format and encoding

- Encryption tools and formats (algorithm specific):
  - mcrypt, BitLocker, TrueCrypt, AES, Blowfish…

Meaning from bits: format

- Interpreting decoded bit-groups (bytes, characters) = format
- Using standard file formats
  - netCDF, XML, HDF5, PDF, CSV, JSON, GRIB, proprietary formats
- Or using custom-specified file formats
  - "A header of 8 4-byte integers, with the rest of the file 64-bit floating point"
- You can think of formats as being at a file or database level

- Before we go on: what is a *file*?

Files & file systems

- In most computer systems, a *file* is the basic unit of data organisation
  - Essentially, a *pointer* to a sequence of bits, and a (small) collection of descriptive information (*metadata*) about that sequence
- The nature of the pointer and the metadata are determined by the *file system*
  - A framework for creating, manipulating, deleting & keeping track of files
  - In *POSIX-compliant* file systems, {*pointer* + *metadata*} = *inode*
- The file system overlays the actual data stored on disk / tape / flash / CD / whatever
  - It's a logical view of the data and their organisation
    - and typically involves a big look-up table of inodes
  - Contrast with *block device*

Common native file systems

- "Desktop"
  - Windows: FAT, NTFS
  - MacOS: HFS, HFS+
  - Linux: ext2, ext3, ext4
- "Network" or "distributed"
  - Linux/Unix: NFS
  - Windows: SMB/CIFS
  - MacOS: AFP

- Modern computer systems support a wide range of file systems as well as their "native" ones
- FAT support is widespread (eg. USB devices)
- As are NFS and SMB

- "HPC"
  - Linux-based systems: ext3, ext4 + NFS
  - GPFS: IBM's proprietary high-performance parallel file system
  - Lustre: open source high-performance parallel file system
  - HDFS: open source clone of Google's distributed file system

FORTRAN binary files: a warning!

- FORTRAN-created binary files are “special”
  - FORTRAN i/o is *record-based*
- A single FORTRAN write statement – **`write(1,*) array`** – writes this into the associated file:

  `<number of bytes in array><array><number of bytes in array>`
  - i.e. the binary data in array is written as a *record*, bracketed by its *size in bytes*
  - the number *size* may itself be 4 to 12 bytes in length
    - it’s implementation dependent!
- Thus FORTRAN-created binary files are distinctly unportable
  - and thus to be avoided!

File format good practice

- In general, don't invent your own formats!
- Be aware of standards used in your research area
- Be aware of general standards (*open if possible*)
- Use standard formats, standard i/o libraries
- If you do need something custom, at least adopt a standard "framework"!
  - CSV + UTF-8
  - XML (see later lectures)
  - key-value pairs (cf. JSON)

- Let's take a quick look at some important scientific formats

CSV

- Comma Separated Value
  - The simplest “standard” file format!
- A trivial way to write tabular (row-column) data:
  - rows are separated by newlines
  - columns are delimited by commas
- Assumes a character encoding like ASCII or UTF
- Not standard, but described in a standard way in RFC 4180
- http://tools.ietf.org/html/rfc4180

HDF5

- Hierarchical Data Format v5
  - "a data model, library, and file format for storing and managing data"
  - originally developed at the National Center for Supercomputing Applications (NCSA) at the University of Illinois
- Data model based on *groups* and *datasets*
  - can think of groups like directories/folders, datasets like files
  - both can have (user-defined) *attributes*
- HDF5 files are binary but portable
  - HDF5 model take care of types, endianness etc.
- Rich API library (in C) with bindings for FORTAN, Java
  - and plenty of example code / data snippets
- http://www.hdfgroup.org/HDF5/

CDF

- NASA Common Data Format
  - “a conceptual data abstraction for storing, manipulating, and accessing multidimensional data sets”
  - developed by NASA Goddard Space Flight Center
- Data model quite similar to HDF5
  - both have *datasets* and *attributes*; CDF does not have *groups*
- CDF files are binary and portable
  - similar to HDF5
- The CDF library has C, Java and FORTRAN APIs
- http://cdf.gsfc.nasa.gov/

CDF is not to be confused with…

netCDF

- Network Common Data Format
  - "a machine-independent format for representing scientific data"
  - developed by Unidata at UCAR (US University Corporation for Atmospheric Research)
  - completely incompatible with CDF ☺
- Two data models
  - "classic", representing most netCDF files found today
  - "enhanced" or "netCDF-4", which layers netCDF on top of HDF5
- Again, netCDF is a portable binary format with a rich set of libraries
  - for C, FORTRAN, Java
- http://www.unidata.ucar.edu/software/netcdf/

Discipline-specific formats

- GRIB: GRIdded Binary
  - World-wide standard for weather/meteorological data
- FITS: Flexible Image Transport System
  - Widely used open standard for astronomy data
  - Designed specifically for scientific images: comprehensive metadata
- CFD GNS: CFD General Notation System
  - Common open standard in computational fluid dynamics simulation
  - Supported by CFX, Fluent, Star-CD & others; OpenFOAM converters
- DICOM: Digital Imaging and Communications in Medicine
  - Semi-proprietary medical imaging standard (scanner manufacturers)
  - Covers not only file format but comms protocols and more
- All come with i/o library support

Digital forensics

- Use of tools to recover digital data from files or disk images
  - for disaster recovery
  - for investigation or law enforcement purposes
- Most work on the principle that deleting files doesn't actually delete data
  - "delete" by default usually just removes the link to the data from the inode table (*unlinks* the file's inode)
  - but the bit sequence remains on disk – unless it's later overwritten
    - (and no forensics firm will offer to recover overwritten data)
- Digital forensics is about recovering the original meaning of files from isolated bit streams

Simple forensic file tools: file

- Determines file type by reading file headers
  - Works for many binary / text (ASCII) file types
  - Doesn't rely on the '`.xyz`' filename extension ☺
  - Uses a look-up table (defaults to `/etc/magic` on Unix-like)

```
me@solaris$ file s_factor.o
s_factor.o: ELF 32-bit MSB relocatable SPARC Version 1
```

```
me@linux$ file factor.exe
factor.exe: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV)
dynamically linked (uses shared libs), for GNU/Linux 2.6.15, BuildID
[sha1]=0xabca77787115d8fb7fc615cc7f245660248f2625, stripped
```

```
me@linux$ file s_factor.c
s_factor.c:     English text
```

Simple forensic file tools: strings

- Finds printable strings in object or binary files

```
me@cygwin$ strings s_factor.o
s_fact_allocate_vecs
s_fact_allocate_vecs
```

  - These correspond to the program lines

```
exit_err("s_fact_allocate_vecs", malloc_err)
```

- Can be used to embed CVS/SVN ids in objects
  - eg. in C: `static char *cvsid = "$Id$";`
  - CVS/SVN expands `$Id$` to current revision info
  - This revision info is embedded in compiled code as a string!

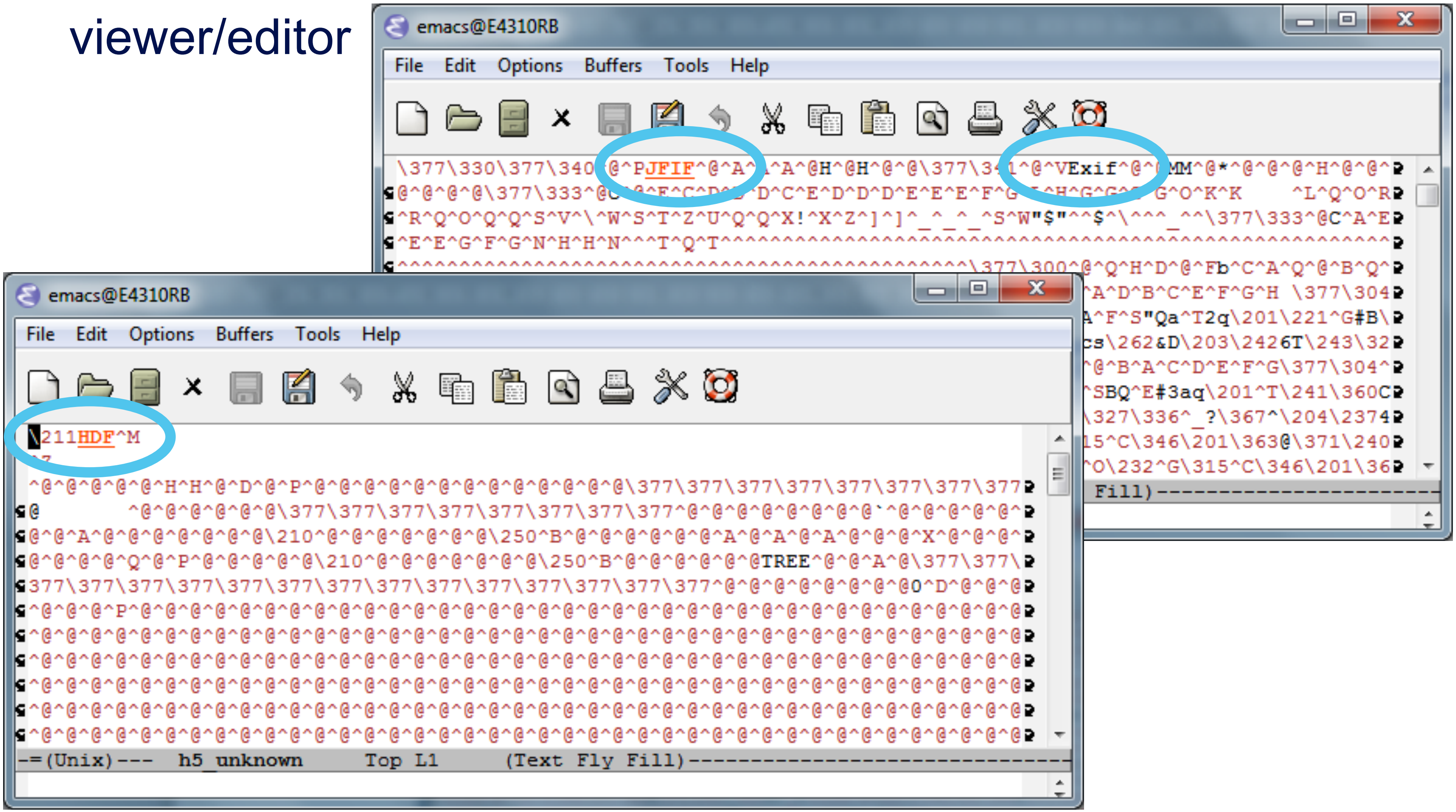
Simple forensic file tools: od

- Stands for ‘octal dump’ (it’s an old function ☺)
- Displays contents of binary files
  - Flags specify display format: chars, ints, floats, doubles,...
  - This command says: interpret each 8-bit octet as an ASCII character:

```
$ od -c unknown-file
0000000 377 330 377 340  \0 020   J   F   I   F  \0 001 001  \0  \0 001
0000020  \0 001  \0  \0 377 333  \0   C  \0  \b 006 006  \a 006 005  \b
0000040  \a  \a  \a  \t  \t  \b  \n  \f 024  \r  \f  \v  \v  \f 031 022
0000060 023 017 024 035 032 037 036 035 032 034 034       $   .   '    
0000100   "   ,   # 034 034   (   7   )   ,   0   1   4   4   4 037   '
0000120   9   =   8   2   <   .   3   4   2 377 333  \0   C 001  \t  \t
0000140  \t  \f  \v  \f 030  \r  \r 030   2   ! 034   !   2   2   2   2
0000160   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
```

Simple forensic tools: Emacs

- The venerable Emacs editor is a very powerful binary file viewer/editor



More sophisticated forensics

- There are dozens of data recovery / forensics firms
  - Some develop their own tools, most provide services

- The most well-known open-source forensics tools are
  - The Sleuth Kit, http://www.sleuthkit.org/
    - Windows, Linux, MacOS X, Unix
  - Foremost, http://foremost.sourceforge.net/
    - Linux, Unix

Summary

- Digital data are stored – in memory, on disk, on tape – as simple bits, 1s and 0s
- The descriptive overlays – file system, byte encoding, file format – are what give those bits meaning
  - And losing these descriptive metadata are a real challenge for long-term preservation!
- Storage of data from and for HPC systems is increasingly standardised
  - Standard formats offer greater portability and longevity
  - And make long-term data management planning that bit easier

Acknowledgements