Improving file formats

From 

to 

?

Ange Albertini

In 1989...

our computer

(10 MHz CPU, 20 Mb HDD)

was infected by a virus...

Thankfully,

a french magazine explained

how to remove it...

...BY YOURSELF, WITH A HEX EDITOR!

"...At the end of the first file allocation table of the hard disk, replace the last 3 bytes FF 7F FF by FF 0F 00. Then find the code of the virus itself which starts with FF 06 F3 7D 8B 1E and overwrite it (including all following bytes, until 55 AA) by F6..."

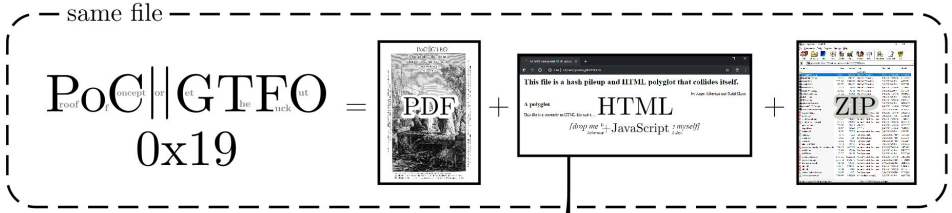
THIS WAS MY INTRODUCTION TO HEX EDITORS AND MALWARE.!

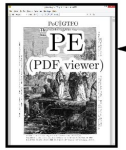
comme endommagés dans la table d'alloca-
tion des fichiers. Par chance, il n'attaque que les IBM PC-XT. Pour s'en débarrasser, il faut rétablir les pistes de démarrage dans leur état d'origine. Avec un éditeur d'octets du type PC-Tools, vérifiez la présence des octets 33 C0 dans les zones 30 et 31 du secteur d'amorçage du disque dur ; s'ils sont bien présents, mieux vaut exécuter la commande SYS depuis une disquette Système saine ; à la fin de la première table d'allocation des fichiers du disque dur, remplacez les trois derniers octets (FF 7F FF) par FF 0F 00. Puis localisez le code du virus lui-même, qui commence par FF 06 F3 7D 8B 1E, et remplacez-le (ainsi que tous les octets qui suivent, jusqu'à 55 AA) par F6 si le formatage est dû à la commande FORMAT du système, ou par 00 s'il provient de PC-Tools. Si l'opération vous semble trop com-

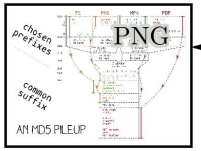
About the author

- 13 years of malware analysis
- now Information Security Engineer

Note: this talk reflects my **own** opinion, not my employer.





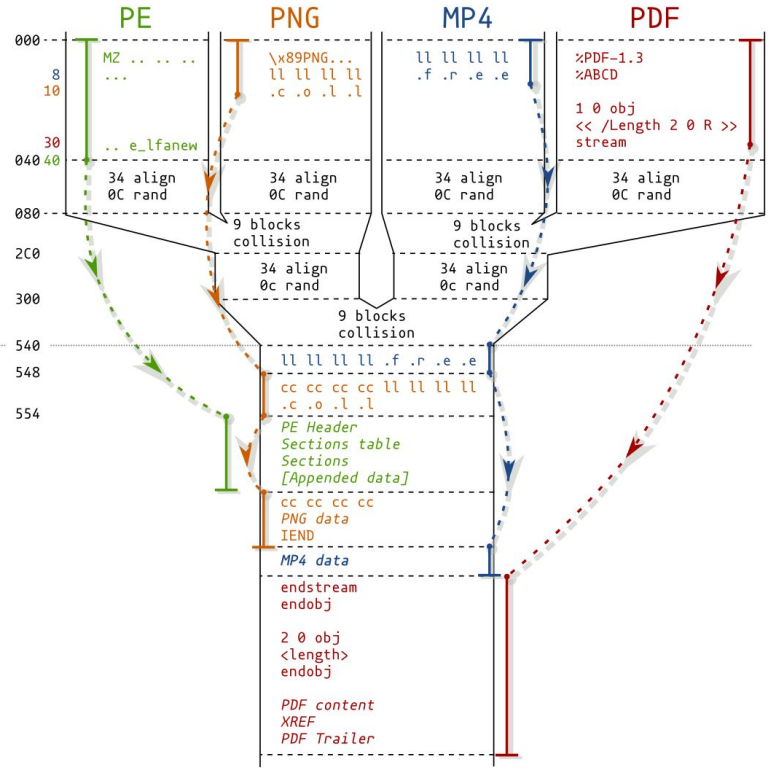


All these files
have the **same** MD5

generates

chosen prefixes

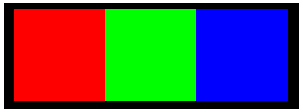
common suffix

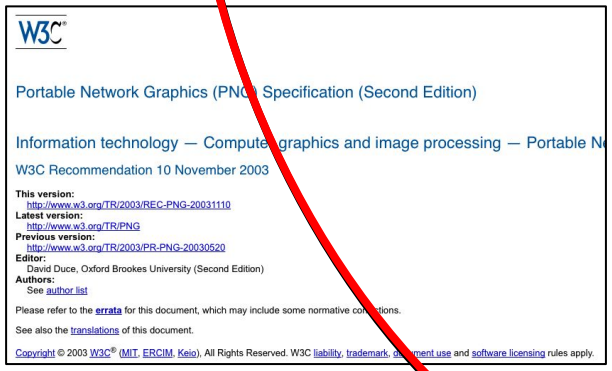


My latest creation:

6 file types, 4 prefixes, 3 hashes collisions.

https://github.com/angea/pocorgtfo#0x19





W3C

Portable Network Graphics (PNG) Specification (Second Edition)

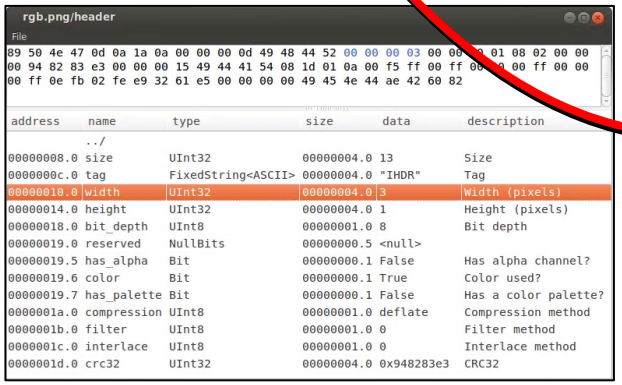
Information technology — Computer graphics and image processing — Portable N

W3C Recommendation 10 November 2003

This version:
http://www.w3.org/TR/2003/REC-PNG-20031110
Latest version:
http://www.w3.org/TR/PNG
Previous version:
http://www.w3.org/TR/2003/PR-PNG-20030520
Editor:
David Duce, Oxford Brookes University (Second Edition)
Authors:
See author list

Please refer to the **errata** for this document, which may include some normative corrections.

See also the translations of this document.



rgb.png/header

File

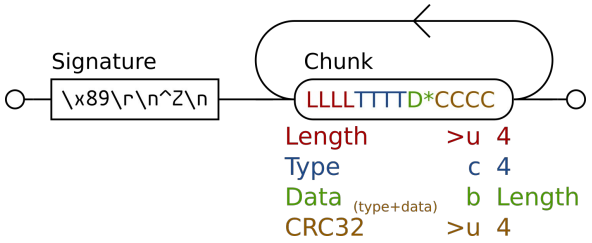
```
89 50 4e 47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52 00 00 00 03 00 00 00 01 08 02 00 00
00 94 82 83 e3 00 00 00 15 49 44 41 54 08 1d 01 0a 00 f5 ff 00 ff 00 0a 00 ff 00 00
00 ff 0e fb 02 fe e9 32 61 e5 00 00 00 00 49 45 4e 44 ae 42 60 82
```

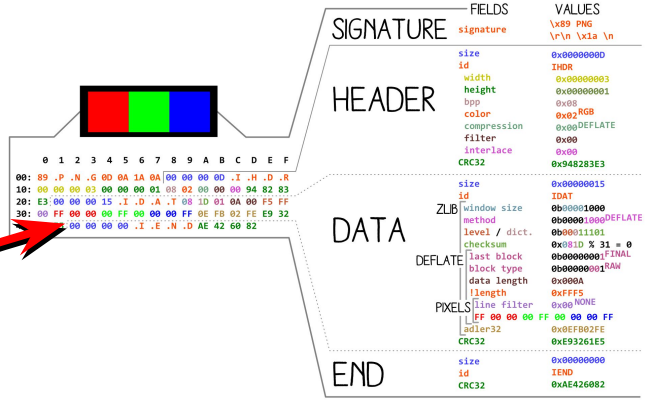
address	name	type	size	data	description
	../				
00000008.0	size	UInt32	00000004.0	13	Size
0000000c.0	tag	FixedString<ASCII>	00000004.0	"IHDR"	Tag
00000010.0	width	UInt32	00000004.0	3	Width (pixels)
00000014.0	height	UInt32	00000004.0	1	Height (pixels)
00000018.0	bit_depth	UInt8	00000001.0	8	Bit depth
00000019.0	reserved	NullBits	00000000.5	<null>	
00000019.5	has_alpha	Bit	00000000.1	False	Has alpha channel?
00000019.6	color	Bit	00000000.1	True	Color used?
00000019.7	has_palette	Bit	00000000.1	False	Has a color palette?
0000001a.0	compression	UInt8	00000001.0	deflate	Compression method
0000001b.0	filter	UInt8	00000001.0	0	Filter method
0000001c.0	interlace	UInt8	00000001.0	0	Interlace method
0000001d.0	crc32	UInt32	00000004.0	0x948283e3	CRC32

```python
import struct
import binascii
import zlib

def make(chunks):
  s = ["\x89PNG\x0d\x0a\x1a\x0a"]
  for type_, data in chunks:
    s += [
      struct.pack(">I", len(data)), # length
      type_,                        # type
      data,                         # data
      struct.pack(">I",             # CRC
          binascii.crc32(type_ + data)
            % 0x100000000)
      ]
  return "".join(s)

with open("rgb.png", "wb") as f:
  f.write(make([
    ["IHDR", # Image Header
      struct.pack(">IIBBBBB",
        3, # Width
        1, # Height
        8, # Depth: 8
        2, # Colour type: RGB
        0, # Compression: Deflate
        0, # Filter: adaptive filtering
        0  # Interlace: no
        )],
    ["IDAT", # Image Data
      zlib.compress(
      "\0" +           # no interlacing
      "\xFF\x00\x00" +  # Red pixel
      "\x00\xFF\x00" +  # Green pixel
      "\x00\x00\xFF"    # Blue pixel
      ,0)], # no compression
    ["IEND", # Image End
      ""]    # Empty chunk
    ]))
```





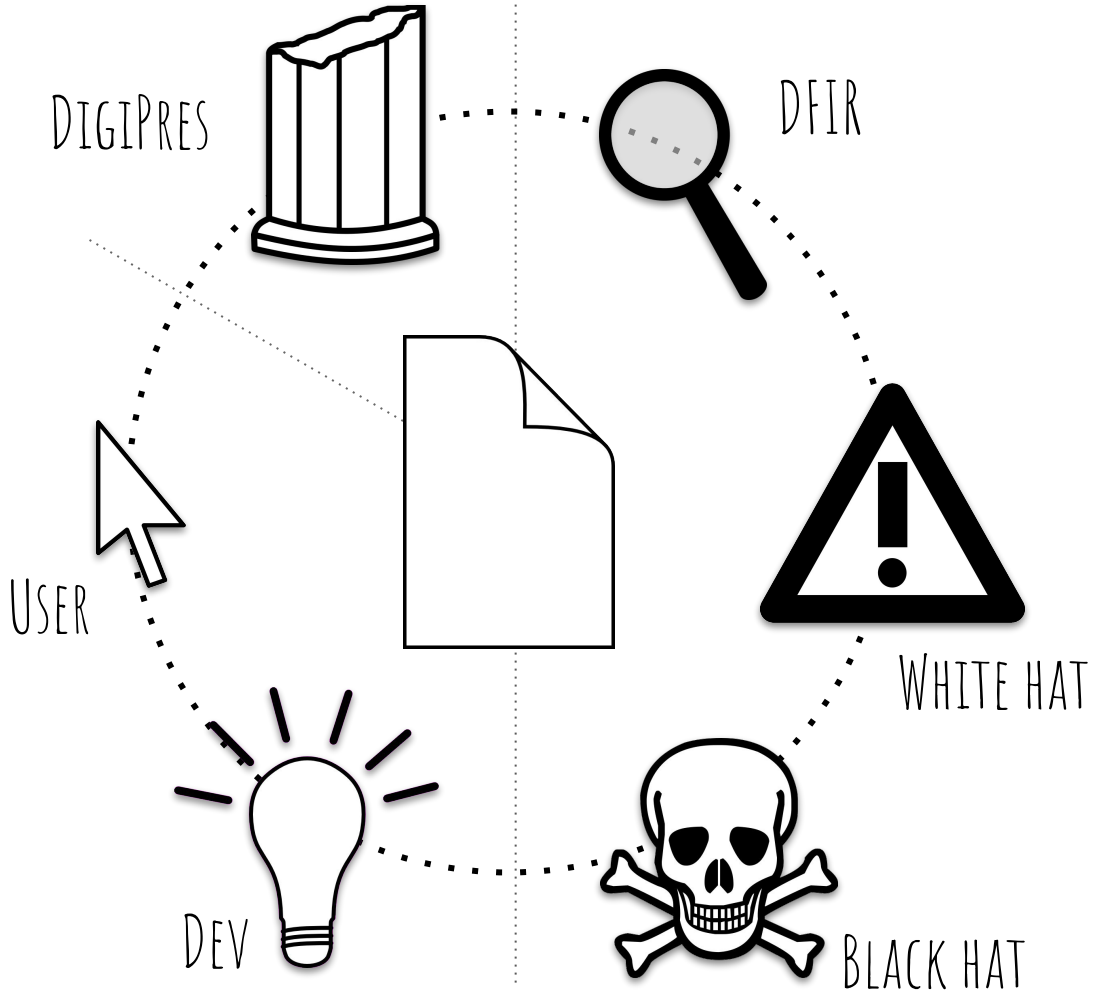
Document, visualize draw, teach.

There are various

(with a few things in common)

communities around file formats

...and I'm interested in all of them



As a starter...

Let's craft a (commercial & successful) software from scratch...

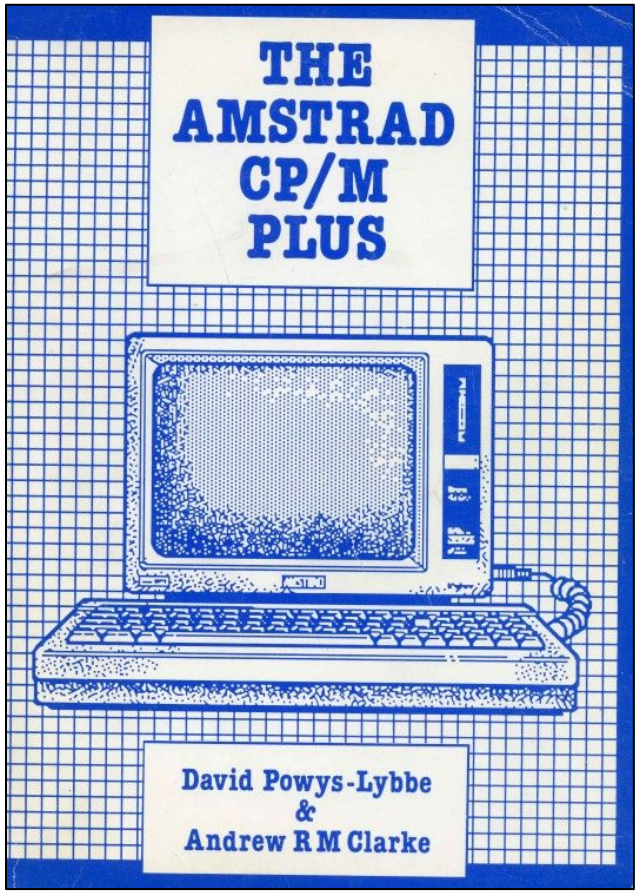
(Yes, really)

On this computer...


Amstrad 128K Microcomputer (v3)
©1985 Amstrad Consumer Electronics plc
and Locomotive Software Ltd.
BASIC 1.1
Ready
|CPM
LET'S LAUNCH...

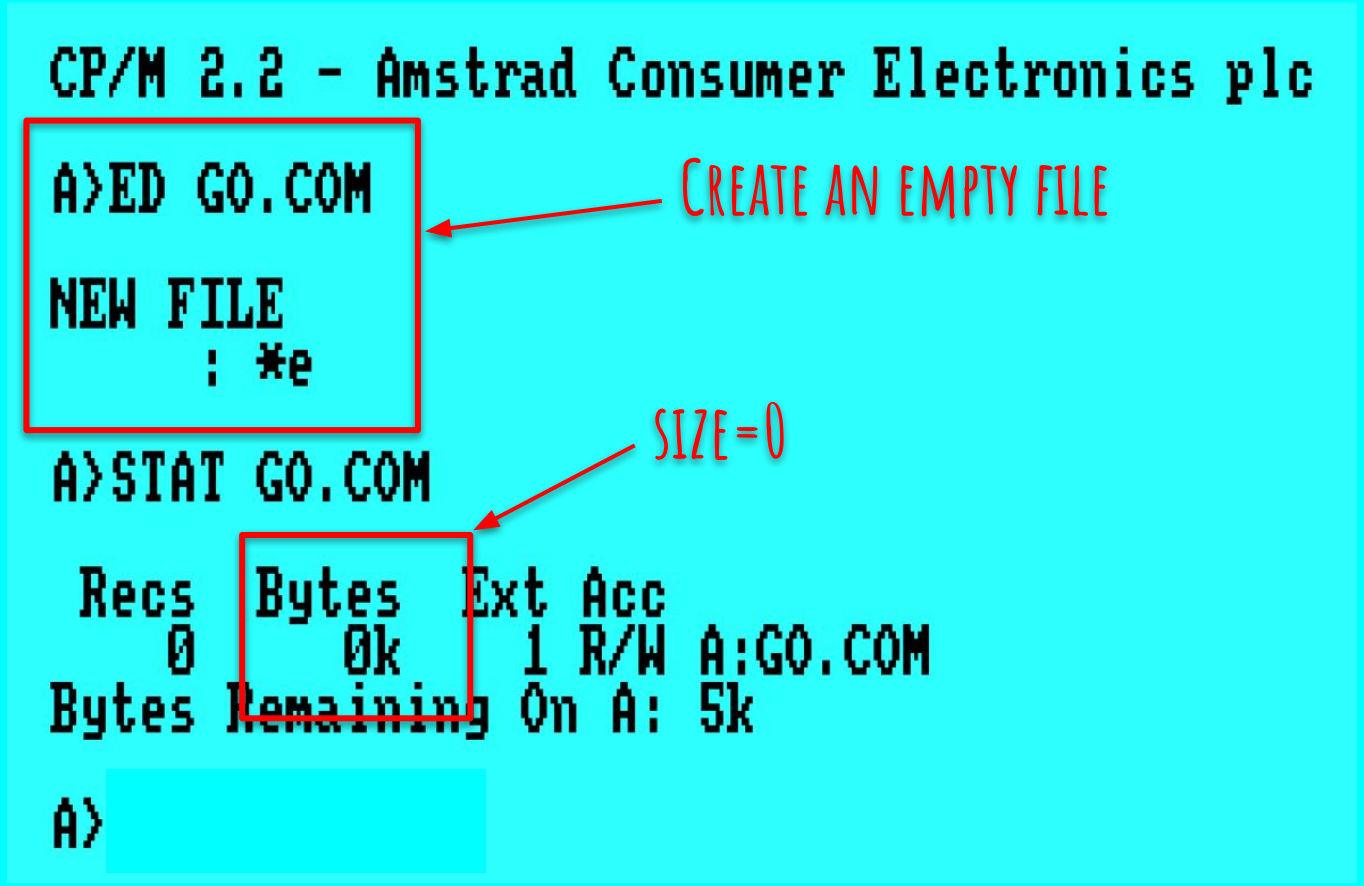
...this OS.

CP/M 1974 -> DOS 1981 -> Windows 1985



3" Compact Floppy 2
180 Kb / side

Let's create... an EMPTY executable!



Is it valid?

Yes: **Transient Commands** are blindly loaded

(that's how executables
were called on CP/M)

and execution is started at offset zero.

Does it do anything?

The Transient Memory Area is **not**
Cleared between executions,
so the previous command is re-executed.

WORKING AS INTENDED
(REPEATS PREVIOUS COMMAND)

Consistent & reliable.

```
      R 03 02
     COMMODORE 64  44k CP/M vers 2.2
     ==============================
     Copyright (C) 1988, ROSSMOELLER
A>ED GO.COM
NEW FILE
     : *e

A>DUMP GO.COM

A>DUMP
NO INPUT FILE PRESENT ON DISK
A>GO DUMP.TXT
0000 0D 0A 4D 49 54 20 44 45
0008 4D 20 42 45 46 45 48 4C

A>
```

Under a commercial OS from 1985,
the **empty** file is valid, useful and reliable.

It was even **sold** as a commercial program for £5.

Lessons learned

- Many things have changed since the 80s :)

But....

- weird files are nothing new.
- Software always defined the rules.
  - Specifications are entirely optional.
  - There's no "that's not how it works".

The file format problem

A misunderstood field -"specs are enough"

-> received less attention

-> least rigourous field of computing.

Not enough pre-natal checks.

Lacking growth control.

Crypto

File formats

We need...

Better controls when designing a format.

Better checks to follow its evolution.

And we need to educate the different communities.

There is hope:

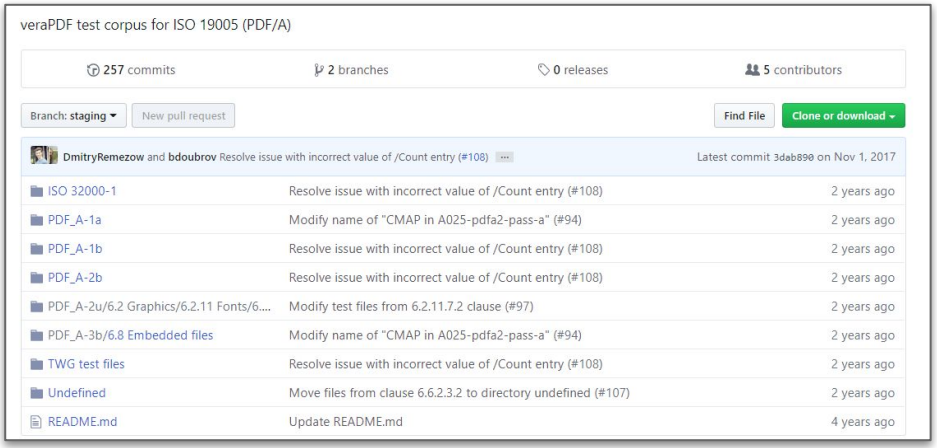
some great formats-focused projects...

Note that none of these projects is from the original developer
and was started long after the format became mainstream.

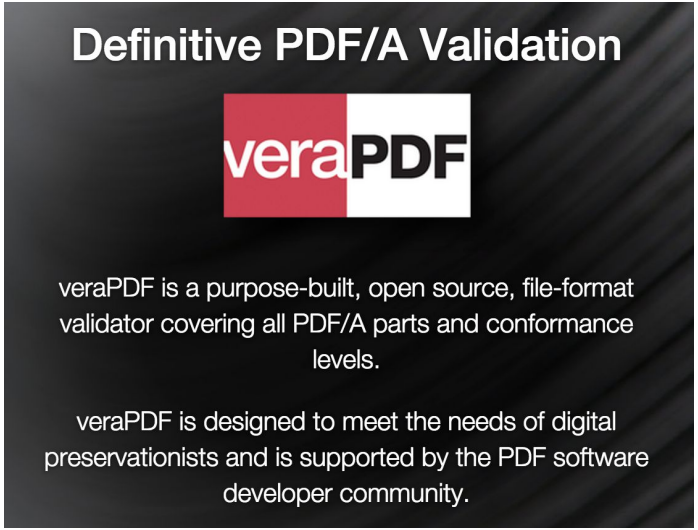
I.E. a format must be mainstream for a very long time
until someone started something similar, much later.

VeraPDF

open source PDF/A validator and its corpus, and more...



PDF: Adobe 1993 VeraPDF: ISO 2014



CaraDoc

https://github.com/caradoc-org/caradoc

Caradoc - a PDF parser and validator

Caradoc is a parser and validator of PDF files written in OCaml. This is version 0.3 (beta).

Caradoc provides many commands to analyze PDFs, as well as an interactive user interface in console.

Caradoc was presented at the the third Workshop on Language-Theoretic Security (LangSec) in May 2016.

Nicolas Seriot's JSON parsers analysis

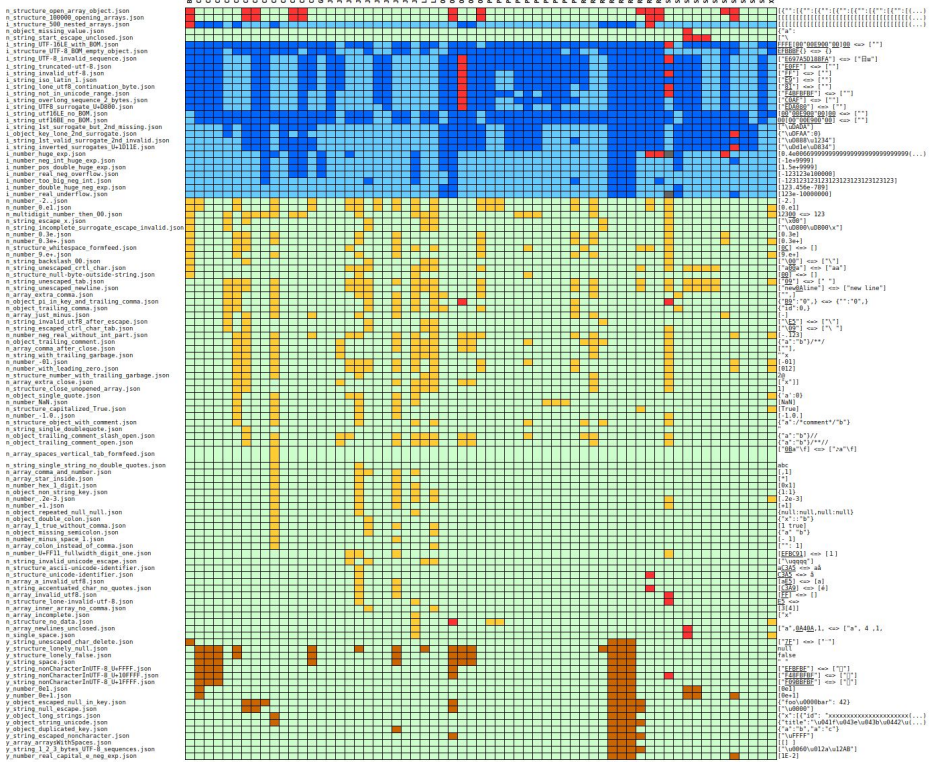
Cornercases. PoCs. Test suite.

Comparative charts...

http://seriot.ch/parsing_json.php

While JSON is fairly simple,

it's still a huge effort for a single person.



Michał Górny's TAR analysis

Format	**GNU + base-256**									**pax**													
Implementation	**p**	**s**	**u**	**m**	**n**	**r**	**l**	**y**	**v**	**p**	**s**	**u**	**g**	**m**	**n**	**t**	**a**	**f**	**x**	**r**	**l**	**v**	**y**
GNU tar	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
libarchive	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
star	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
NetBSD pax	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
busybox	✓	✓	✓	✓	✓	✗	✗		✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓
Python	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✓
p7zip	✓	✓	✗	✓	✗	✓	✗		✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
7-Zip	✓	✓		✓	✓	✓	✗		✗	✗	✗			✗	✗	✗				✗	✓	✗	✗
WinRAR	✓	✓		✗	✗	✗	✓	✗	✗	✓	✓			✗	✗	✗				✗	✓	✗	✓

https://dev.gentoo.org/~mgorny/articles/portability-of-tar-features.html

BMP Suite

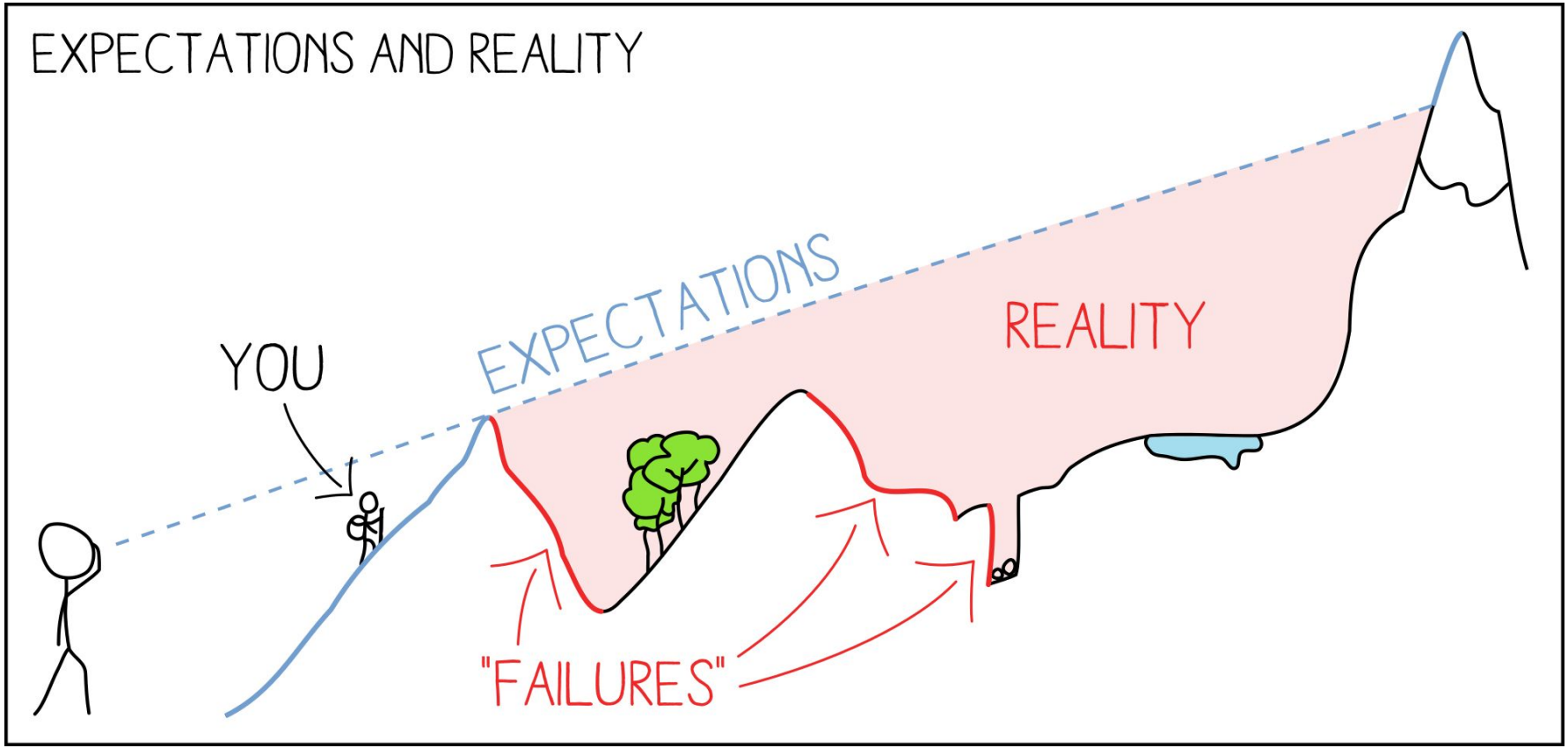
https://github.com/jsummers/bmpsuite

q/pal2.bmp	3			A paletted image with 2 bits/pixel. Usually only 1, 4, and 8 are allowed, but 2 is legal on Windows CE.
q/pal2color.bmp	3			Same as pal2.bmp, but with a color palette instead of grayscale palette.

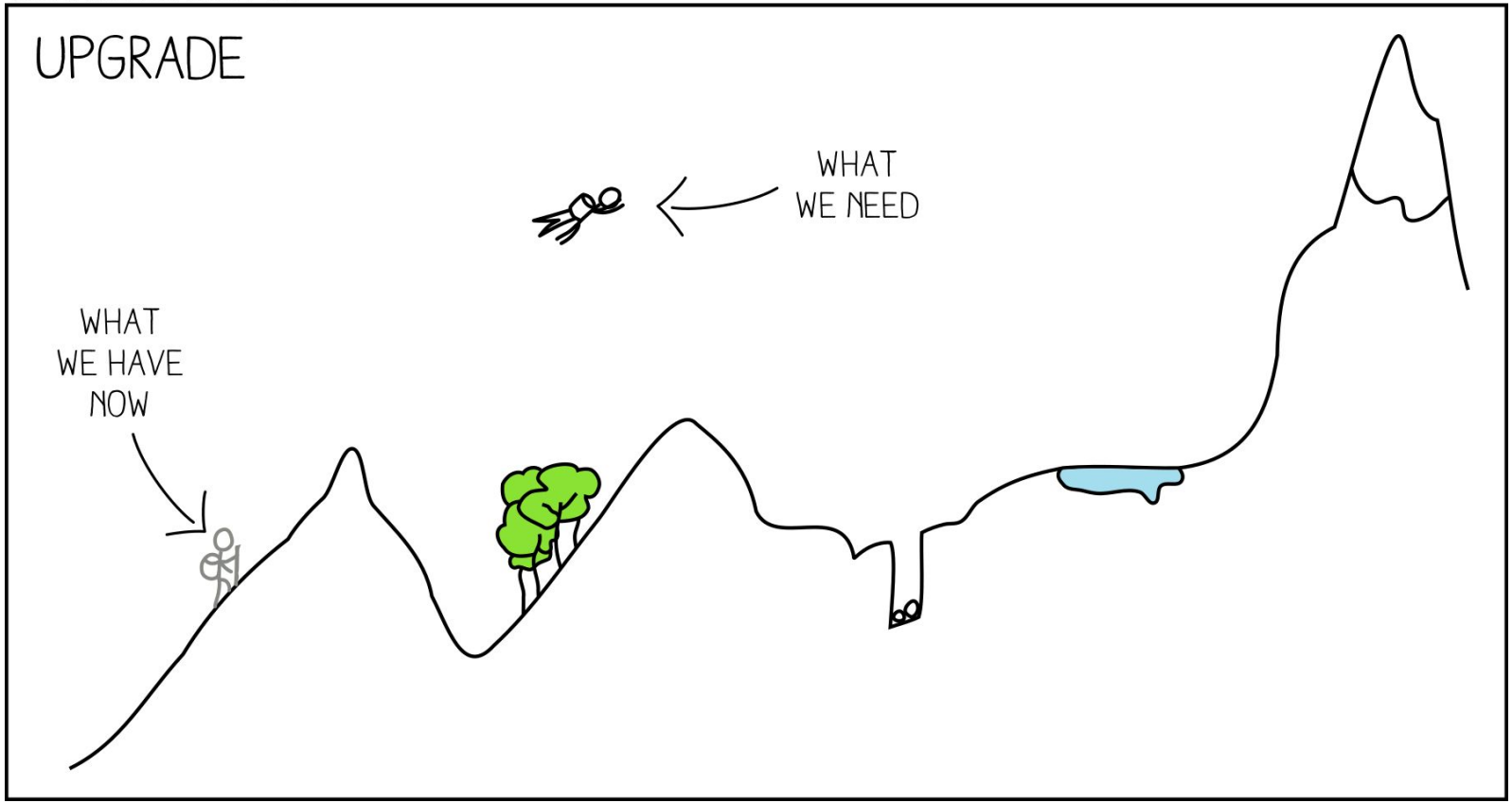
q/pal4rletrns.bmp	3	 or  or 		An RLE-compressed image that uses “delta” codes to skip over some pixels, leaving them undefined. Some viewers make undefined pixels transparent, others make them black, and others assign them palette color 0 (purple, in this case).

We need new tools to define the (current) ground truth.

New (automated, scalable) tools -> visibility of the landscape
-> understanding (documentations and metrics)
-> update of the state of the art
-> educating communities -> change the landscape



There are always unknown unknowns.



We need to explore at scale.

From GIF to JIF

GIF (1987) used LZW - patented, and enforced in 1994

JIF was created: GIF (LZW 1984) -> JIF (zlib 1990)

Technically, JIFs had all reasons to replace GIFs.

.jif: jeff's image format

Jeff's Image Format (.jif) is an easy replacement for CompuServe's *Graphic Interchange Format* (.gif). An easy replacement for the GIF standard is warranted since the GIF standard uses LZW which is patented by Unisys. Unisys requires licensing for use of the technology covered by their patent. The *Portable Network Graphics* (.png) format is a well-designed standard that can be used to replace static (non-animated) GIF files.

There are many reasons why the GIF standard has remained the standard format for graphics on the web. The PNG standard weakens many of them, but several problems remain:

- animated GIFs cannot be encoded as PNG files, although there is another standard, Multiple-image Network Graphics (.mng), that addresses this;
- properly processing PNG images requires more sophisticated image algorithms - this is probably one of the main reasons why some key software still does not fully implement PNG support (for example, the introduction of an alpha channel requires more sophisticated image layering algorithms) - the required infrastructure is not yet present; and
- automatic conversion of PNG files into GIF files is problematic (this is desirable when a GIF client is requesting an image stored in the PNG format).

When PNG was constructed, many thought it an opportune time to introduce new features into a baseline image format and clearly separate static and animated image formats. Unfortunately, the mutual incompatibility of the PNG and GIF standards has lengthened the amount of time the computing public has had to endure using the patent-encumbered GIF standard. I am proposing that, as a partial step, a less aggressive change be pursued: simply replace the LZW compression used in the GIF standard with the LZ77-derived compression used in the PNG standard. This realizes several advantages:

- any GIF file may be automatically converted into a corresponding JIF file,
- any JIF file may be automatically converted into a corresponding GIF file,
- the changes required to add JIF compatibility to a GIF-compatible program are as trivial as possible - file parsing code may be kept and all valid assumptions regarding image processing are still valid (no introduction of new features).

The differences between GIF and JIF are summarized in the following table.

	GIF	JIF
extension	gif	jif
type (for Macintosh files)	GIFf	JIFf
signature (first six bytes of file)	GIF87a or GIF89a	JIF99a
compression	LZW	LZ77 derivative

JIF: AN OBVIOUS IDEA, LOST IN TIME.

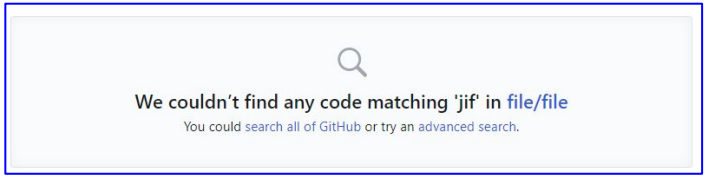
IN PRACTICE, JIF DOESN'T EXIST:

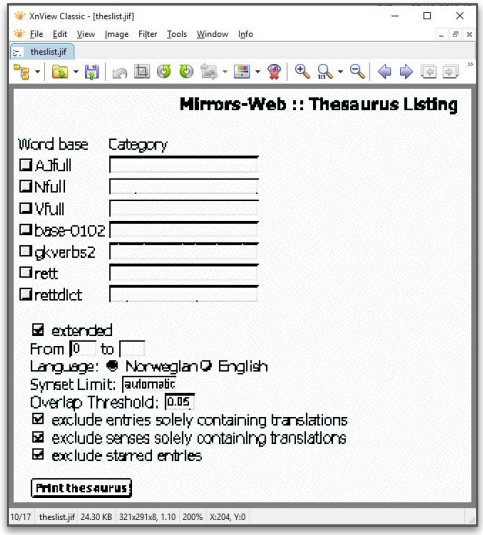
- UNKNOWN TO `file`
- UNKNOWN TO VIRUSTOTAL
  - A SINGLE FILE, THAT I UPLOADED RECENTLY.

BUT IT'S SUPPORTED BY XNVIEW

-> DEPRECATION IS VERY HARD.

-> INFOSEC DOESN'T OVERLAP WITH DIGIPRES.





https://folk.uib.no/hfohd/SLF/Dyvik/theslist.jif
0fb6018a224cfd9926968c80621f20660b825ec17ef4707b64a0a1d77abf9281

Deprecation? fear, uncertainty, doubt.

GIF deprecation == "no more memes/cat pics"? -> irrationality

Fight irrationality with 'data-driven explanations'.

-> documentations and metrics.

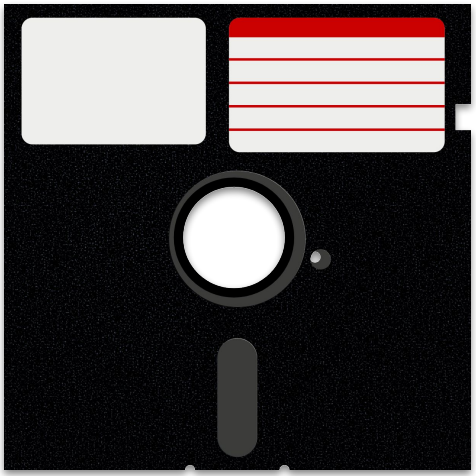
Which, for now, means just "original specs".

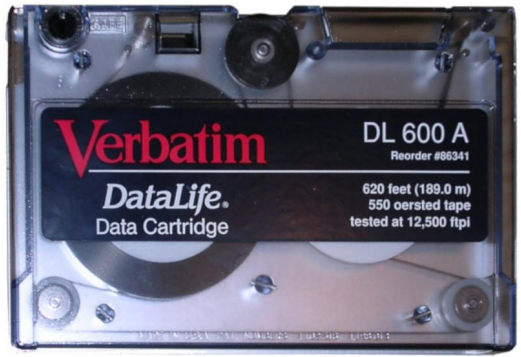
(that are 30+ year old)

Yet we still use Tape/floppies oriented feature!

We can't kill ZIP/Tar.

Because of no visibility or way to enforce a successor.

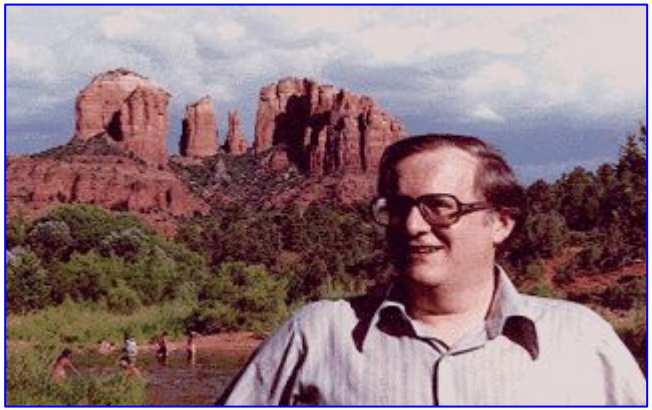




GIF Plain Text Extension

https://github.com/corkami/formats/blob/WIP/image/gif89a.md#plain-text-extension

A long forgotten (yet official) way for GIF to display text (they're not comments)



BOB_89A.GIF

This image contains these text frames

```
--------: Introducing GIF89a :--------

When you finish reading this, press
any key to continue. If you just sit
back and watch, we'll continue when
the built-in delay runs out.

GIF89a provides for "disposing of"
an image or text. All the text in
this GIF is "restore to previous",
so that the underlying image is
restored when you press a key or
the delay runs out.

"Transparent" images
or text can be written
over an underlying
image so that parts of
the old image "show
through" the new one.

Oh, incidentally, it's
pronounced "JIF"
```

Specifications

Written years/decades ago.

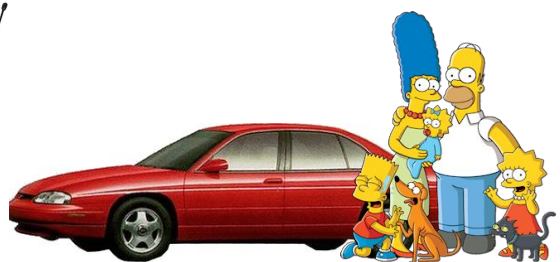
Originally made for 80x25 screens :)

Never updated.

Some features are lost

or never implemented.

Novelties from 1989



Graphics Interchange Format Data Definition

GENERAL FILE FORMAT

It's not just GIF! Another obvious absence in 2019...

No standard way to make transparent JPGs(1992)

There are many possible ways (PDF, SVG, TIF, PSD)
but no generalized one.

A typical file format timeline

Good intentions:

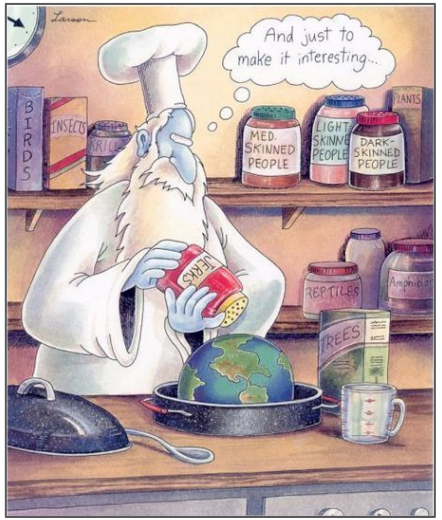
proper planning. Official specs. Set in stone.

Bad things happen:

Interpretation blur, unofficial extensions.

Format is now used everywhere:

Misunderstood. Unmovable.

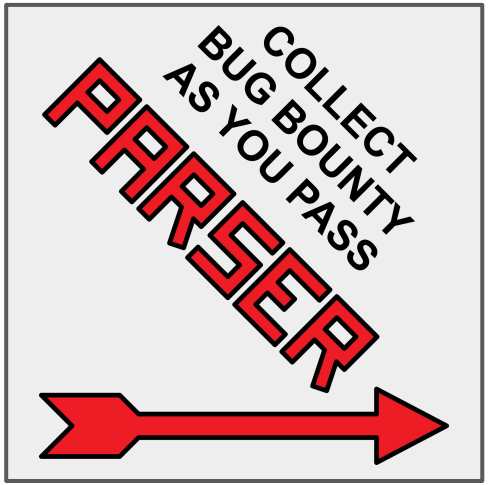


A new (version of a) parser is out?

Fuzz. Get bug fixed. Collect pride & glory.

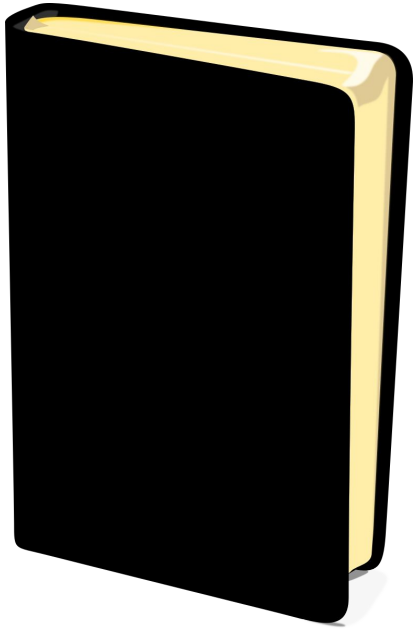
Rinse. Repeat.

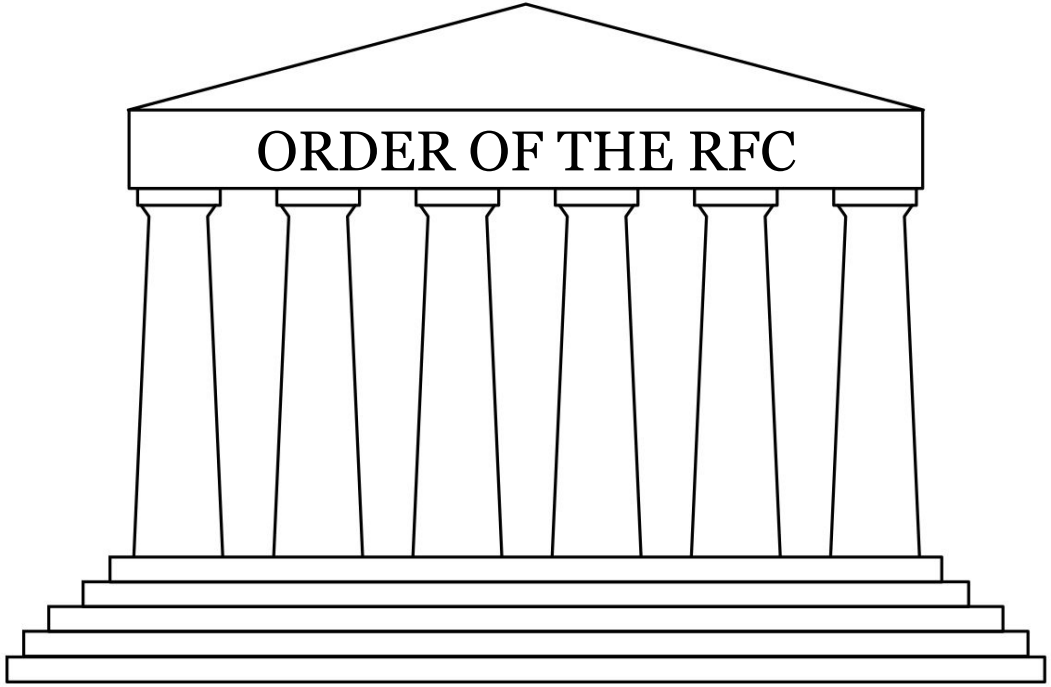
```
10 ParserUpdate
20 Fuzz
30 Fail
40 Collect
50 GOTO 10
```



How we perceive file formats:

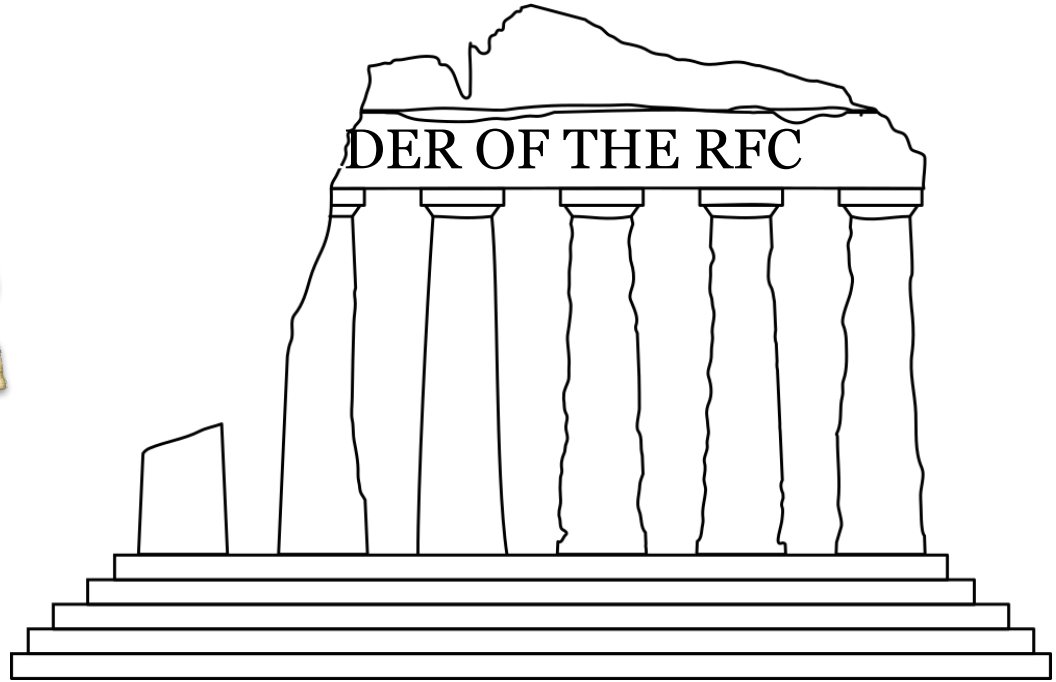
A holy text and its cult.





More like...

outdated and irrelevant practices.



WHAT WE HAVE

(WHAT WE'RE LEFT WITH)

SH*TMYSPECSSAYS

(OUTDATED/IRRELEVANT)

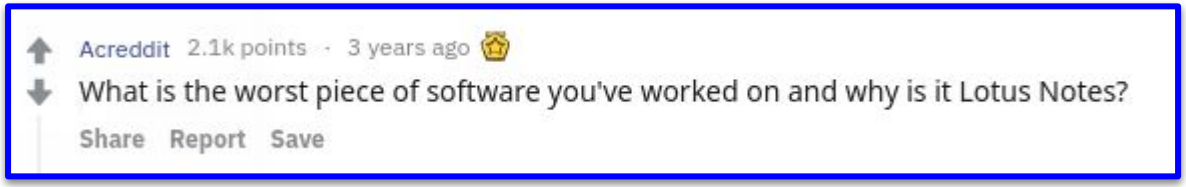
[GIF]
The Plain Text Extension contains textual data and the parameters necessary to render that data as a graphic, in a simple form.

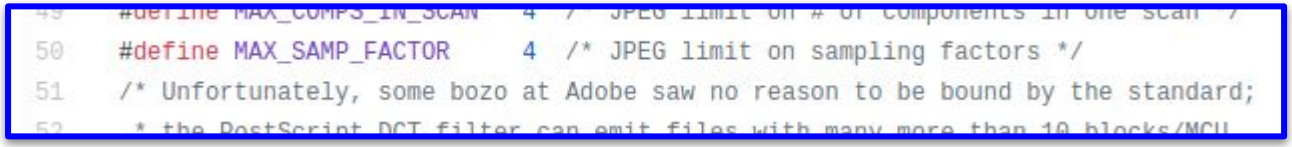
Spanning is the process of segmenting a ZIP file across multiple removable media. This support has typically only been provided for DOS formatted floppy diskettes.

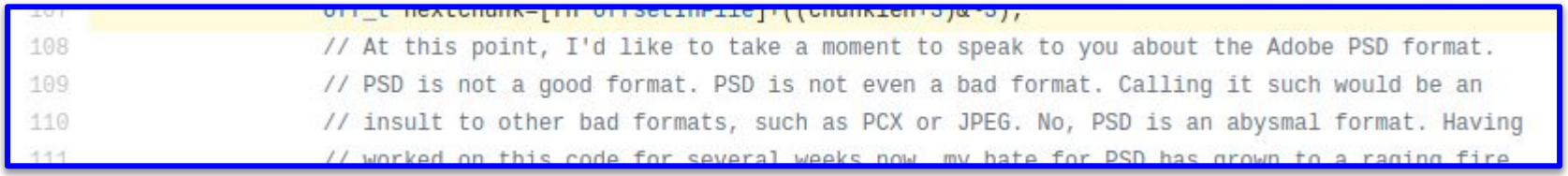
The following GIF Capabilities Response message describes three standard IBM PC Enhanced Graphics Adapter configurations with no printer; the GIF data stream can be processed within an error correcting protocol:

[PNG]
For colour types 2 and 6 (truecolour and truecolour with alpha), the PLTE chunk is optional. If present, it provides a suggested set of from 1 to 256 colors to which the truecolor image can be quantized if the viewer cannot display truecolor directly.
...
A CRC should be checked before processing the chunk data.

[JPEG]
The APP0 marker is used to identify a JPEG FIF file.
The JPEG FIF APP0 marker is mandatory right after the SOI marker.







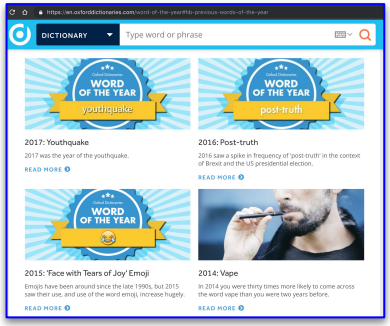
What we see...

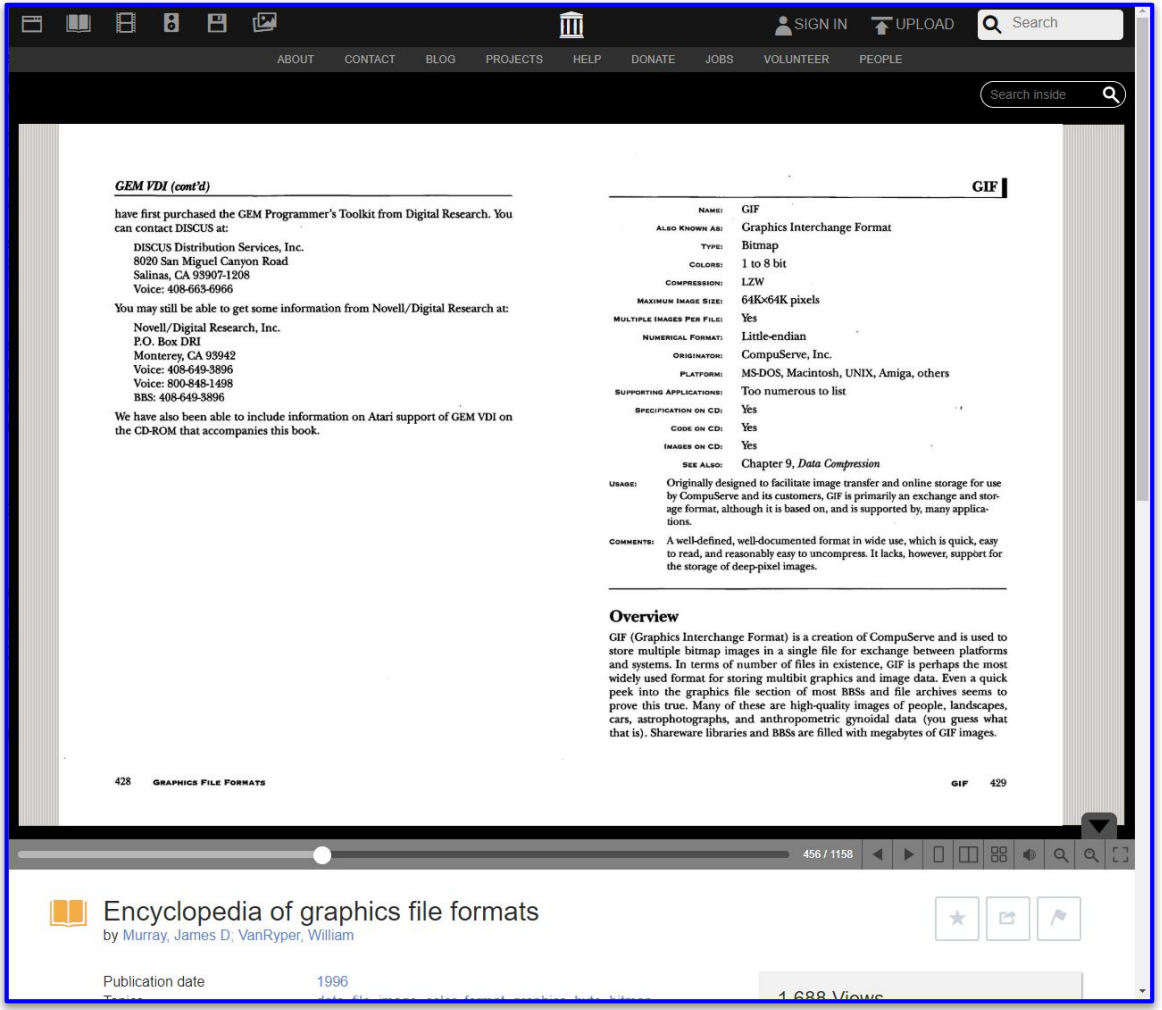
Sh*tMyParserSays

Encyclopedia of graphics file formats

A 'good' reference but:
- outdated (1996).
- doesn't reflect the current landscape.

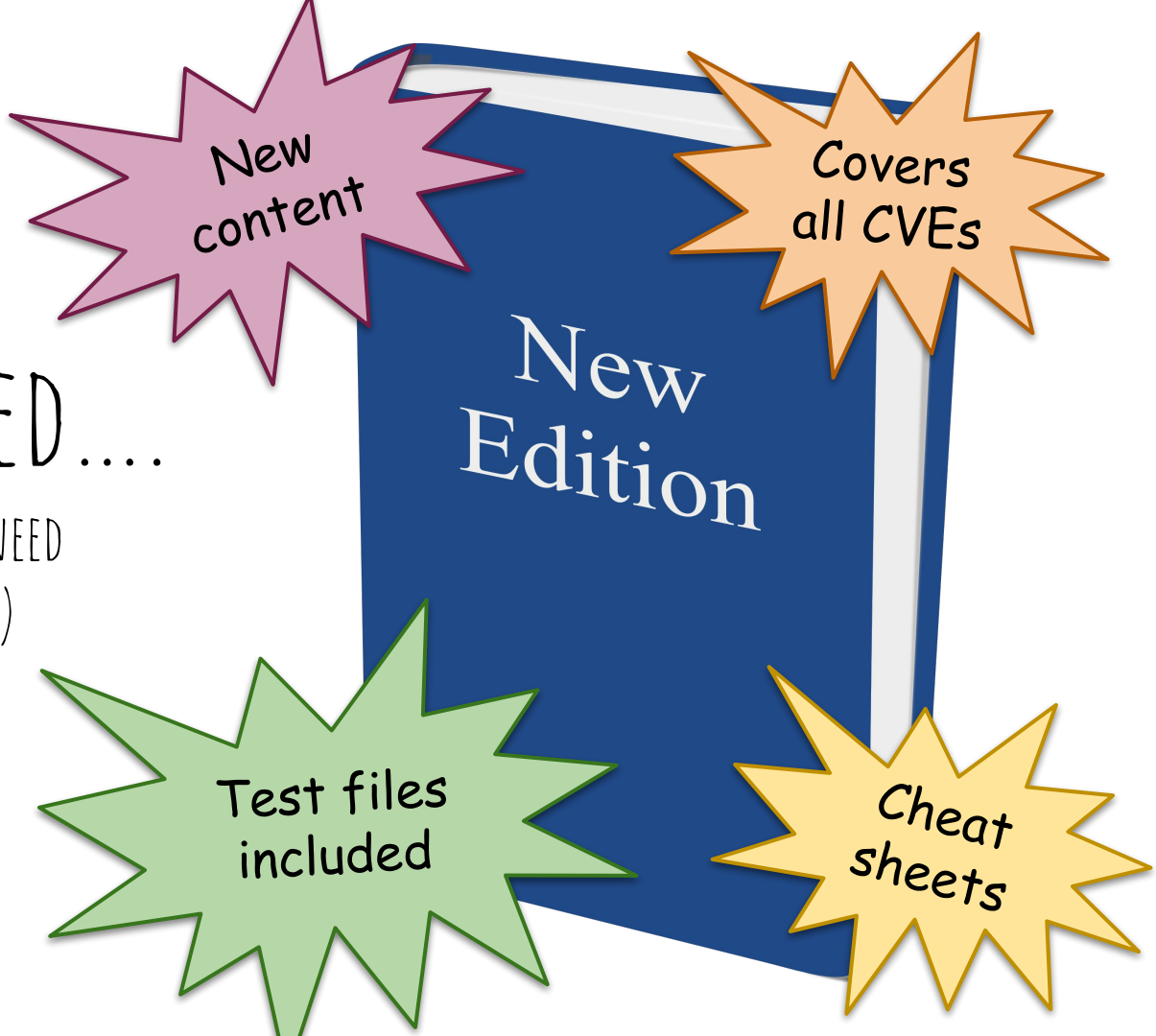
Oxford dictionary: still fresh





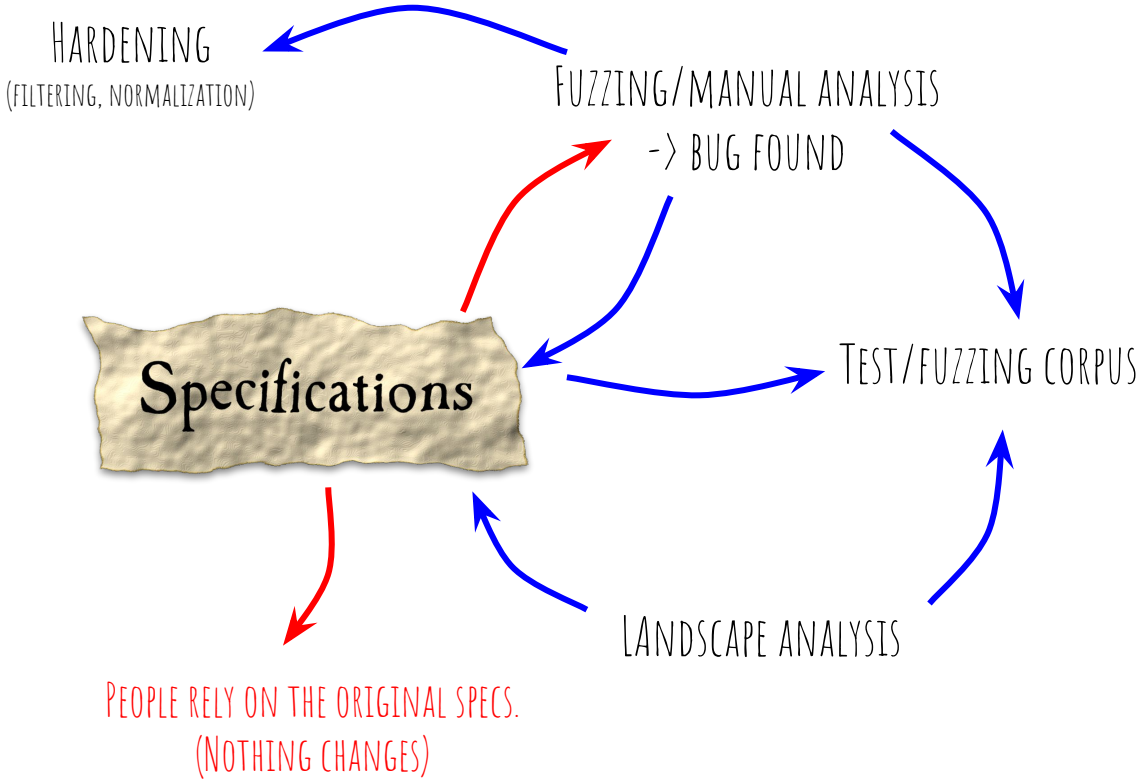
What we'd need....

(more exactly, we first need the tools to get there)



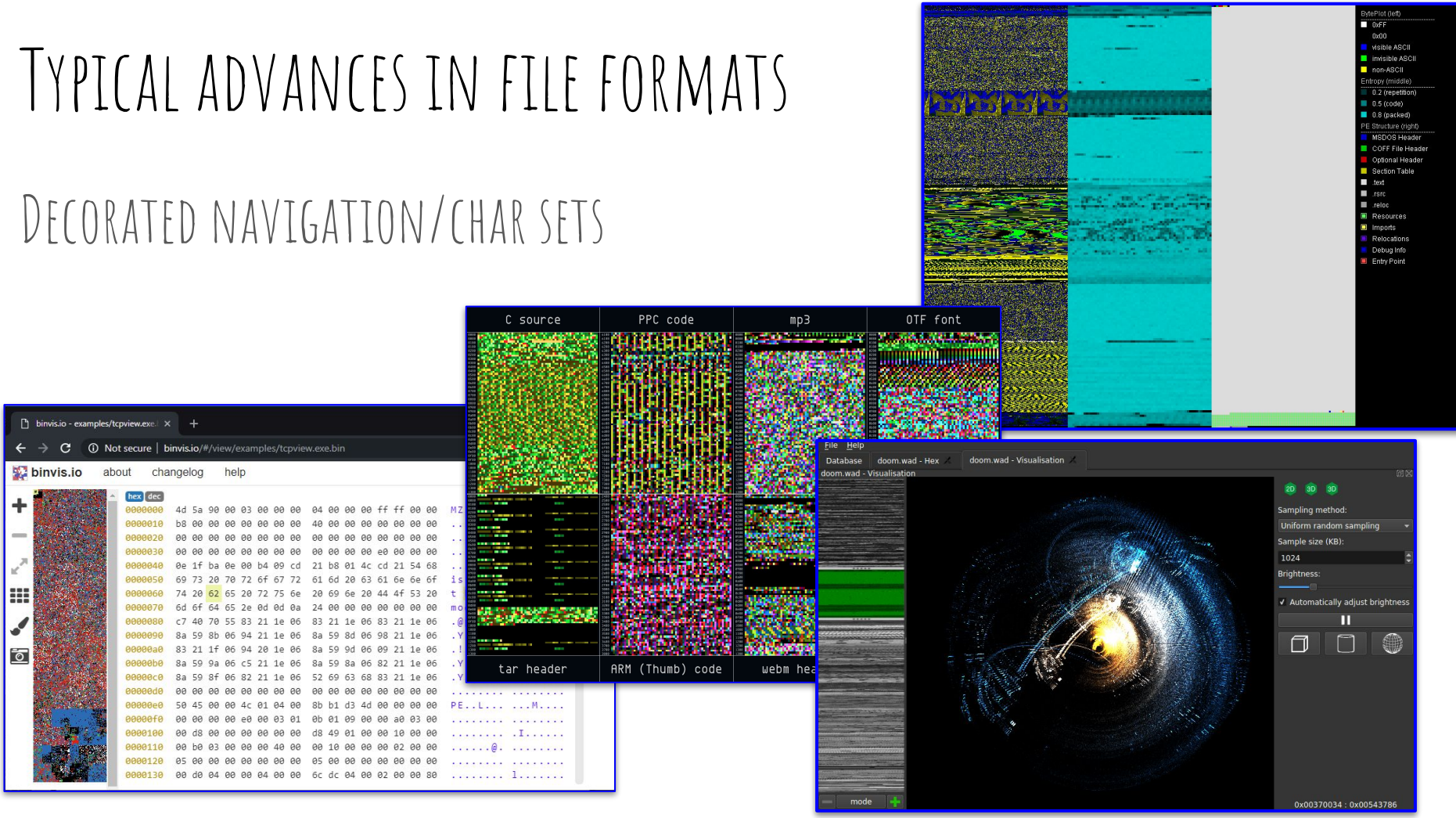
The status quo

How it is (mostly)
How it should be.



Typical advances in file formats

Decorated navigation/char sets



Kaitai

From Yaml grammar to...

```
meta:
  id: bmp
  file-extension: bmp
  endian: le
  license: CC0-1.0
  ks-version: 0.8
seq:
  - id: file_hdr
    type: file_header
  - id: len_dib_header
    type: s4
  - id: dib_header
    size: len_dib_header - 4
    type:
      switch-on: len_dib_header
      cases:
        12: bitmap_core_header
        40: bitmap_info_header
        104: bitmap_core_header
        124: bitmap_core_header
types:
  file_header:
    -orig-id: BITMAPFILEHEADER
    seq:
      - id: magic
        -orig-id: bfType
        contents: "BM"
      - id: len_file
        -orig-id: bfSize
        type: u4
      - id: reserved1
        -orig-id: bfReserved1
        type: u2
```

Archive Files
cpio_old_le , gzip , lzh , rar , zip

Commonly Used Data Types
bcd , vlq_base128_be , vlq_base128_le

Executables and Byte-code
dex , dos_mz , elf , java_class , mach_o , microsoft_pe , python_pyc_27 , swf

Firmware
andes_firmware , ines , uimage

Game Data Files
allegro_dat , doom_wad , dune_2_pak , fallout2_dat , fallout_dat , ftl_dat , gran_turismo_vol , heaps_pak , heroes_of_might_and_magic_agg , heroes_of_might_and_magic_bmp , quake_mdl , quake_pak , renderware_binary_stream , saints_row_2_vpp_pc , warcraft_2_pud

Hardware Protocols
edid , mifare_classic

Logs
aix_utmp , glibc_utmp , hashcat_restore , sudoers_ts , systemd_journal , windows_evt_log

macOS-specific
ds_store

Networking Protocols
bitcoin_transaction , dns_packet , ethernet_frame , hccap , hccapx , icmp_packet , ipv4_packet , ipv6_packet , microsoft_network_monitor_v2 , packet_ppi , pcap , protocol_body , rtcp_payload , rtp_packet , tcp_segment , tls_client_hello , udp_datagram , websocket

Security
efivar_signature_list , openpgp_message , ssh_public_key

Windows-specific
regf , windows_lnk_file , windows_minidump , windows_resource_file , windows_shell_items , windows_systemtime

CAD
monomakh_sapr_chg

Databases
dbf , gettext_mo , sqlite3 , tsm

Filesystems
apm_partition_table , apple_single_double , cramfs , ext2 , gpt_partition_table , iso9660 , luks , lvm2 , mbr_partition_table , tr_dos_image , vdi , vfat , vmware_vmdk , zx_spectrum_tap

Fonts
ttf

Geospatial (Maps)
shapefile_index , shapefile_main

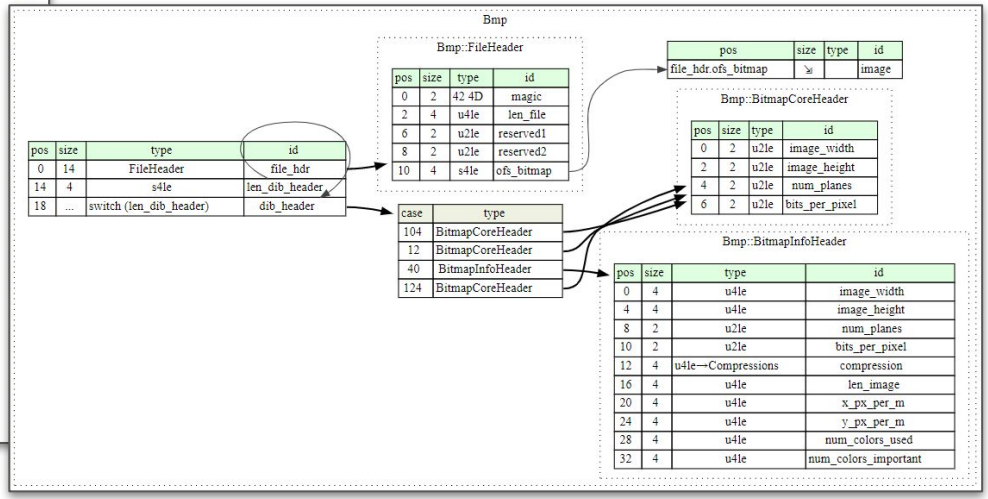
Image Files
bmp , dicom , exif , exif_be , exif_le , gif , icc_4 , ico , jpeg , pcx , pcx_dcx , png , psx_tim , tga , wmf , xwd

CPU / Machine Code Disassembly
code_6502

Multimedia Files
avi , blender_blend , creative_voice_file , fasttracker_xm_module , genmidi_op2 , id3v1_1 , id3v2_3 , id3v2_4 , magicavoxel_vox , ogg , quicktime_mov , s3m , standard_midi_file , stl , vp8_ivf , wav

Scientific Applications
avantes_roh60 , nt_mdt , nt_mdt_pal , specpr

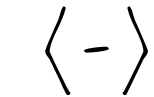
Serialization Protocols
asn1_der , bson , google_protobuf , microsoft_cfb , msgpack , ruby_marshal



Kaitai: Many formats (and grammar visualisation)

https://github.com/kaitai-io/dicom.ksy/blob/master/dicom.ksy

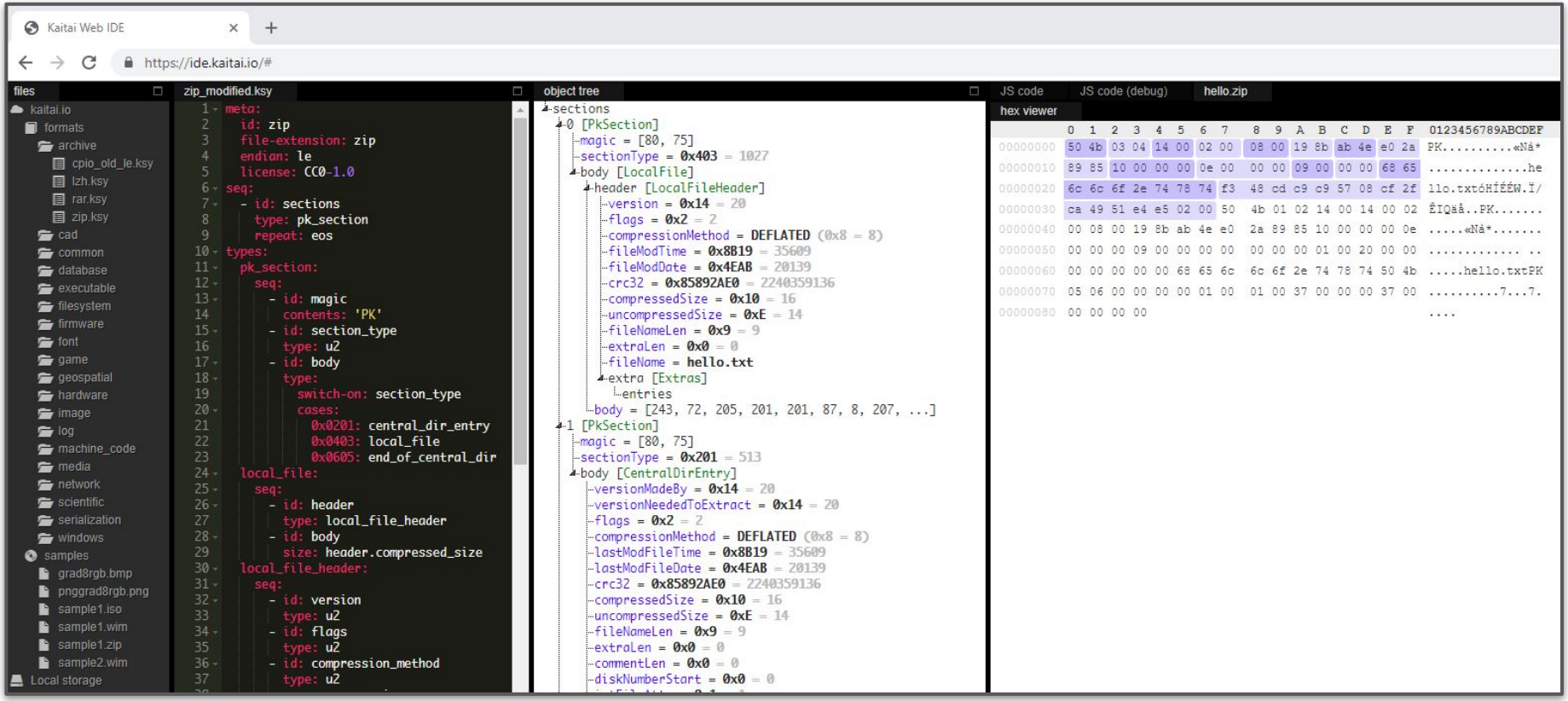
```
meta:
 id: dicom
 file-extension: dcm
 license: MIT
 endian: le
seq:
 - id: file_header
   type: t_file_header
 - id: elements
   type: t_data_element_implicit
   repeat: eos
types:
 t_file_header:
   seq:
     - id: preamble
       size: 128
     - id: magic
       contents: 'DICM'
[...]
```



The DICOM Standard

Title
DICOM Part 1: Introduction and Overview
DICOM Part 2: Conformance
DICOM Part 3: Information Object Definitions
DICOM Part 4: Service Class Specifications
DICOM Part 5: Data Structures and Encoding
DICOM Part 6: Data Dictionary
DICOM Part 7: Message Exchange
DICOM Part 8: Network Communication Support for Message Exchange
DICOM Part 10: Media Storage and File Format for Media Interchange
DICOM Part 11: Media Storage Application Profiles
DICOM Part 12: Media Formats and Physical Media for Media Interchange
DICOM Part 14: Grayscale Standard Display Function
DICOM Part 15: Security and System Management Profiles
DICOM Part 16: Content Mapping Resource
DICOM Part 17: Explanatory Information
DICOM Part 18: Web Services
DICOM Part 19: Application Hosting
DICOM Part 20: Imaging Reports using HL7 Clinical Document Architecture
DICOM Part 21: Transformations between DICOM and other Representations
DICOM Parts 1-21: Bulk Download

KAITAI GRAMMARS: READABLE, CONCISE -> A GOOD STARTER FOR UNDERSTANDING



Kaitai's great IDE

(read-only file-wise, classic offset/hex/ascii view)

```
void bmp_t::file_header_t::_read() {
    m_magic = m__io->ensure_fixed_contents(std::string("\x42\x4D", 2));
    m_len_file = m__io->read_u4le();
    m_reserved1 = m__io->read_u2le();
    m_reserved2 = m__io->read_u2le();
    m_ofs_bitmap = m__io->read_s4le();
}
```

```
private void _read() {
    this.magic = this._io.ensureFixedContents(new byte[] { 66, 77 });
    this.lenFile = this._io.readU4le();
    this.reserved1 = this._io.readU2le();
    this.reserved2 = this._io.readU2le();
    this.ofsBitmap = this._io.readS4le();
}
```

```
private void _read() {
    _magic = m_io.EnsureFixedContents(new byte[] { 66, 77 });
    _lenFile = m_io.ReadU4le();
    _reserved1 = m_io.ReadU2le();
    _reserved2 = m_io.ReadU2le();
    _ofsBitmap = m_io.ReadS4le();
}
```

```
FileHeader.prototype._read = function() {
    this.magic = this._io.ensureFixedContents([66, 77]);
    this.lenFile = this._io.readU4le();
    this.reserved1 = this._io.readU2le();
    this.reserved2 = this._io.readU2le();
    this.ofsBitmap = this._io.readS4le();
}
```

```
private function _read() {
    $this->_m_magic = $this->_io->ensureFixedContents("\x42\x4D");
    $this->_m_lenFile = $this->_io->readU4le();
    $this->_m_reserved1 = $this->_io->readU2le();
    $this->_m_reserved2 = $this->_io->readU2le();
    $this->_m_ofsBitmap = $this->_io->readS4le();
}
```

```
def _read(self):
    self.magic = self._io.ensure_fixed_contents(b"\x42\x4D")
    self.len_file = self._io.read_u4le()
    self.reserved1 = self._io.read_u2le()
    self.reserved2 = self._io.read_u2le()
    self.ofs_bitmap = self._io.read_s4le()
```

```
sub _read {
    my ($self) = @_;
    $self->{magic} = $self->{_io}->ensure_fixed_contents(pack('C*', (66, 77)));
    $self->{len_file} = $self->{_io}->read_u4le();
    $self->{reserved1} = $self->{_io}->read_u2le();
    $self->{reserved2} = $self->{_io}->read_u2le();
    $self->{ofs_bitmap} = $self->{_io}->read_s4le();
}
```

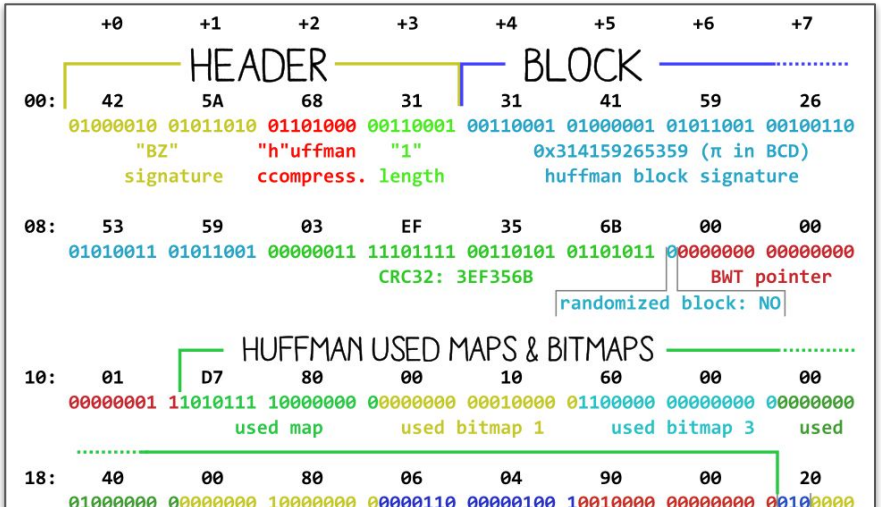
```
def _read
  @magic = @_io.ensure_fixed_contents([66, 77].pack('C*'))
  @len_file = @_io.read_u4le
  @reserved1 = @_io.read_u2le
  @reserved2 = @_io.read_u2le
  @ofs_bitmap = @_io.read_s4le
  self
end
```

Kaitai parser compiler

Kaitai limitations

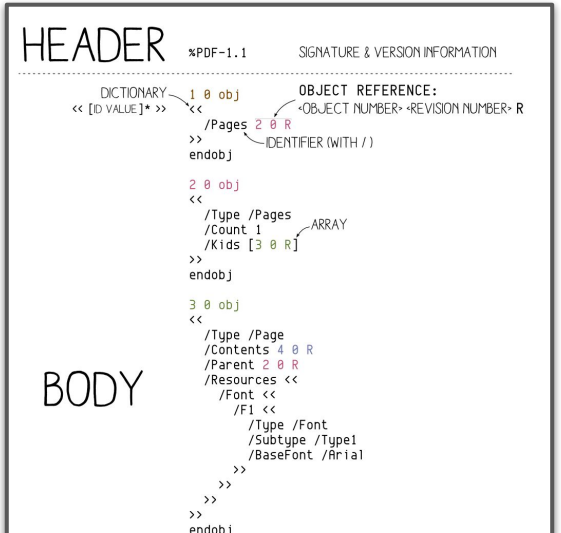
Not everything can be expressed with Yaml.

Mixed formats (PDF) or bit-level (BZip2) can't work.



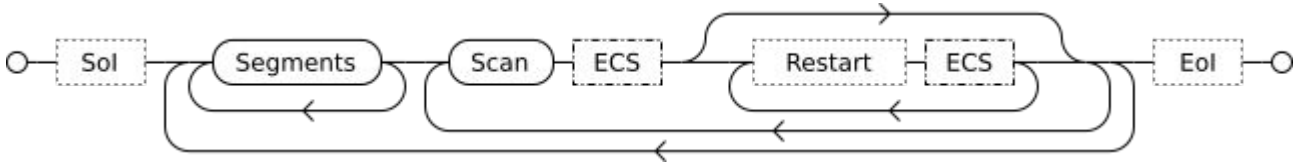
<= BZip2
(bit-based)

PDF =>
(Text skeleton)

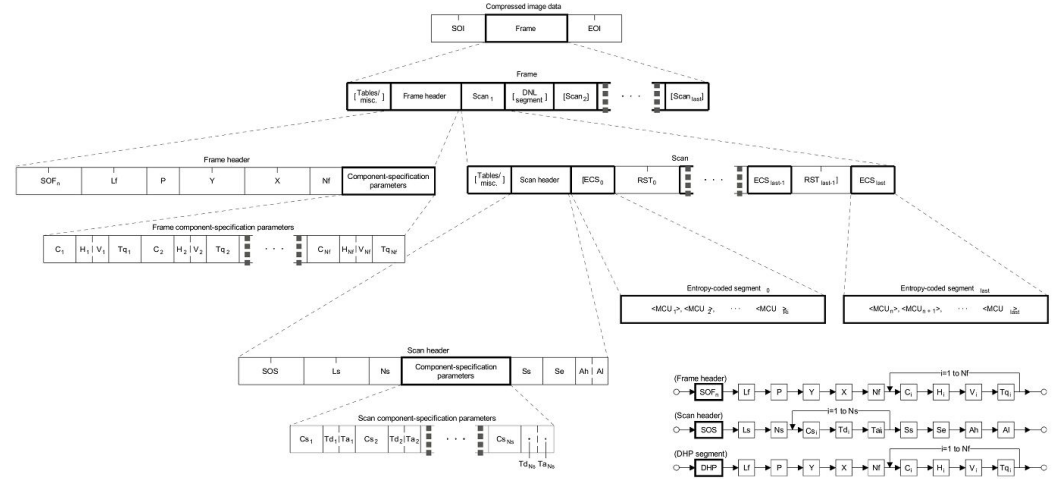


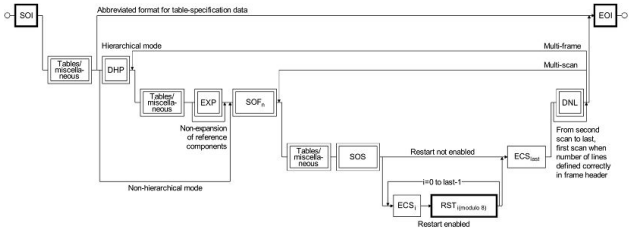
Syntax diagrams

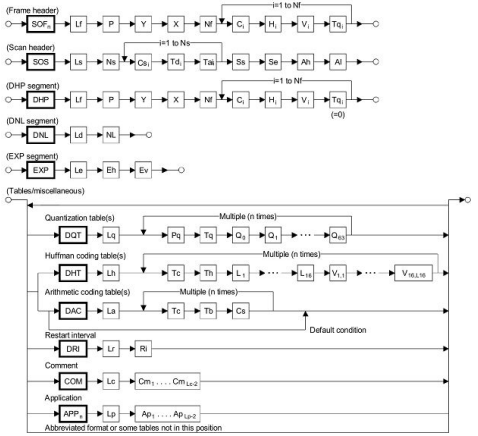
Very good to explain logic at various level.
Underrated, underused.



2 syntax diagrams of the same format (JPEG)



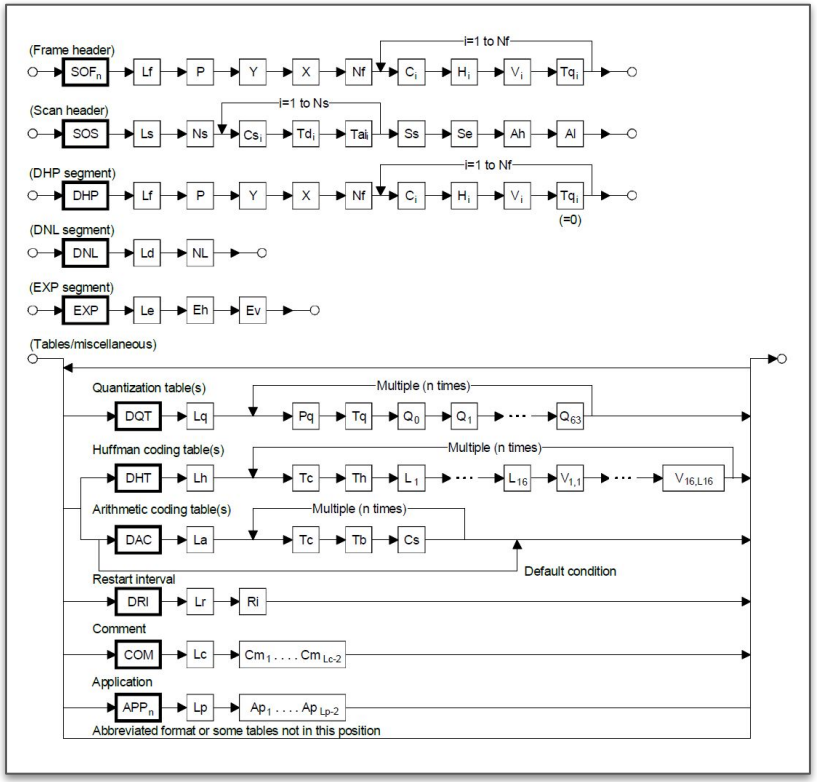


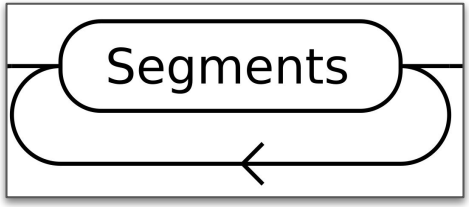


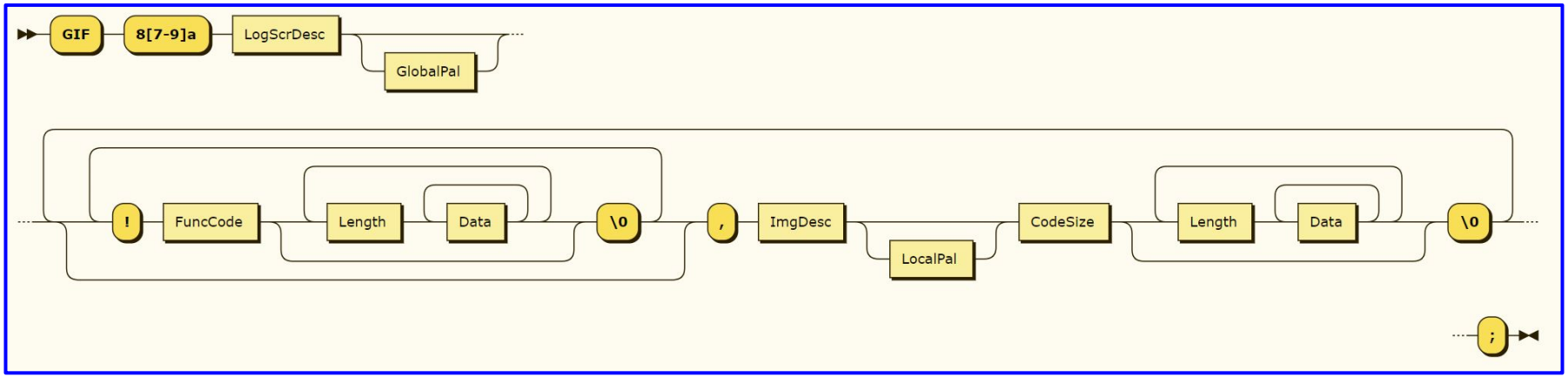
Different levels of details for different goals.

It can be useful to see every detail.

It can be overwhelming (and intimidating) and prevent us to grasp their generic structure.







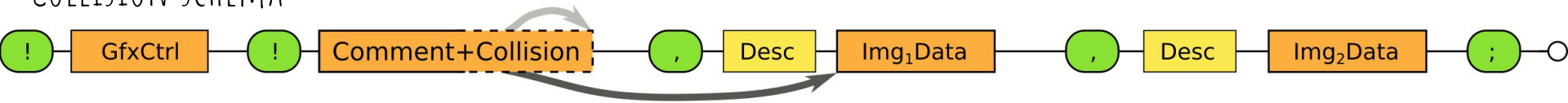
Already simplified, yet not so clear

-> you may miss some important points.

Useful to explain specific concepts.

Same color+shape = same data structur

Collision schema



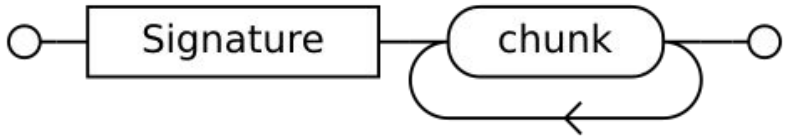
Short comment: comment stops before the first image.

Long comment: 1st image extended as a comment

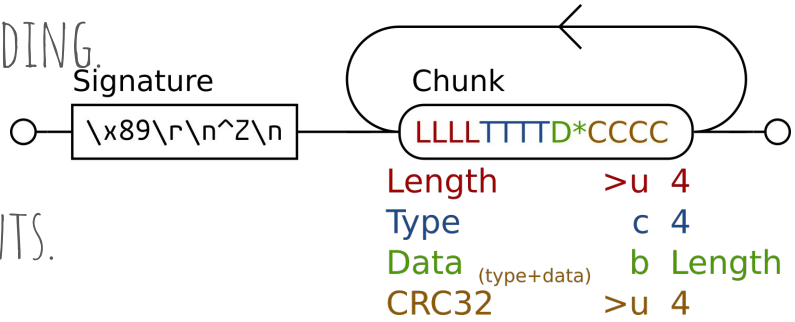
What do they lack? 1/2

Different views are needed:

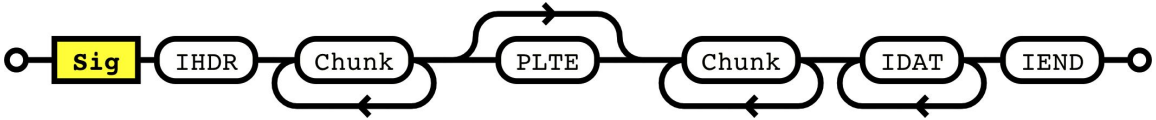
Sometimes, you need just the logic.



Sometimes, you need to explain the bytes and encoding.



Sometimes, you want to show the basic requirements.

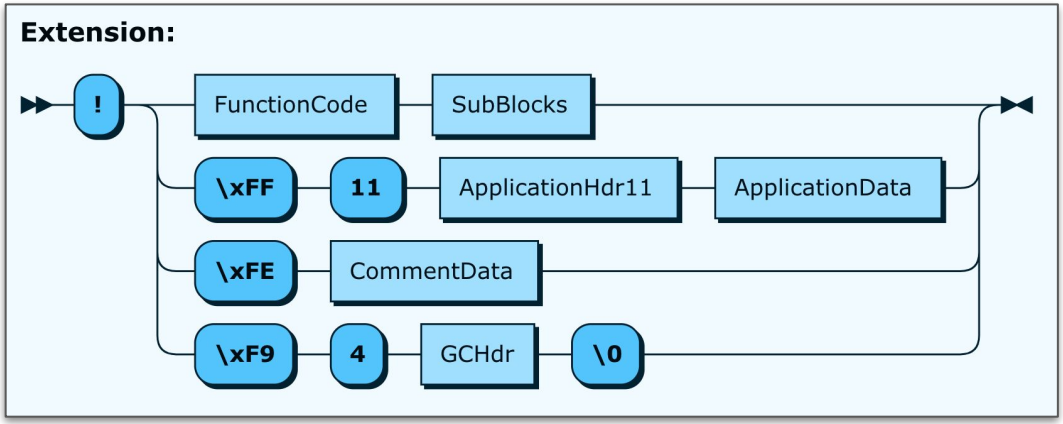


What do they lack? 2/2

No collapsable groups - that could be annotated.

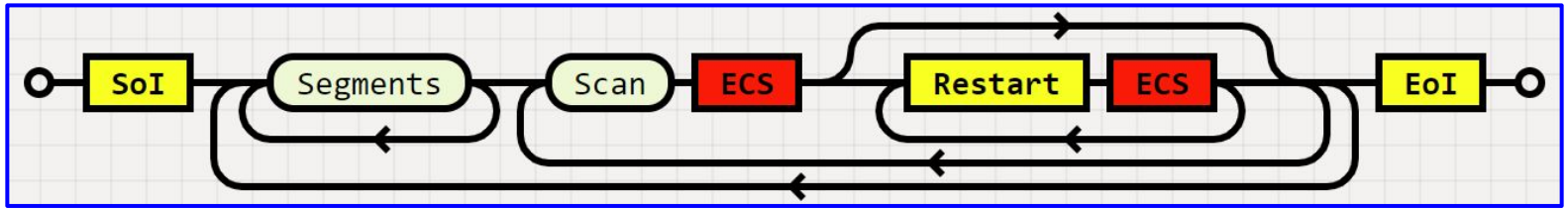
No relations between elements:

Ex: isPalettePresent bit then Palette array.



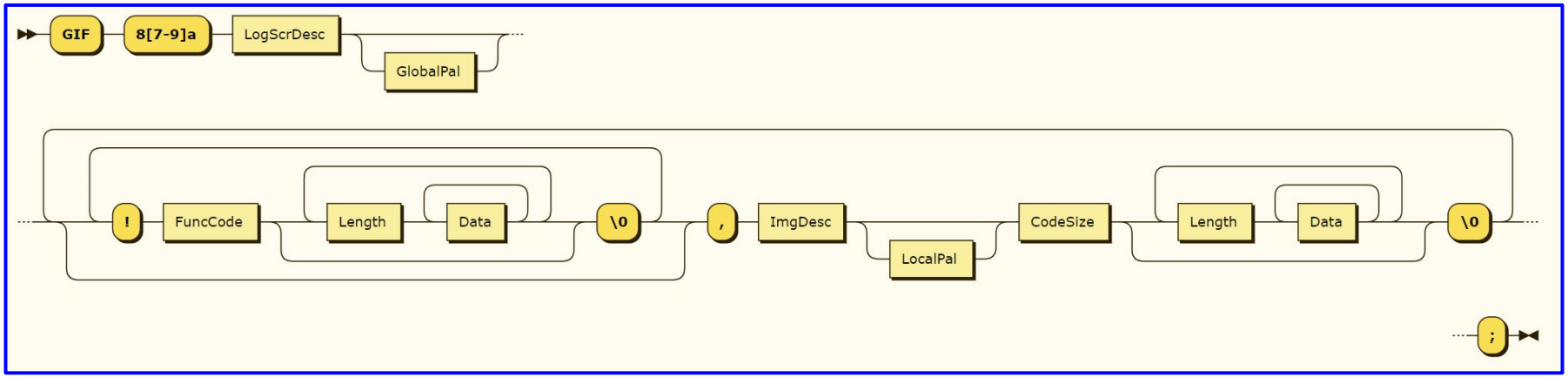
Some (limited but worth knowing) tools for syntax diagrams

JPEG




```
macro_rules! ECS {
   ($SoI:expr $( $( Segments )+ $( Scan $ECS:ty $( $Restart:expr $ECS:ty)* )+ )+ $EoI:expr ) => { ... };
}
```

GIF




```
File ::= 'GIF' '8[7-9]a' LogScrDesc GlobalPal? \
          (('!' FuncCode ( Length Data+)* '\0')* ',' ImgDesc LocalPal? CodeSize (Length Data+)* '\0')+ ';'
```

My own contributions...

(Individual effort in my spare time)

Better specs: first, better format. Then, rewrite.

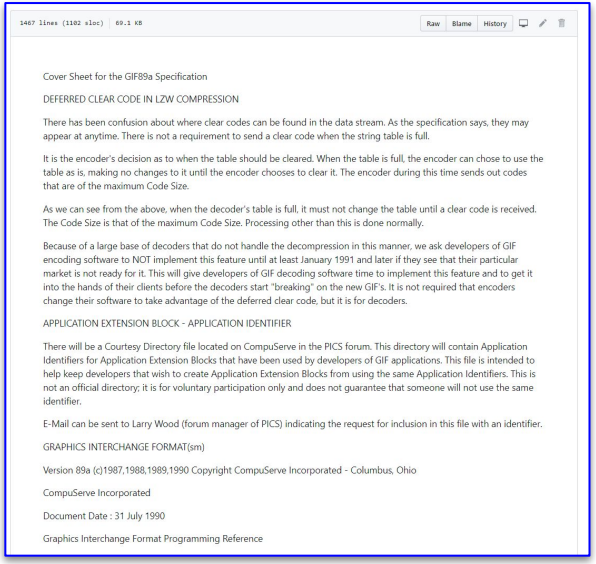
Specs are either ASCII-formatted, HTML, PDFs - not so convenient.

Reformatting as MarkDown with rendered elements

with parsable diagrams and logic.

```
GIFDataStream          ::= Header LogicalScreen Data* Trailer
LogicalScreen          ::= LogicalScreenDescriptor GlobalColorTable?
Data                   ::= GraphicBlock | SpecialPurposeBlock
GraphicBlock           ::= GraphicControlExtension? GraphicRenderingBlock
GraphicRenderingBlock  ::= TableBasedImage | PlainTextExtension
TableBasedImage        ::= ImageDescriptor LocalColorTable? ImageData
SpecialPurposeBlock    ::= ApplicationExtension | CommentExtension

Complete ::= Header LogicalScreenDescriptor GlobalColorTable? (GraphicControlExtension?
((ImageDescriptor LocalColorTable? ImageData) | PlainTextExtension) | (ApplicationExtension |
CommentExtension))* Trailer
```



1467 lines (1102 sloc) | 69.1 KB

Raw | Blame | History

Cover Sheet for the GIF89a Specification

DEFERRED CLEAR CODE IN LZW COMPRESSION

There has been confusion about where clear codes can be found in the data stream. As the specification says, they may appear at anytime. There is not a requirement to send a clear code when the string table is full.

It is the encoder's decision as to when the table should be cleared. When the table is full, the encoder can chose to use the table as is, making no changes to it until the encoder chooses to clear it. The encoder during this time sends out codes that are of the maximum Code Size.

As we can see from the above, when the decoder's table is full, it must not change the table until a clear code is received. The Code Size is that of the maximum Code Size. Processing other than this is done normally.

Because of a large base of decoders that do not handle the decompression in this manner, we ask developers of GIF encoding software to NOT implement this feature until at least January 1991 and later if they see that their particular market is not ready for it. This will give developers of GIF decoding software time to implement this feature and to get it into the hands of their clients before the decoders start "breaking" on the new GIF's. It is not required that encoders change their software to take advantage of the deferred clear code, but it is for decoders.

APPLICATION EXTENSION BLOCK - APPLICATION IDENTIFIER

There will be a Courtesy Directory file located on CompuServe in the PICS forum. This directory will contain Application Identifiers for Application Extension Blocks that have been used by developers of GIF applications. This file is intended to help keep developers that wish to create Application Extension Blocks from using the same Application Identifiers. This is not an official directory; it is for voluntary participation only and does not guarantee that someone will not use the same identifier.

E-Mail can be sent to Larry Wood (forum manager of PICS) indicating the request for inclusion in this file with an identifier.

GRAPHICS INTERCHANGE FORMAT(sm)

Version 89a (c)1987,1988,1989,1990 Copyright CompuServe Incorporated - Columbus, Ohio

CompuServe Incorporated

Document Date : 31 July 1990

Graphics Interchange Format Programming Reference

https://github.com/corkami/formats

Better binary visualisation

Scalable and readable hex representation that could be plugged to any parser

even w/ just dynamic instrumentation.

Outputs:

- ANSI text -> HTML / RTF / TeX
- CSS-less SVG -> PDF

```
Type:Png [file]                                          Field       Value
  000:  89 .P .N .G \r \n 1a \n                          +00 signature  \x89PNG\r\n\x1a\n
        0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f

Chunk: Image Header [chunk]                              Field    Value
  000:                          00 00 00 0D .I .H .D .R  +00 length  13
  010:  00 00 00 03 00 00 00 01 08 02 00 00 00 94 82 83  +04 type    IHDR
  020:  E3                                               +15 crc-32  0x948283e3
        0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f

Chunk: Image Data [chunk]                                Field    Value
  020:     00 00 00 15 .I .D .A .T 08 1D 01 0A 00 F5 FF  +00 length  21
  030:  00 FF 00 00 00 FF 00 00 00 FF 0E FB 02 FE E9 32  +04 type    IDAT
  040:  61 E5                                            +1d crc-32  0xe93261e5
        0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f

Chunk: Image End [chunk]                                 Field    Value
  040:           00 00 00 00 .I .E .N .D AE 42 60 82      +00 length  0
        0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f   +04 type    IEND
                                                         +08 crc-32  0xae426082
```

https://github.com/corkami/sbud

Better binary dissection+edition

Raw assembly (no opcodes) with many macros and structure decoration.

A new language could be better,
But the jump is not clearly needed yet.

https://github.com/corkami/sbud

```
db `\x89PNG\r\n\x1a\n`                          ; signature                 ;0000:

chunk1:                                         ; chunk1 { //Image Header
 ddbe 13                                        ;  length                   ;0008:
;ddbe (chunk1.crc32 - chunk1.data)
.type db `IHDR`                                 ;  type                     ;000c:
.data:                                          ; Data {
  incbin 'rgb.png', 0x10, 0xd                                               ;0010:
  ;}                                            ;   } //Data
.crc32 ddbe 0x948283e3                          ;  crc-32                   ;001d:
 ;> chunk1.crc32=CRC32(chunk1.type,chunk1.crc32)
;}                                              ; } //chunk

chunk2:                                         ; chunk2 { //Image Data
 ddbe 21                                        ;  length                   ;0021:
;ddbe (chunk2.crc32 - chunk2.data)
.type db `IDAT`                                 ;  type                     ;0025:
.data:                                          ; Data {
  incbin 'rgb.png', 0x29, 0x15                                              ;0029:
  ;}                                            ;   } //Data
.crc32 ddbe 0xe93261e5                          ;  crc-32                   ;003e:
 ;> chunk2.crc32=CRC32(chunk2.type,chunk2.crc32)
;}                                              ; } //chunk

chunk3:                                         ; chunk3 { //Image End
 ddbe 0                                         ;  length                   ;0042:
;ddbe (chunk3.crc32 - chunk3.data)
.type db `IEND`                                 ;  type                     ;0046:
.data:
.crc32 ddbe 0xae426082                          ;  crc-32                   ;004a:
 ;> chunk3.crc32=CRC32(chunk3.type,chunk3.crc32)
;}                                              ; } //chunk
```

No "Executable" GUI please!

GUIs give fancy representation easily,

but then we're left with ugly screenshots.

-> better output parseable/reusable format from the beginning

Eventually with an interactive webpage

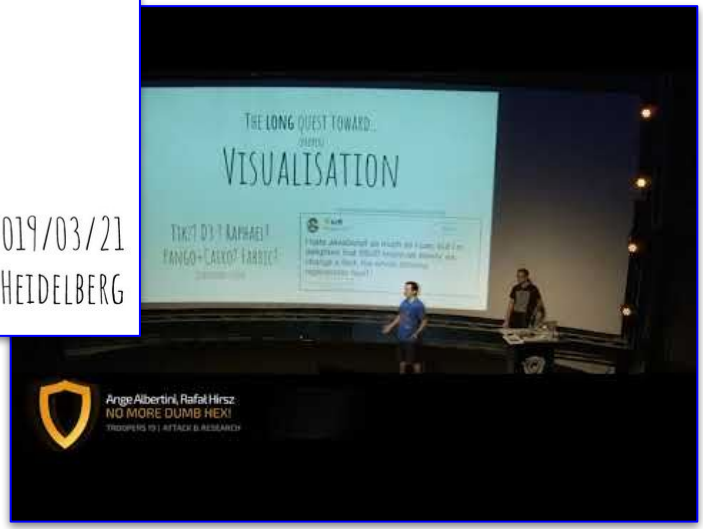
and showing a rendering in the browser.

No more dumb hex!

Rethinking binary tooling

Rafał Hirsz
Ange Albertini

2019/03/21
Troopers, Heidelberg



More info @ https://speakerdeck.com/ange/no-more-dumb-hex

PDF: HERE BE DRAGONS

PDF has a lot of hard problems such as..

Whitespace in PDF (all readers don't agree)

4.42
space character
text string character used to represent orthographic white space in text strings

NOTE 2 space characters include HORIZONTAL TAB (U+0009), LINE FEED (U+000A), VERTICAL TAB (U+000B), FORM FEED (U+000C), CARRIAGE RETURN (U+000D), SPACE (U+0020), NOBREAK SPACE (U+00A0), EN SPACE (U+2002), EM SPACE (U+2003), FIGURE SPACE (U+2007), PUNCTUATION SPACE (U+2008), THIN SPACE (U+2009), HAIR SPACE (U+200A), ZERO WIDTH SPACE (U+200B), and IDEOGRAPHIC SPACE (U+3000)

4.46
white-space character
characters that separate PDF syntactic constructs such as names and numbers from each other; white space characters are HORIZONTAL TAB (09h), LINE FEED (0Ah), FORM FEED (0Ch), CARRIAGE RETURN (0Dh), SPACE (20h); (see Table 1 in 7.2.2, “Character Set”)

HEADER

```
%PDF-1.1
```

SIGNATURE & VERSION INFORMATION

BODY

DICTIONARY `<< [ID VALUE]* >>`

OBJECT REFERENCE: `<OBJECT NUMBER> <REVISION NUMBER> R`

IDENTIFIER (WITH /)

ARRAY

STREAM PARAMETERS: LENGTH, COMPRESSION.....

STRING

```
1 0 obj
<<
  /Pages 2 0 R
>>
endobj

2 0 obj
<<
  /Type /Pages
  /Count 1
  /Kids [3 0 R]
>>
endobj

3 0 obj
<<
  /Type /Page
  /Contents 4 0 R
  /Parent 2 0 R
  /Resources <<
    /Font <<
      /F1 <<
        /Type /Font
        /Subtype /Type1
        /BaseFont /Arial
      >>
    >>
  >>
>>
endobj

4 0 obj
<< /Length 50 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!)Tj
ET
endstream
endobj
```

BEGIN TEXT
FONT F1 (ARIAL) SET TO SIZE 110
MOVE TO COORDINATE 10, 400
OUTPUT TEXT "HELLO WORLD!"
END TEXT

WHAT A NORMAL PDF USUALLY LOOKS LIKE.

XREF TABLE

CROSS REFERENCE

```
xref
0 5
0000000000 65535 f
0000000010 00000 n
0000000047 00000 n
0000000111 00000 n
0000000313 00000 n
```

CROSS REFERENCES
5 OBJECTS, STARTING AT INDEX 0
(STANDARD FIRST EMPTY OBJECT 0
OFFSET TO OBJECT 1, REV 0
TO OBJECT 2...
3...
4

TRAILER

```
trailer
<<
  /Root 1 0 R
>>

startxref
413
%%EOF
```

What a weird PDF can look like.

This one works fine
with all readers
without any warning.

No XREF, no /Length, no /Size

```
%PDF-1.3
1 0 obj<</Type/Catalog/Pages 2 0 R>>endobj
2 0 obj<</Type/Pages/Count 1/Kids[3 0 R]>>endobj
3 0 obj<</Type/Page/Contents 4 0 R/Parent 2 0
R/Resources<</Font<</F<</Type/Font/Subtype/Type1/Base
Font/Arial>>>>>>>>endobj
4 0 obj<<>>stream
BT/F 55 Tf 10 400 Td(http://www.corkami.com)' ET
endstream
endobj
trailer <</Root 1 0 R>>
```

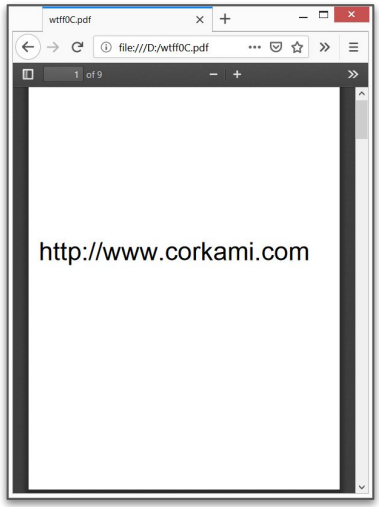
What a crazy PDF can look like....

```
\t1\t0\tobj<</Resources<</Font<</<</BaseFont//Subtype/>>>>>>/Contents<<>>stream\n
/\t50Tf20\r450Td(http://www.corkami.com)Tjendstream>>endobj\x20
trailer<</Root<</Pages<</Kids[1\t0R]/Count\f9
```

This is a valid PDF for firefox.

It breaks so many rules, and yet...

it works without any warning!



```
\t1\t0\tobj<</Resources<</Font<</<</BaseFont/Subtype/>>>>>>/Contents<<>>stream\n
/\t50Tf20\r450Td(http://www.corkami.com)Tjendstream>>endobj\x20
trailer<</Root<</Pages<</Kids[1\t0R]/Count\f9
```

No %PDF signature,no Type, no Parent...

Mixed whitespace. Empty font name, BaseFont, Subtype.

Recursive & inline stream object. Non-closed dictionaries.

No whitespace between keywords and numbers.

9 pages counted but only **1** kid.

We really have a lot of cleaning to do...

```
$ mutool clean wtff0C.pdf
error: cannot recognize version marker
warning: trying to repair broken xref
error: invalid key in dict
error: cannot parse dict
error: invalid indirect reference in dict
error: cannot parse dict
error: cannot parse dict
error: cannot parse dict
error: invalid key in dict
error: cannot parse dict
error: cannot load object (1 0 R) into cache
warning: ignoring broken object (1 0 R)
error: invalid key in dict
error: cannot parse dict
error: cannot load object (1 0 R) into cache
warning: cannot load object (1 0 R) into cache

$ qpdf wtff0C.pdf repaired.pdf
WARNING: wtff0C.pdf: can't find PDF header
WARNING: wtff0C.pdf: file is damaged
WARNING: wtff0C.pdf: can't find startxref
WARNING: wtff0C.pdf: Attempting to reconstruct cross-reference table
wtff0C.pdf: unable to find trailer dictionary while recovering damaged file

$
```

Output from `mutool`:

```
%PDF-0.0
%%µÕ

1 0 obj
null
endobj
xref
0 2
0000000000 65536 f
0000000018 00000 n

trailer
<</Size 2>>

startxref
38
%%EOF
```

This crazy PDF can't be repaired with standard tools.

Conclusion

A REALLY LONG WAY TO GO...

https://reference.pdfa.org/iso/32000/

Adobe documents referenced from ISO 32000-2:2017

This page lists Adobe documents normatively referenced from ISO 32000-2:

PostScript Language Third Edition (February, 1999), Adobe Systems Incorporated

Adobe Acrobat 3D JavaScript Reference, Version 9.1 (August 2009), Adobe Systems Incorporated **(to be added)**

Adobe Glyph List, 2015, Adobe Systems Incorporated **(to be added)**

Adobe Glyph List for New Fonts, Version 1.6, (January 2006), Adobe Systems Incorporated **(to be added)**

Adobe JavaScript for Acrobat API Reference, Version 9.1, (August 2009), Adobe Systems Incorporated **(to be added)**

Adobe PDF Signature Build Dictionary Specification v.1.4, (March 2008), Adobe Systems Incorporated

Adobe TIFF Revision 6.0, Final, (June 1992), Adobe Systems Incorporated

Adobe Type 1 Font Format, Version 1.1, (February 1993), Addison-Wesley, ISBN 0-201-57044-01

Adobe XML Architecture, XML Forms Architecture (XFA) Specification, version 3.3, (January 2012), Adobe Systems Incorporated

Adobe XMP: Extensible Metadata Platform, Part 2: Standard Schemas, (July 2010), Adobe Systems Incorporated. **(to be added)**

Adobe XMP: Extensible Metadata Platform, Part 3: Storage in Files, (July 2010), Adobe Systems Incorporated. **(to be added)**

Adobe Technical Note #5014, *Adobe CMap and CID Font Files Specification, Version 1.0*, (8 October 1996), Adobe Systems Incorporated

Adobe Technical Note #5078, *Adobe-Japan1-6 Character Collection for CID-Keyed Fonts*, (February, 2008), Adobe Systems Incorporated

Adobe Technical Note #5079, *Adobe-GB1-5 Character Collection for CID-Keyed Fonts*, (January 2012), Adobe Systems Incorporated

Adobe Technical Note #5080, *Adobe-CNS1-6 Character Collection for CID-Keyed Fonts*, (January 2012), Adobe Systems Incorporated

Adobe Technical Note #5093, *Adobe-Korea1-2 Character Collection for CID-Keyed Fonts*, (January 2012), Adobe Systems Incorporated

Adobe Technical Note #5097, *Adobe-Japan2-0 Character Collection for CID-Keyed Fonts*, (May 2003), Adobe Systems Incorporated

Adobe Technical Note #5116, *Supporting the DCT Filters in PostScript Level 2*, (November 1992), Adobe Systems Incorporated

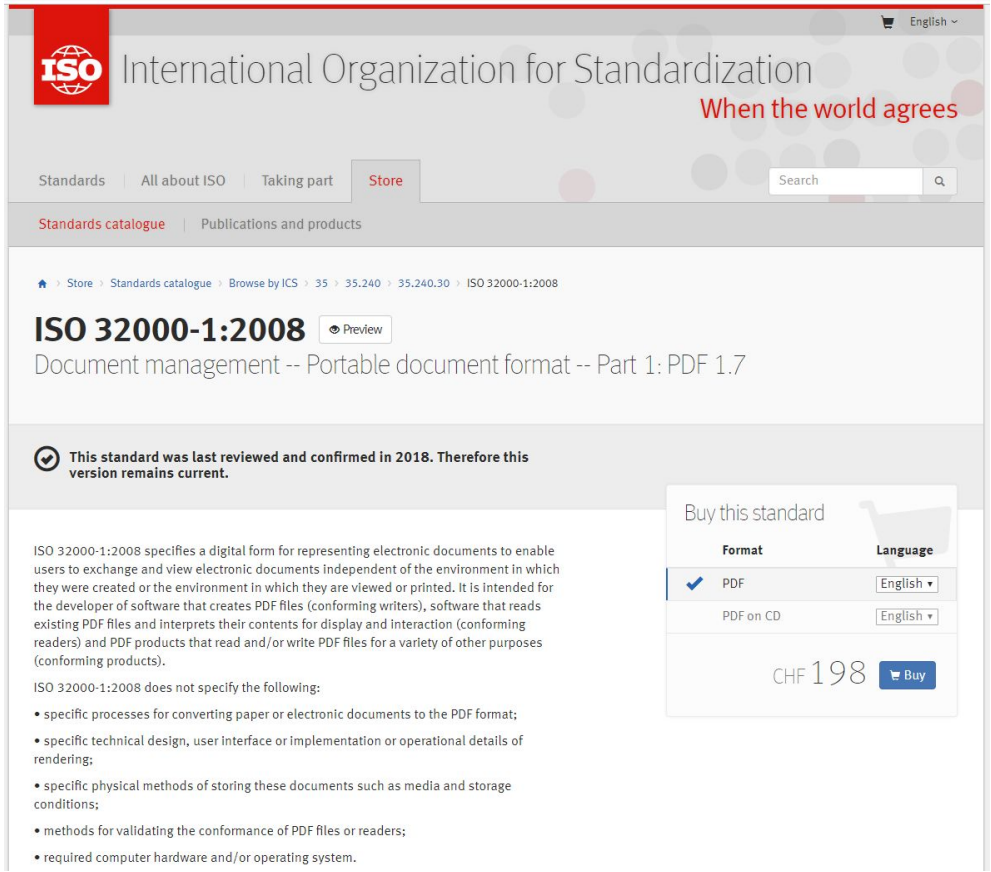
Adobe Technical Note #5176, *The Compact Font Format Specification, version 1.0*, (December 2003), Adobe Systems Incorporated

Adobe Technical Note #5177, *The Type 2 Charstring Format*, (March 2000), Adobe Systems Incorporated

Adobe Technical Note #5620, *Portable Job Ticket Format, Version 1.1*, (April 1999), Adobe Systems Incorporated

Adobe Technical Note #5660, *Open Prepress Interface (OPI) Specification, Version 2.0*, (January 2000), Adobe Systems Incorporated

Warnock, John, *The Camelot Project*, Adobe Systems 1990

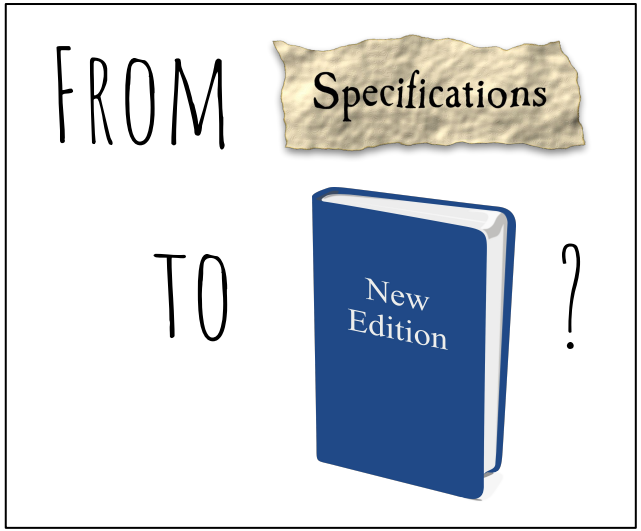



```
tests/
  valid
  extreme
  obsolete
  invalid
  bug
  exploit
tools/
  reader
  writer
  visualizer
  fuzzer
  sanitizer
docs/
  specs
  grammars
```

What we want in the future...? VeraPDF^X

Thank you!

Any feedback?



ANGE ALBERTINI
reverse engineering
VISUAL DOCUMENTATIONS
@angealbertini
ange@corkami.com
http://www.corkami.com