

FROM Specifications

TO ?



IMPROVING FILE FORMATS

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...

SCIENCE & VIE MICRO

SVM

LE N° 1 DE LA PRESSE INFORMATIQUE

REDÉCOUVRIR
LE LOGICIEL INTÉGRÉ :
TRAVAUX PRATIQUES AVEC WORKS

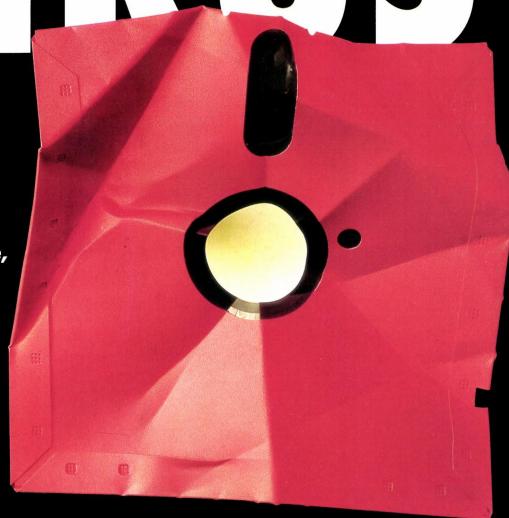
UNE RENCONTRE DÉCAPANTE
AVEC BILL GATES,
PATRON DE MICROSOFT

MACINTOSH IIci
LE CHEF D'ŒUVRE D'APPLE

L'AFFAIRE DES

VIRUS

Que s'est-il
vraiment passé
le vendredi
13 octobre ?
Une hallucination
collective
à l'échelle mondiale,
ou les prémisses
d'une réelle
catastrophe
technologique ?
Une analyse
à la loupe
de la vie
et de la mort
des virus
informatiques



NOVEMBER 1989 160 FR - 6,50 FS - \$ 8,50 - 450 Pcs - 250 Es - 28 Dh - 2,000 Dt - RC - 1,680 CFA - USA NYC \$ 4,25 - ISSN 0760-6516

N°66

...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.!

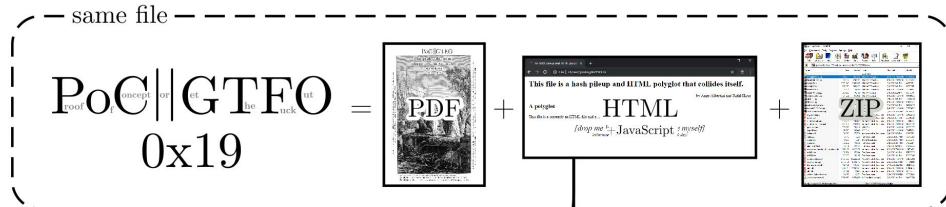
comme endommages dans la table d'allocation des fichiers. Par chance, il n'attaque que les IBM PC-XT. Pour s'en débarrasser, il faut rétablir les pistes de démarrage dans leur état d'origine. Avec un éditeur d'octets du type PC-Tools, vérifiez la présence des octets 33 C0 dans les zones 30 et 31 du secteur 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

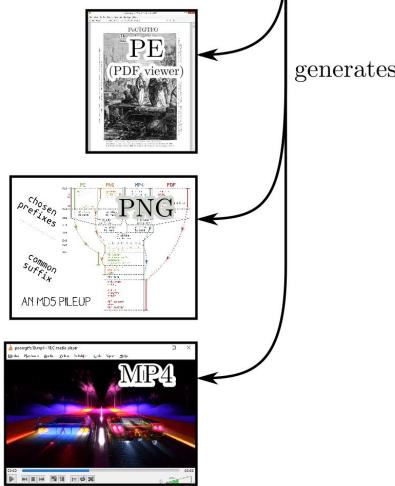
- 13 YEARS OF MALWARE ANALYSIS
- NOW INFORMATION SECURITY ENGINEER



NOTE: THIS TALK REFLECTS MY OWN OPINION, NOT MY EMPLOYER.

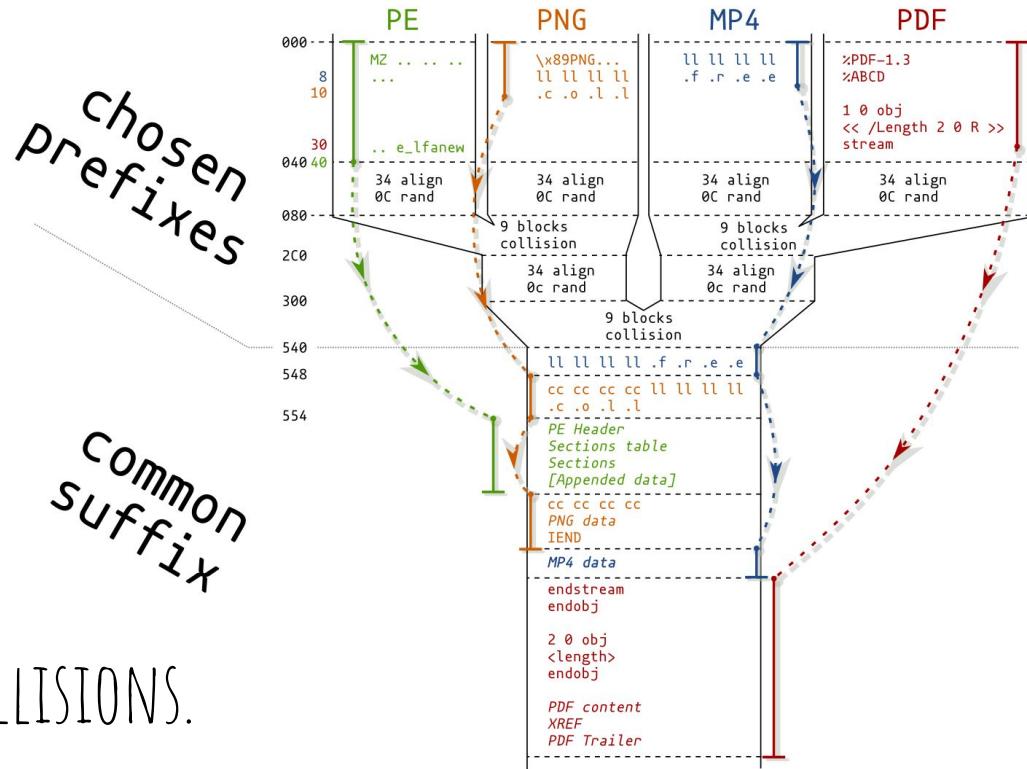


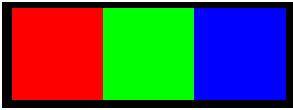
All these files have the **same** MD5



MY LATEST CREATION:
6 FILE TYPES, 4 PREFIXES, 3 HASHES COLLISIONS.

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





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.

Copyright © 2003 W3C® (MIT, ERCIM, Keio). All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

rgb.png/header					
address	name	type	size	data	description
00000000.0	size	UInt32	00000004.0	13	Size
0000000c.0	tag	FixedSize<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

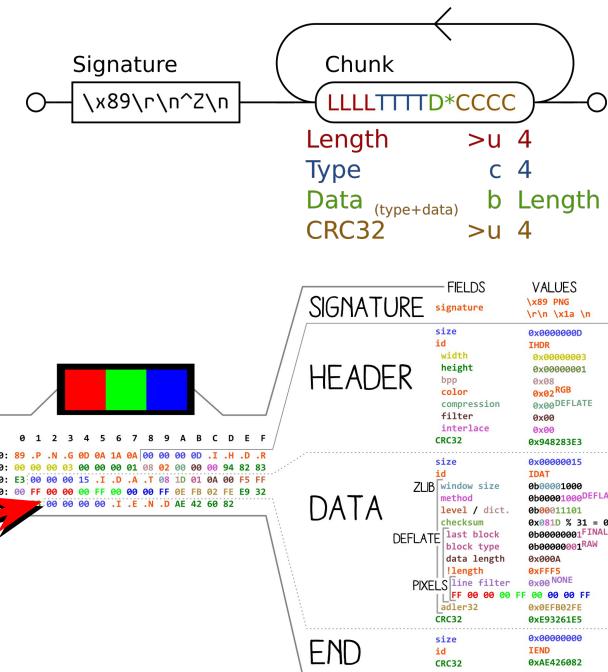
```

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",
                       binascii.crc32(type_ + data)
                         % 0x100000000)
        ]
    return "".join(s)

with open("rgb.png", "wb") as f:
    f.write(make([
        ["IHDR", # Image Header
         struct.pack(">I" * 6,
                     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
             ,), # no compression
        ["IEND", # Image End
         ""], # Empty chunk
    ]))

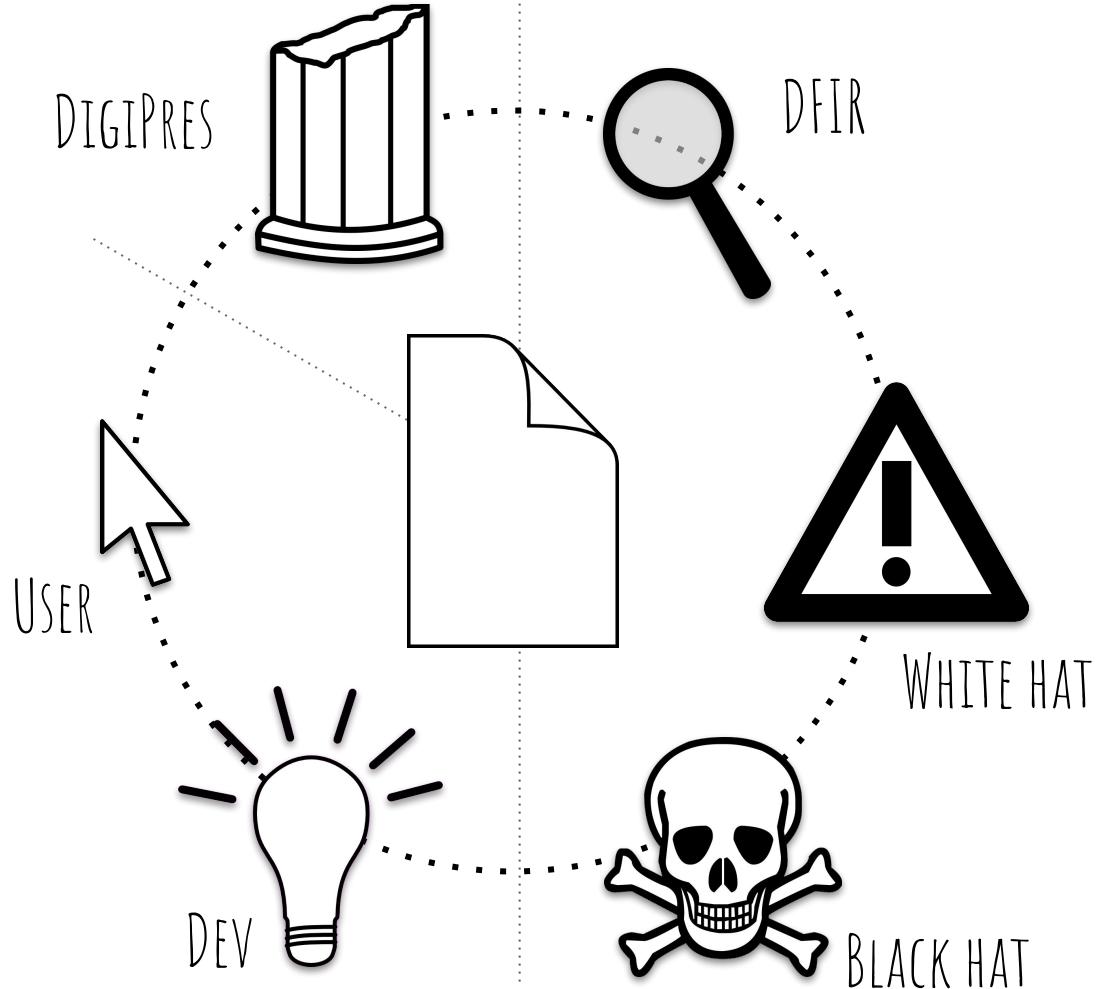
```



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

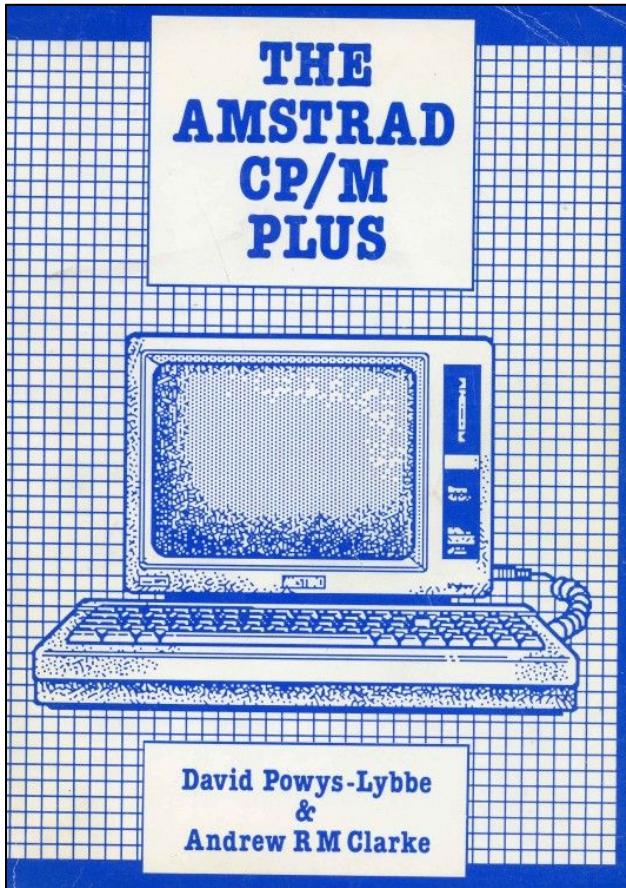
**Ready
ICPM**

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!

CP/M 2.2 - Amstrad Consumer Electronics plc

A>ED GO.COM

NEW FILE
: *e

CREATE AN EMPTY FILE

A>STAT GO.COM

Recs	Bytes	Ext	Acc	
0	0k	1	R/W	A:GO.COM
Bytes Remaining On A:	5k			

SIZE=0

A>

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)

A>STAT GO.COM

Recs	Bytes	Ext	Acc
0	0k	1	R/W A:GO.COM
Bytes Remaining On A: 5k			

A>GO GO.COM

Recs	Bytes	Ext	Acc
0	0k	1	R/W A:GO.COM
Bytes Remaining On A: 5k			

A>■

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 00 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.

CRYPTO

FILE FORMATS

NOT ENOUGH PRE-NATAL CHECKS.

LACKING GROWTH CONTROL.



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...

veraPDF test corpus for ISO 19005 (PDF/A)

257 commits 2 branches 0 releases 5 contributors

Branch: staging New pull request Find File Clone or download

DmitryRemezov and bdoubrov Resolve issue with incorrect value of /Count entry (#108) ... Latest commit 3dab890 on Nov 1, 2017

ISO 32000-1	Resolve issue with incorrect value of /Count entry (#108)	2 years ago
PDF_A-1a	Modify name of "CMAP in A025-pdfa2-pass-a" (#94)	2 years ago
PDF_A-1b	Resolve issue with incorrect value of /Count entry (#108)	2 years ago
PDF_A-2b	Resolve issue with incorrect value of /Count entry (#108)	2 years ago
PDF_A-2u/6.2 Graphics/6.2.11 Fonts/6....	Modify test files from 6.2.11.7.2 clause (#97)	2 years ago
PDF_A-3b/6.8 Embedded files	Modify name of "CMAP in A025-pdfa2-pass-a" (#94)	2 years ago
TWG test files	Resolve issue with incorrect value of /Count entry (#108)	2 years ago
Undefined	Move files from clause 6.6.2.3.2 to directory undefined (#107)	2 years ago
README.md	Update README.md	4 years ago

PDF: ADOBE 1993 VERAPDF: ISO 2014

Definitive PDF/A Validation



veraPDF is a purpose-built, open source, file-format validator covering all PDF/A parts and conformance levels.

veraPDF is designed to meet the needs of digital preservationists and is supported by the PDF software developer community.

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.

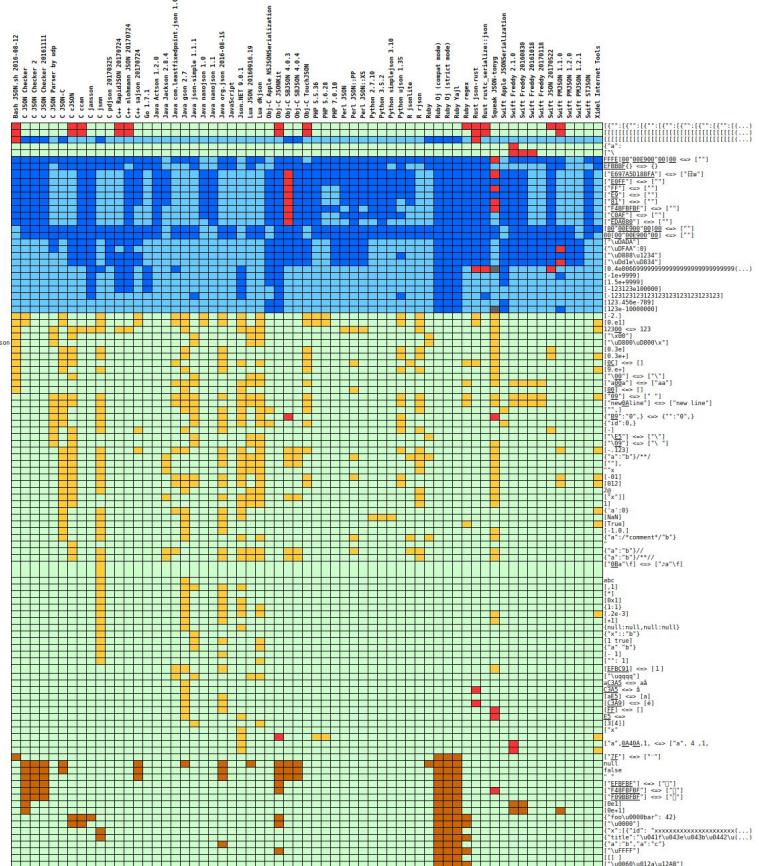
NICOLAS SERIOT'S JSON PARSERS ANALYSIS

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

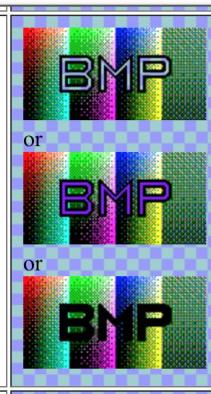
Format	GNU + base-256										pax												
Implementation	p	s	u	m	n	r	l	y	v	p	s	u	g	m	n	t	a	f	x	r	l	v	y
GNU tar	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
libarchive	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
star	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
NetBSD pax	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗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

EXPECTATIONS AND REALITY



THERE ARE ALWAYS UNKNOWN UNKNOWNS.

UPGRADE



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**

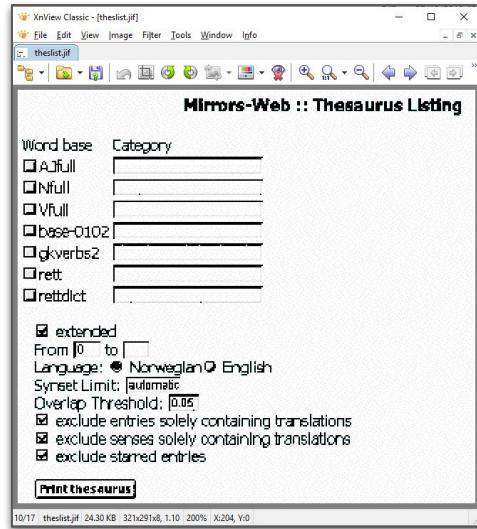
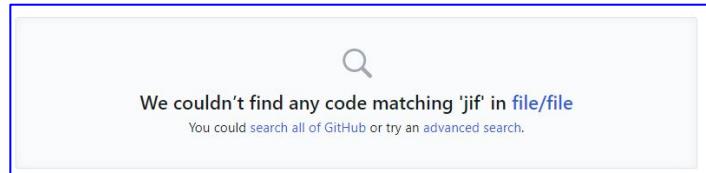
UNKNOWN TO VIRUSTOTAL

A SINGLE FILE, THAT I UPLOADED RECENTLY.

BUT IT'S SUPPORTED BY XNVIEW

-> DEPRECATION IS VERY HARD.

-> INFOSEC DOESN'T OVERLAP WITH DIGIPRES.



DEPRECATION? FEAR, UNCERTAINTY, DOUBT.

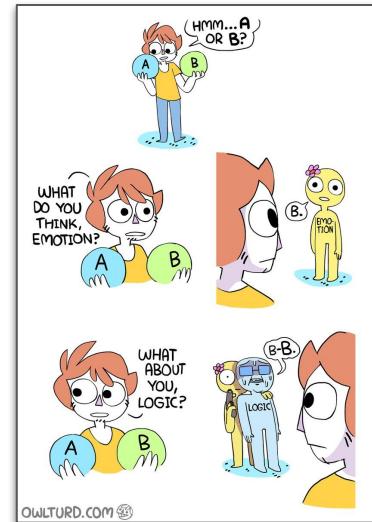
GIF DEPRECATION == "NO MORE MEMES/CAT PICS"? -> IRRATIONALITY

FIGHT IRRATIONALITY WITH 'DATA-DRIVEN EXPLANATIONS'.

-> DOCUMENTATIONS AND METRICS.

WHICH, FOR NOW, MEANS JUST "ORIGINAL SPECS".

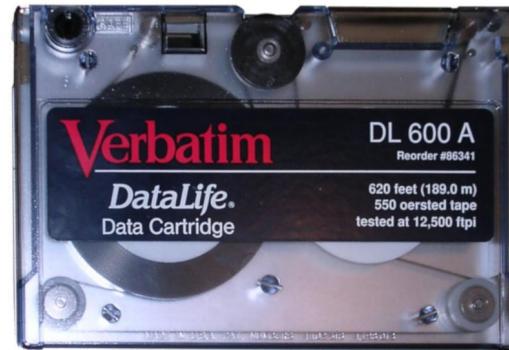
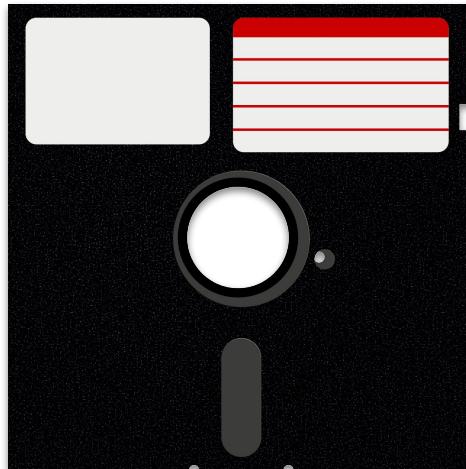
(THAT ARE 30+ YEAR OLD)



YET WE STILL USE TAPE/FLOPPIES ORIENTED FEATURE!

WE CAN'T KILL ZIP/TAR.

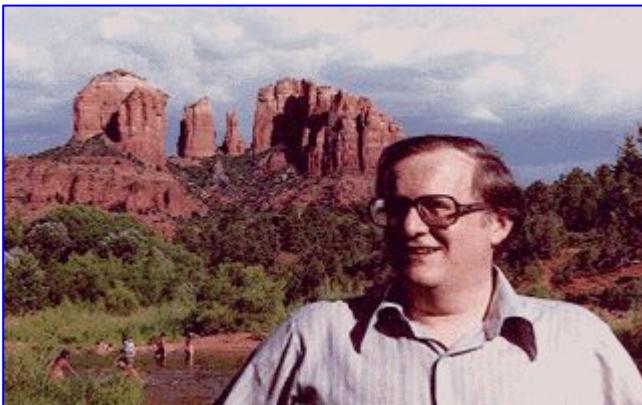
BECAUSE OF NO VISIBILITY OR WAY TO ENFORCE A SUCCESSOR.



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.

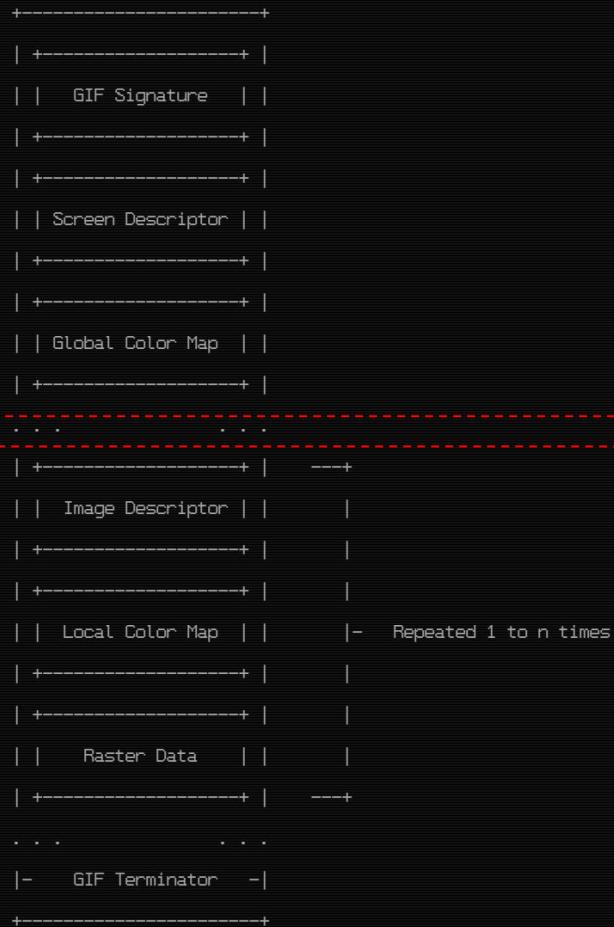
SOME FEATURES ARE LOST

OR NEVER IMPLEMENTED.

NOVELTIES FROM 1989



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:

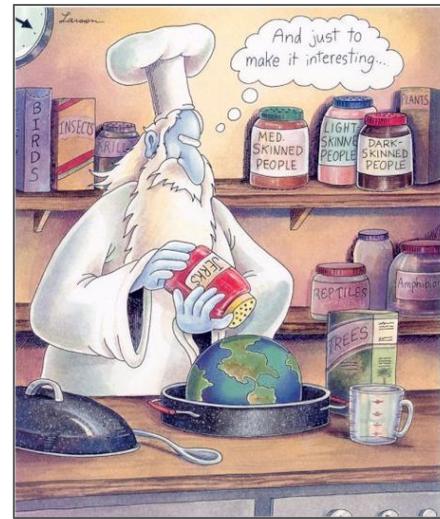
PROPER PLANNING. OFFICIAL SPECS. SET IN STONE.

BAD THINGS HAPPEN:

INTERPRETATION BLUR, UNOFFICIAL EXTENSIONS.

FORMAT IS NOW USED EVERYWHERE:

MISUNDERSTOOD. UNMOVABLE.

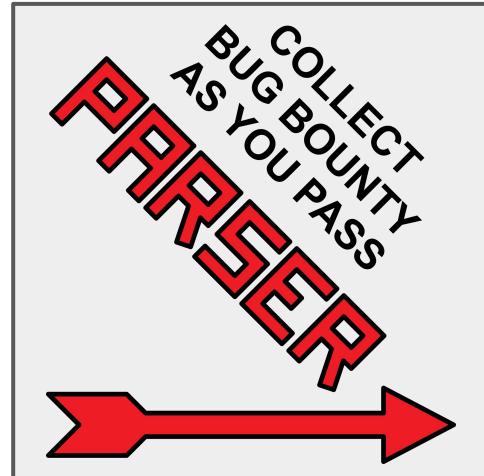


A NEW (VERSION OF A) PARSER IS OUT?

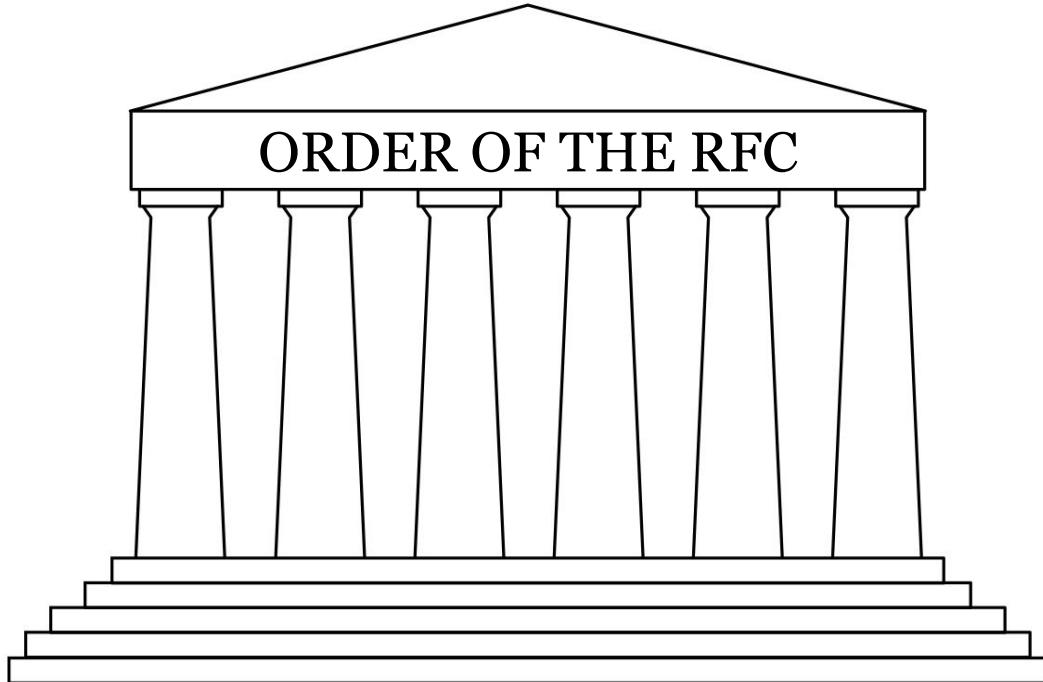
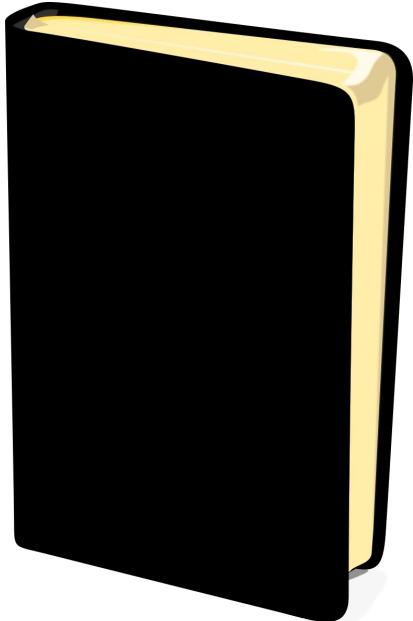
FUZZ. GET BUG FIXED. COLLECT PRIDE & GLORY.

RINSE. REPEAT.

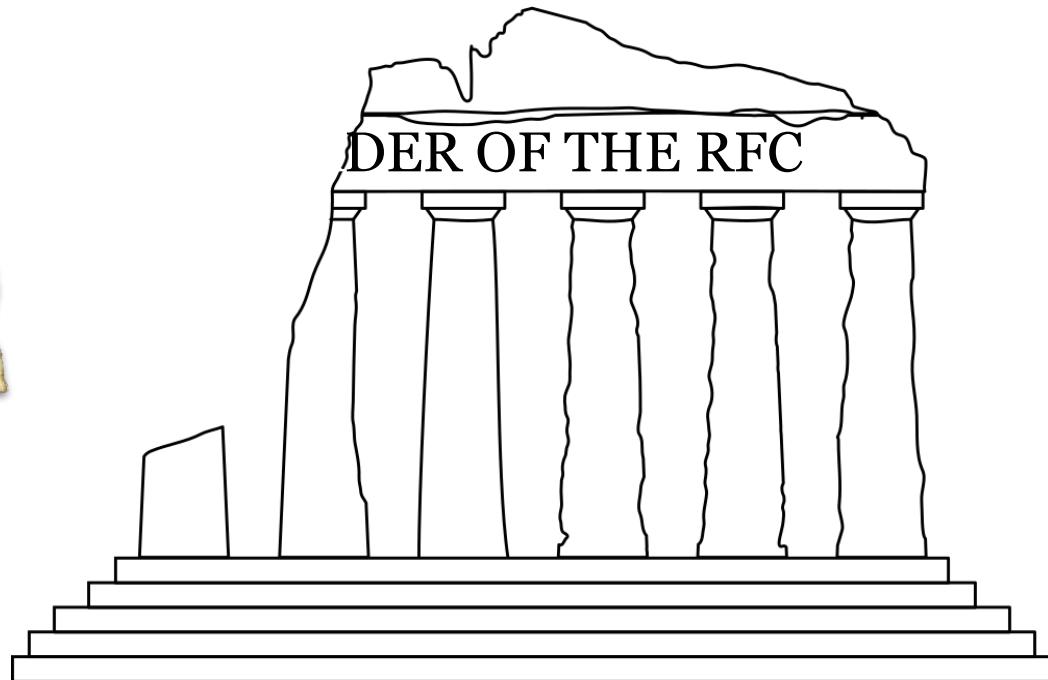
```
10 ParserUpdate
20 Fuzz
30 Fail
40 Collect
50 GOTO 10
```



HOW WE PERCEIVE FILE FORMATS: A HOLY TEXT AND ITS CULT.



MORE LIKE...
OUTDATED AND IRRELEVANT PRACTICES.



Specifications

WHAT WE HAVE

(WHAT WE'RE LEFT WITH)

SH*T MY SPECSSAYS
(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.

▲ Acredit 2.1k points · 3 years ago 

▼ What is the worst piece of software you've worked on and why is it Lotus Notes?

Share Report Save

```
49 #define MAX_COMPONENTS_IN_SCAN 4 /* JPEG limit on # of components in one scan */
50 #define MAX_SAMP_FACTOR 4 /* JPEG limit on sampling factors */
51 /* Unfortunately, some bozo at Adobe saw no reason to be bound by the standard;
52 * the PostScript DCT filter can emit files with many more than 10 blocks/MCU.
```

```
101 off_t nextchunk=[m_offsetinfile]+((chunklen+3)&~3);
102 // At this point, I'd like to take a moment to speak to you about the Adobe PSD format.
103 // PSD is not a good format. PSD is not even a bad format. Calling it such would be an
104 // insult to other bad formats, such as PCX or JPEG. No, PSD is an abysmal format. Having
105 // worked on this code for several weeks now, my hate for PSD has grown to a raging fire.
```

WHAT WE SEE...

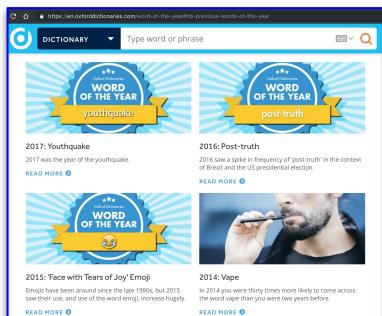
SH*T MY PARSER SAYS

ENCYCLOPEDIA OF GRAPHICS FILE FORMATS

A 'GOOD' REFERENCE BUT:

- OUTDATED (1996).
- DOESN'T REFLECT THE CURRENT LANDSCAPE.

OXFORD DICTIONARY: STILL FRESH



GEM VDI (cont'd)

have first purchased the GEM Programmer's Toolkit from Digital Research. You can contact DISCUS at:

DISCUS Distribution Services, Inc.
8020 San Miguel Canyon Road
Salinas, CA 93907-1208
Voice: 408-663-6966

You may still be able to get some information from Novell/Digital Research at:

Novell/Digital Research, Inc.
P.O. Box DR
Monterey, CA 93942
Voice: 408-649-3896
Voice: 800-848-1498
BBS: 408-649-3896

We have also been able to include information on Atari support of GEM VDI on the CD-ROM that accompanies this book.

GIF

NAME: GIF
ALSO KNOWN AS: Graphics Interchange Format
TYPE: Bitmap
COLORS: 1 to 8 bit
COMPRESSION: LZW
MAXIMUM IMAGE SIZE: 64Kx64K pixels
MULTIPLE IMAGES PER FILE: Yes
NUMERICAL FORMAT: Little-endian
ORIGINATOR: CompuServe, Inc.
PLATFORM: MS-DOS, Macintosh, UNIX, Amiga, others
SUPPORTING APPLICATIONS: Too numerous to list
SPECIFICATION ON CD: Yes
CODE ON CD: Yes
IMAGES ON CD: Yes
SEE ALSO: Chapter 9, *Data Compression*

USAGE: Originally designed to facilitate image transfer and online storage for use by CompuServe and its customers, GIF is primarily an exchange and storage format, although it is based on, and is supported by, many applications.

COMMENTS: A well-defined, well-documented format in wide use, which is quick, easy to read, and reasonably easy to uncompress. It lacks, however, support for the storage of deep-pixel images.

Overview

GIF (Graphics Interchange Format) is a creation of CompuServe and is used to store multiple bitmap images in a single file for exchange between platforms and systems. In terms of number of files in existence, GIF is perhaps the most widely used format for storing multibit graphics and image data. Even a quick peek into the graphics file section of most BBSs and file archives seems to prove this true. Many of these are high-quality images of people, landscapes, cars, astrophotographs, and anthropometric gynoidal data (you guess what that is). Shareware libraries and BBSs are filled with megabytes of GIF images.

428 GRAPHICS FILE FORMATS 429

456 / 1158

Encyclopedia of graphics file formats
by Murray, James D., VanRyper, William

Publication date
1996

1,638 Views

WHAT WE'D NEED....

(MORE EXACTLY, WE FIRST NEED
THE TOOLS TO GET THERE)

New
Edition

New
content

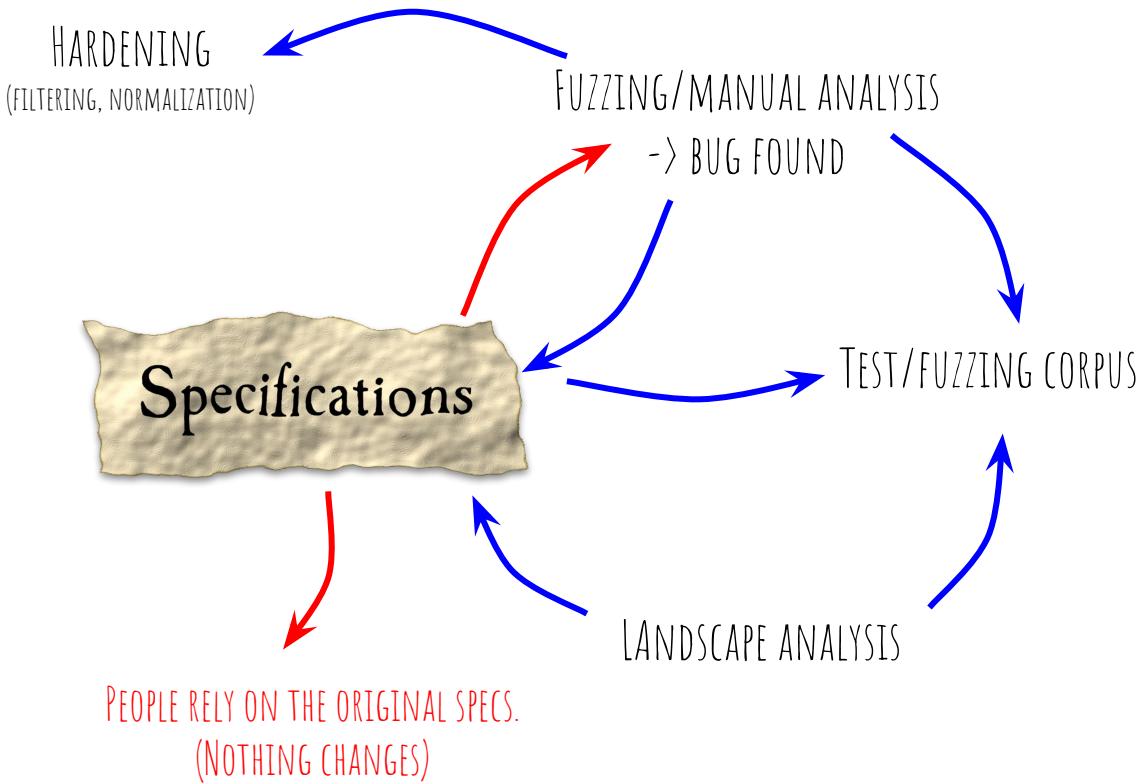
Covers
all CVEs

Test files
included

Cheat
sheets

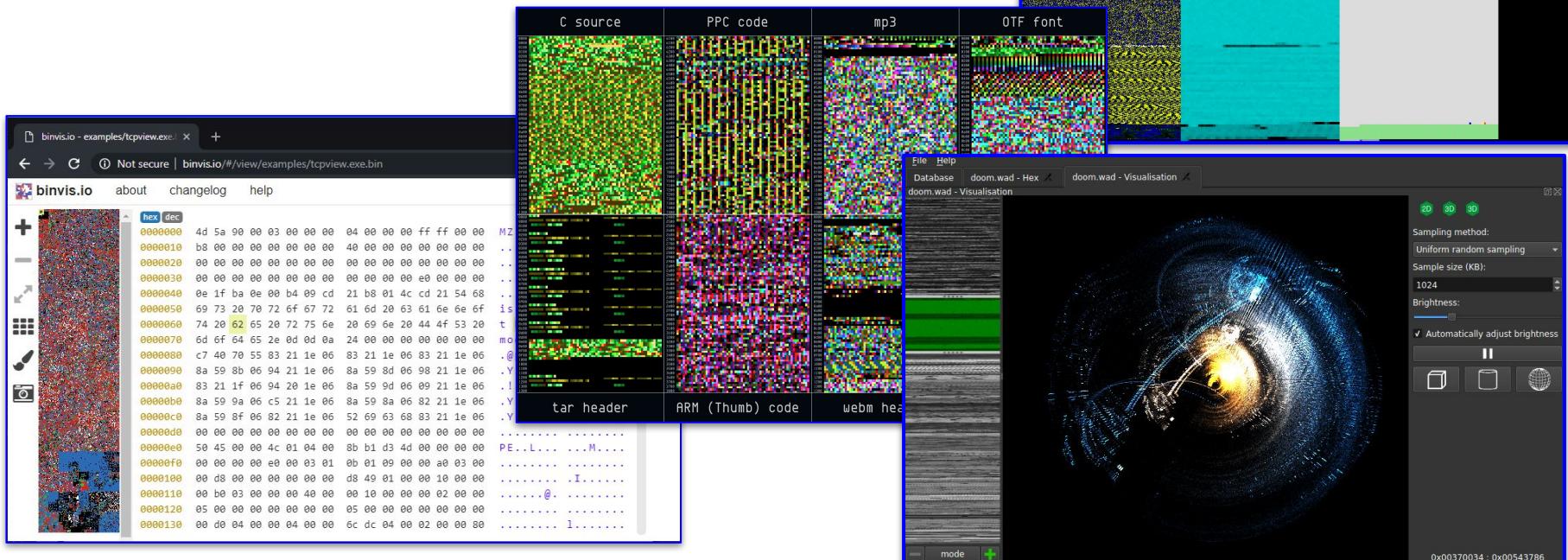
THE STATUS QUO

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, vlx_base128_be, vlx_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, rtp_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

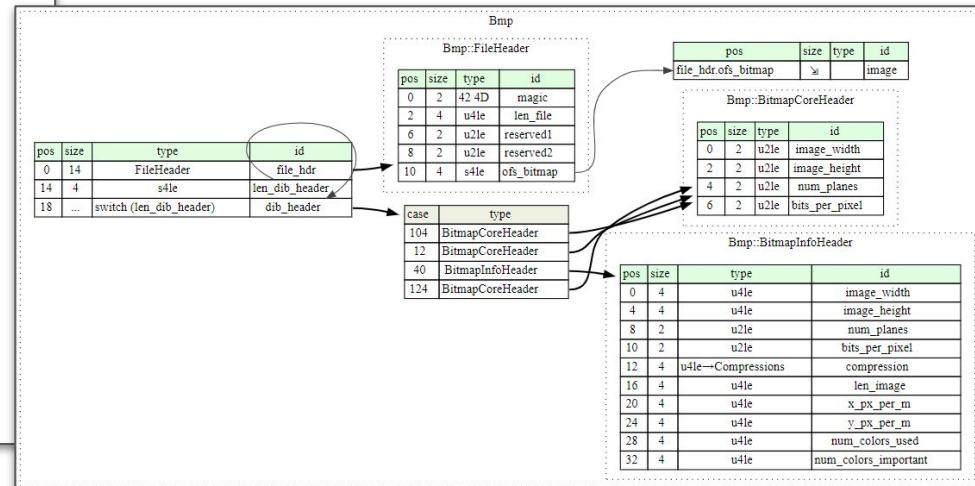
Image Files
bmp, dicom, exif, exif_be, exif_le, gif, ico, jpeg, pcf, pcx_dcx, png, psx_tif, tga, wmf, xwd

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, magicvoxel_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_marshall



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 Web IDE

<https://ide.kaitai.io/#>

files zip_modified.ksy

```

1 meta:
2   id: zip
3   file-extension: zip
4   endian: le
5   license: CC0-1.0
6 seq:
7   - id: sections
8     type: pk_section
9     repeat: eos
10 types:
11   pk_section:
12     seq:
13       - id: magic
14         contents: 'PK'
15       - id: section_type
16         type: u2
17       - id: body
18         type:
19           switch-on: section_type
20             cases:
21               0x0201: central_dir_entry
22               0x0403: local_file
23               0x0605: end_of_central_dir
24 local_file:
25   seq:
26     - id: header
27       type: local_file_header
28     - id: body
29       size: header.compressed_size
30 local_file_header:
31   seq:
32     - id: version
33       type: u2
34     - id: flags
35       type: u2
36     - id: compression_method
37       type: u2

```

object tree

```

- sections
  10 [PkSection]
    | magic = [80, 75]
    | sectionType = 0x403 = 1027
  - body [LocalFile]
    | header [LocalFileHeader]
      | version = 0x14 = 20
      | flags = 0x2 = 2
      | compressionMethod = DEFLATED (0x8 = 8)
      | fileModTime = 0x8819 = 35609
      | fileModDate = 0x4EAB = 20139
      | crc32 = 0x85892AE0 = 2240359136
      | compressedSize = 0x10 = 16
      | uncompressedSize = 0xE = 14
      | fileNameLen = 0x9 = 9
      | extraLen = 0x0 = 0
      | fileName = hello.txt
      | extra [Extras]
        | entries
          | body = [243, 72, 205, 201, 201, 87, 8, 207, ...]
  -1 [PkSection]
    | magic = [80, 75]
    | sectionType = 0x201 = 513
  - body [CentralDirEntry]
    | versionMadeBy = 0x14 = 20
    | versionNeededToExtract = 0x14 = 20
    | flags = 0x2 = 2
    | compressionMethod = DEFLATED (0x8 = 8)
    | lastModFileTime = 0x8819 = 35609
    | lastModFileDate = 0x4EAB = 20139
    | crc32 = 0x85892AE0 = 2240359136
    | compressedSize = 0x10 = 16
    | uncompressedSize = 0xE = 14
    | fileNameLen = 0x9 = 9
    | extraLen = 0x0 = 0
    | commentLen = 0x0 = 0
    | diskNumberStart = 0x0 = 0

```

JS code JS code (debug) hello.zip

hex viewer

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF		
00000000	50	4b	03	04	14	00	02	00	08	00	19	8b	ab	4e	e0	2a	PK.....<Na*	
00000010	89	85	10	00	00	00	00	00	00	00	00	00	00	68	65he		
00000020	6c	6c	6f	2e	74	78	74	f3	48	cd	c9	c9	57	08	cf	2f	llo.txt6HÍÉW.I/	
00000030	ca	49	51	e4	e5	02	00	50	4b	01	02	14	00	14	00	02	ÍIQåâ..PK.....	
00000040	00	08	00	19	8b	ab	4e	e0	2a	89	85	10	00	00	00	0e<Na*.....	
00000050	00	00	00	09	00	00	00	00	00	00	00	00	01	00	20	00	00
00000060	00	00	00	00	00	00	68	65	6c	6c	6f	2e	74	78	74	50	4bhello.txtPK
00000070	05	06	00	00	00	00	01	00	01	00	37	00	00	00	37	007..7.	
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

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() {
}

```

```

    _magic = m__io.EnsureFixedContents(new byte[] { 66, 77 });
    _lenFile = m__io.ReadU4le();
    _reserved1 = m__io.ReadU2le();
    _reserved2 = m__io.ReadU2le();
    _ofsBitmap = m__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();
}

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();
}

```

KAITAI PARSER COMPILER

```

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();
}

```

```

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();
}

```

```

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()

```

```

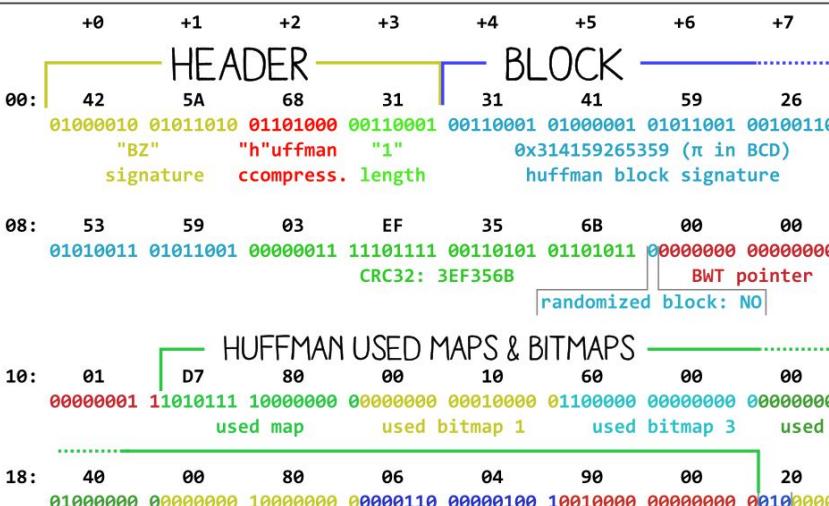
def _read
    @magic = @_io.ensure_fixed_contents([66, 77].pack('C'))
    @len_file = @_io.read_u4le
    @reserved1 = @_io.read_u2le
    @reserved2 = @_io.read_u2le
    @ofs_bitmap = @_io.read_s4le
    self
end

```

KAITAI LIMITATIONS

NOT EVERYTHING CAN BE EXPRESSED WITH YAML.

MIXED FORMATS (PDF) OR BIT-LEVEL (BZIP2) CAN'T WORK.



<= BZIP2
(BIT-BASED)

PDF =>
(TEXT SKELETON)

HEADER

SIGNATURE & VERSION INFORMATION

```
1 0 obj
<< /ID [ ] /Value [ ] >>
/Pages 2 0 R
>>
endobj
```

OBJECT REFERENCE:

DICTIONARY

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

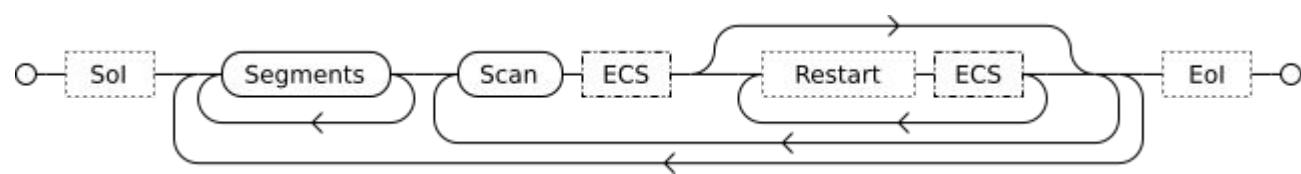
3 0 obj

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

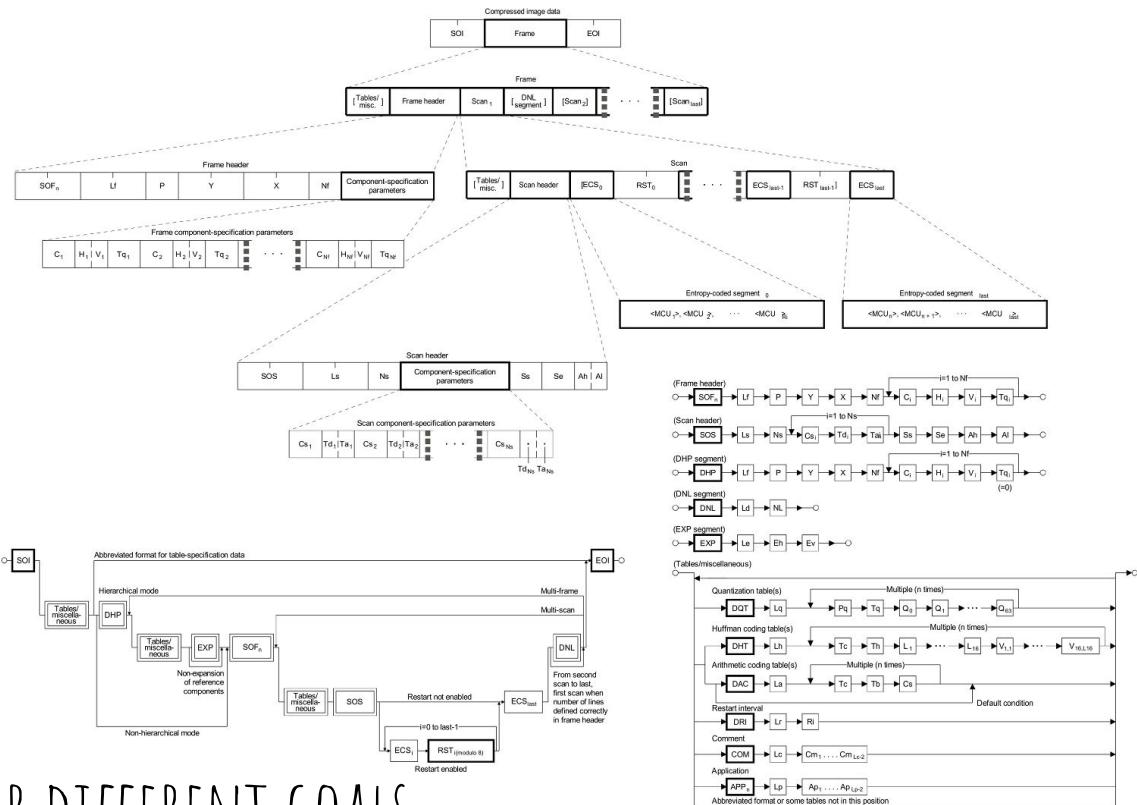
BODY

SYNTAX DIAGRAMS

VERY GOOD TO EXPLAIN LOGIC AT VARIOUS LEVEL.
UNDERRATED, UNDERUSED.



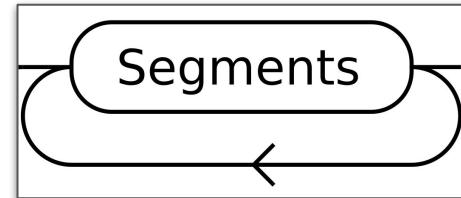
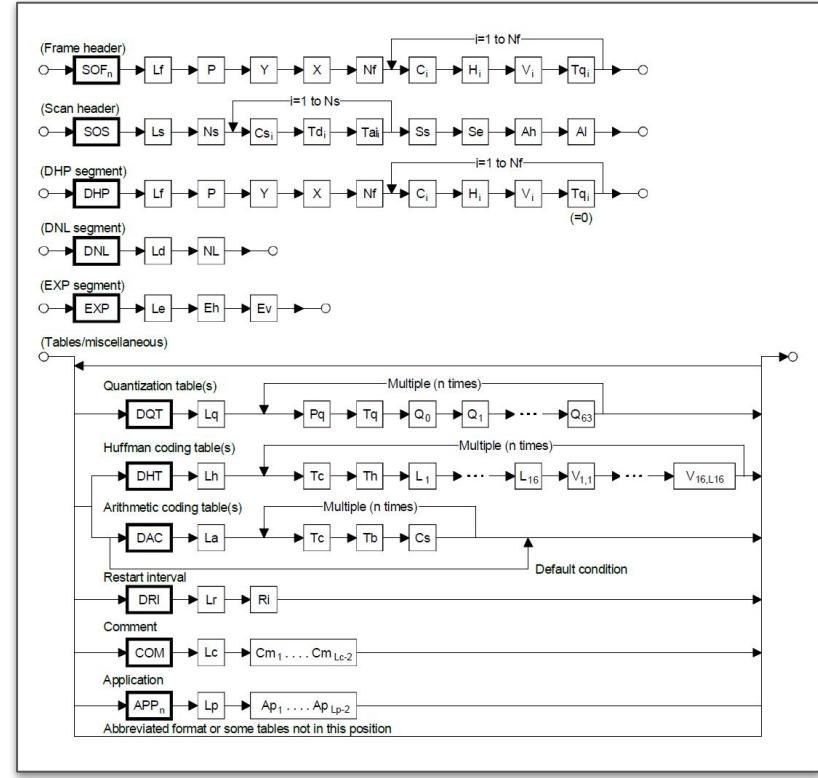
2 SYNTAX DIAGRAMS
OF THE SAME FORMAT
(JPEG)

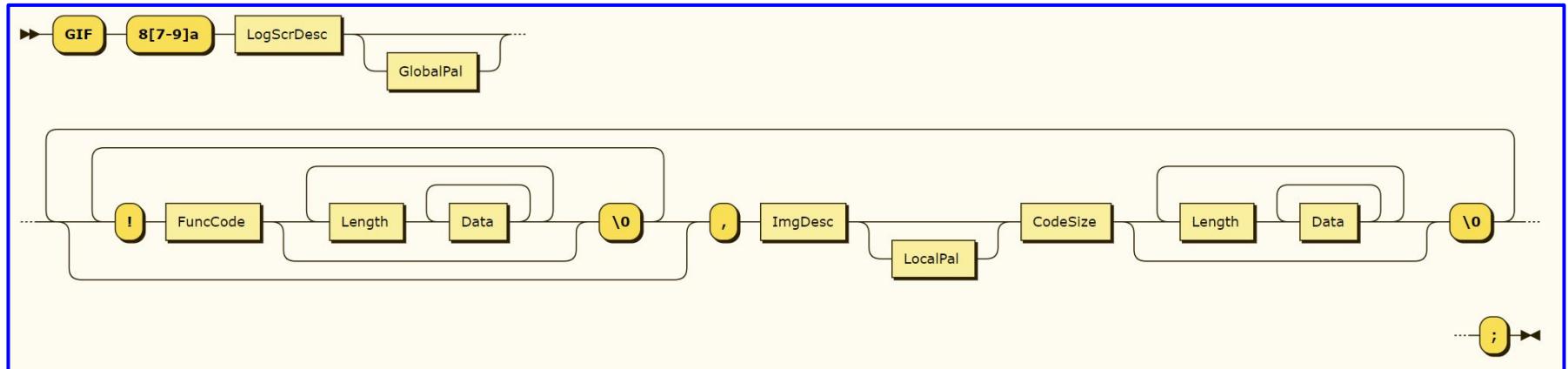


DIFFERENT LEVELS OF DETAILS FOR DIFFERENT GOALS.

IT CAN BE USEFUL TO SEE
EVERY DETAIL.

IT CAN BE OVERWHELMING
(AND INTIMIDATING)
AND PREVENT US TO GRASP
THEIR GENERIC STRUCTURE.



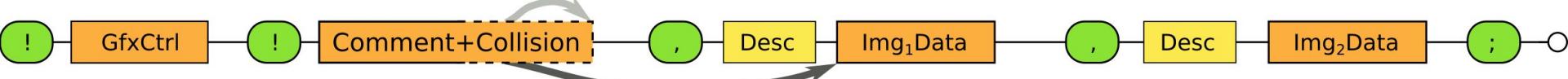


ALREADY SIMPLIFIED, YET NOT SO CLEAR
-> YOU MAY MISS SOME IMPORTANT POINTS.

USEFUL TO EXPLAIN SPECIFIC CONCEPTS.

SAME COLOR+SHAPE = SAME DATA STRUCTURE

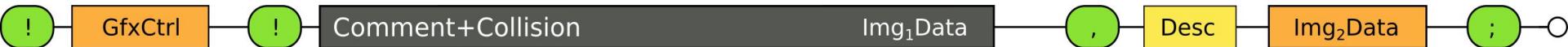
COLLISION SCHEMA



SHORT COMMENT: COMMENT STOPS BEFORE THE FIRST IMAGE.



LONG COMMENT: 1ST IMAGE EXTENDED AS A COMMENT



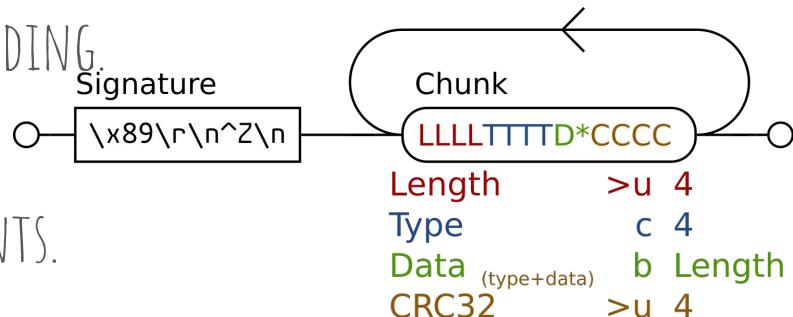
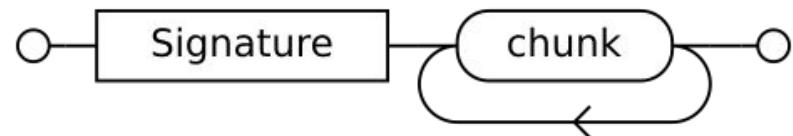
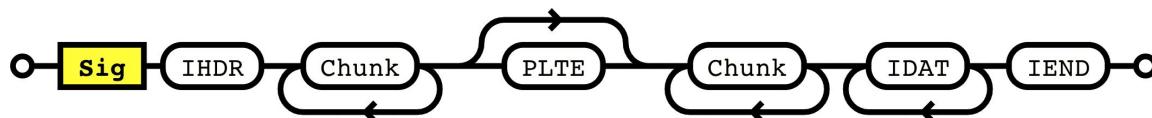
WHAT DO THEY LACK? 1/2

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.

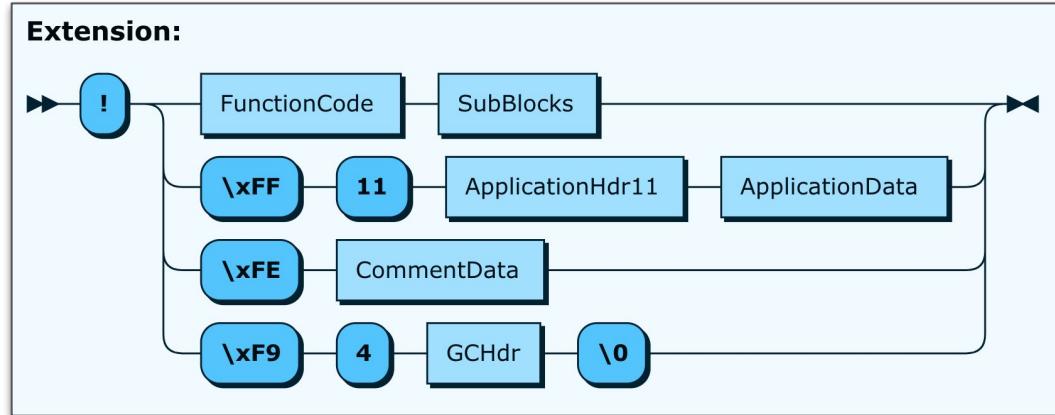


WHAT DO THEY LACK? 2/2

NO COLLAPSABLE GROUPS - THAT COULD BE ANNOTATED.

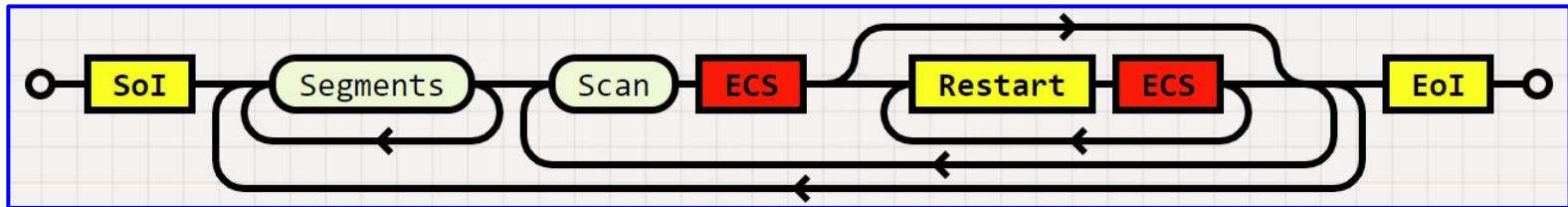
NO RELATIONS BETWEEN ELEMENTS:

Ex: ISPALETTEPRESENT BIT THEN PALETTE ARRAY.



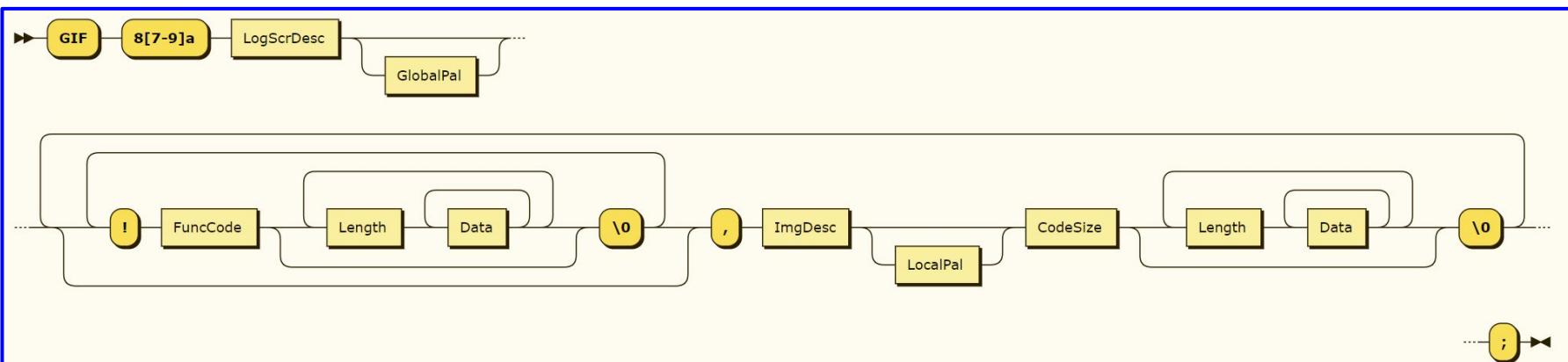
SOME (LIMITED BUT WORTH KNOWING) TOOLS FOR SYNTAX DIAGRAMS

JPEG



```
macro_rules! ECS {  
    ($$Sol:expr $($($ Segments )+ $($ Scan $$ECS:ty $($ $Restart:expr $$ECS:ty)* )+ )+ $$Eol: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
```

The screenshot shows a GitHub Gist page with a blue header bar. The main content area contains the following text:

```
1467 lines (1182 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(s)

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

The screenshot shows a binary visualizer interface with several panels. At the top, a panel titled "Type: PNG" displays the hex dump of the file signature: 000 89 .P .N .G \r \n 1a \n, with columns for hex, ASCII, and decimal values. Below it, a "Chunk: Image Header" panel shows the same data. Further down, there are sections for "TYPE.PNG" and "CHUNK: IMAGE HEADER", each with its own hex dump and field information. The bottom right corner of this section shows the file's length (13) and type (IHDR).

This screenshot shows a more detailed binary analysis tool. It focuses on the "Image Header" chunk, which is identified by its offset (000), length (13), type (IHDR), and CRC-32 value (0x948283e3). The tool also lists other chunks: "Image Data" (length 21, type IDAT, CRC-32 0xe93261e5) and "Image End" (length 0, type IEND, CRC-32 0xae426082). Each chunk entry includes its offset, length, type, and CRC-32 value.

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.

```
db `x89PNG\r\n\x1a\n` ; signature ;0000:  
  
chunk1: ; chunk1 { //Image Header ;0008:  
    ddbe 13 ; Length  
;ddbe (chunk1.crc32 - chunk1.data)  
.type db `IHDR` ; type ;000c:  
.data: ; Data {  
    incbin 'rgb.png', 0x10, 0xd ;0010:  
    ;}  
.crc32 ddbe 0x948283e3 ; crc-32 ;001d:  
;> chunk1.crc32=CRC32(chunk1.type,chunk1.crc32)  
;  
  
chunk2: ; chunk2 { //Image Data ;0021:  
    ddbe 21 ; Length  
;ddbe (chunk2.crc32 - chunk2.data)  
.type db `IDAT` ; type ;0025:  
.data: ; Data {  
    incbin 'rgb.png', 0x29, 0x15 ;0029:  
    ;}  
.crc32 ddbe 0xe93261e5 ; crc-32 ;003e:  
;> chunk2.crc32=CRC32(chunk2.type,chunk2.crc32)  
;  
  
chunk3: ; chunk3 { //Image End ;0042:  
    ddbe 0 ; Length  
;ddbe (chunk3.crc32 - chunk3.data)  
.type db `IEND` ; type ;0046:  
.data: ; Data {  
    .crc32 ddbe 0xae426082 ; crc-32 ;004a:  
    ;> chunk3.crc32=CRC32(chunk3.type,chunk3.crc32)  
    ;}  
;
```

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

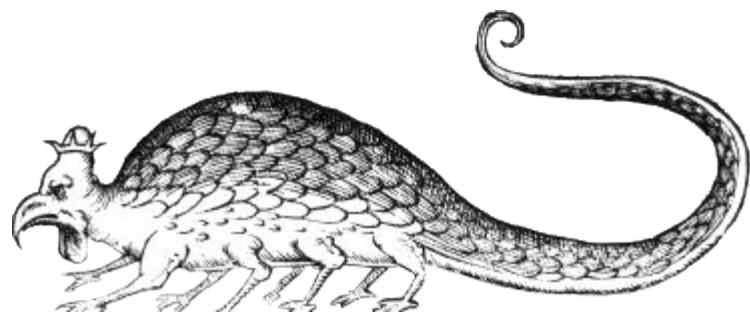
2019/03/21

🇩🇪 TROOPERS, HEIDELBERG



MORE INFO @ [HTTPS://SPEAKERDECK.COM/ANGE/NO-MORE-DUMB-HEX](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

DICTIONARY
 << [ID VALUE]* >>
 1 0 obj
 << /Pages 2 0 R
 >>
 endobj

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

IDENTIFIER (WITH /)

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

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

LENGTH, COMPRESSION....

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

BODY

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\obj<</Resources<</Font<</<</BaseFont//Subtype/>>>>/Contents<<>>stream\n/\t50Tf20\r450Td(http://www.corkami.com)Tjendstream>>endobj\x20\ntrailer<</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 wtf0C.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: 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 wtf0C.pdf repaired.pdf
WARNING: wtf0C.pdf: can't find PDF header
WARNING: wtf0C.pdf: file is damaged
WARNING: wtf0C.pdf: can't find startxref
WARNING: wtf0C.pdf: Attempting to reconstruct cross-reference table
wtf0C.pdf: unable to find trailer dictionary while recovering damaged file

$
```

OUTPUT FROM `mutool`:

```
%PDF-0.0
%%μŪ

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

The screenshot shows the ISO website's store page for ISO 32000-1:2008. At the top, the ISO logo and the text "International Organization for Standardization" and "When the world agrees" are visible. Below the header, there are navigation links for "Standards", "All about ISO", "Taking part", "Store", and a search bar. The main content area shows the product details for ISO 32000-1:2008, including its title, document management -- Portable document format -- Part 1: PDF 1.7. A note indicates that the standard was last reviewed in 2018 and remains current. The product description states that ISO 32000-1:2008 specifies a digital form for representing electronic documents to enable users to exchange and view electronic documents independent of the environment in which they were created or the environment in which they are viewed or printed. It is intended for the developer of software that creates PDF files (conforming writers), software that reads existing PDF files and interprets their contents for display and interaction (conforming readers) and PDF products that read and/or write PDF files for a variety of other purposes (conforming products). A list of what the standard does not specify follows, including specific processes for converting paper or electronic documents to the PDF format, specific technical design, user interface or implementation or operational details of rendering, specific physical methods of storing these documents such as media and storage conditions, methods for validating the conformance of PDF files or readers, and required computer hardware and/or operating system. To the right, a "Buy this standard" box offers the standard in PDF format (checked) or PDF on CD, both in English, for CHF 198, with a "Buy" button.

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?

