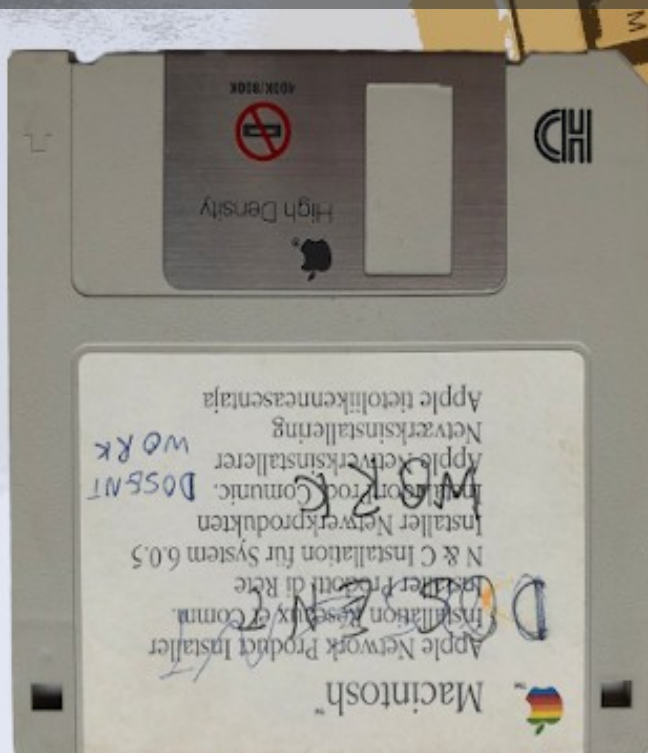


# lab6

## IN THIS ISSUE

- 2 NEW FRONTIERS IN PDF ACCESSIBILITY
- 4 PUBLISHING IN PDF: PRELIMINARY EXPERIMENTAL RESULTS
- 9 FANTASY FOOTBYTE
- 11 COLOPHON
- 12 TRACKING



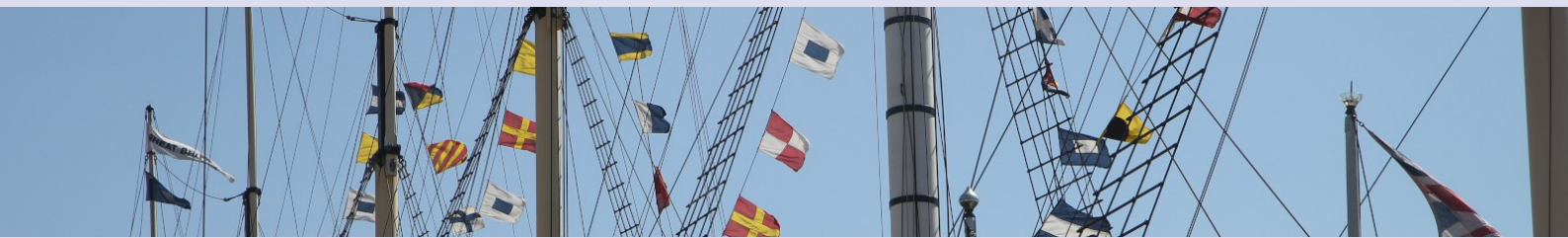
£1.99 JUNE 2021

BURN ALL CRYPTOCOINS!



ISSUE 1

Find the disk image as a PDF attachment



## New Frontiers in PDF Accessibility

This file is both a valid PDF/A-3b document and a valid MP3 file containing a dramatic reading of the content. It is also readable as plain text in any text editor. It has been tested in applications such as Adobe Reader, Windows Media Player, and Vim.

Why? Because sometimes you want to read Lab 6 with full colour and glorious layout, and sometimes you want to read Lab 6 in a text mode terminal, and sometimes you want to listen to it while swimming – and you *don't* want to maintain multiple copies in separate files that might go missing.

How? Through the magic of [binary polyglots](#)!

While many PDF readers take a flexible approach to finding the 5 PDF header bytes `%PDF-`, [the spec](#) requires that they occur at offset zero. Adobe Reader is happy to search the first 1024 bytes for the magic string, but validation tools are rightly stricter. Meanwhile, the MP3<sup>1</sup> file format has no overall header, instead consisting of a [sequence of frames](#) for which players must [search](#). I suspect that this behaviour is helpful in the context of live streams that may deliver partial frames and leading garbage such as HTTP headers; the player just needs to find the frame sync bits and pick up from there.

This tolerance means we can hide the MP3 file away within a PDF stream object. Placing non-PDF data here doesn't clash with PDF content because the PDF file format contains pointers to explicit offsets that allow readers to seek straight to the relevant objects. We still need to place this non-PDF data close to the start of the file as some media players bail out if they search for too long without locating any MP3 frames. Fortunately, there are very few mandatory PDF preamble bytes, so we don't need to keep media players waiting too long.

Since PDF uses hard-coded pointers to objects, adding additional objects near the start is not going to work without rebuilding all those pointers so they point to the new, higher, offsets. Fortunately, [qpdf](#) is adept at inserting

---

<sup>1</sup> Why MP3? Since we live so far into the future, all relevant patents have now expired, making it an unencumbered format with near-universal support.

extra text after the header, and with only a slight modification can be convinced to add arbitrary binary data.

What remains is to add the plain text representation of the document. PDFs treat any line starting with a `%` as a comment, so we could just write the text as a series of comment lines, but this is an unnecessary constraint when we can just wrap it in another stream object, to make the text visually cleaner. The flaw in this plan is that if we add too much plain text, we risk the media players losing interest. Fortunately, plain text is very light weight, so there is substantial headroom available in the file.

It's true that the plain text content will be immediately followed by a wall of binary gibberish that won't look good in a text editor – but it's trivial to just not read any further.

It is important to adhere to the PDF spec precisely, as Adobe have a history of [blacklisting](#)<sup>2</sup> polyglot techniques that bend the spec too far.

And hey presto! A universally accessible PDF/MP3/TXT document, all in one file.

ISO/IEC 11172-3: 1993 (E)

© ISO/IEC


2.4 Requirements

2.4.1 Specification of the coded audio bitstream syntax

2.4.1.1 Audio sequence

Syntax	No. of bits	Mnemonic
<pre>audio sequence() {     while (nextbits()==syncword) {         frame()     } }</pre>		

Figure 1: The MP3 spec confirming that arbitrary data may lawfully precede the sync word

 Find **assemble\_txtmp3\_payload.py** on the coverdisk

 Find **qpdfmod.patch** on the coverdisk

<sup>2</sup> Don't miss the link to the [TIFF/EXT2 polyglot](#) at the MIT Mystery Hunt 2015 site.



## Publishing in PDF: preliminary experimental results

How much of the web ecosystem supports PDF properly? Thankfully, a lot of it. Browsers are particularly good: all of them are happy to load a PDF as the root document on a domain<sup>3</sup>. Google Chrome and Firefox both support searching for phrases that span multiple lines, whereas offline rendering libraries like poppler (which underpins viewers such as Evince, Okular and Inkscape) [do not](#)<sup>4</sup>.

The Archive.org Wayback Machine [supports PDFs flawlessly](#) but [archive.today](#) does not.

PDFs are out of scope of the [W3C validator](#), but there are [alternative validators](#) available.

Google's [PageSpeed Insights](#) tool bails out, describing the page as "NOT\_HTML", which is both disappointing and accurate.

Adobe Reader iOS's [Liquid Mode](#) didn't work on Lab 6 Issue 0 at the start of the year, but does now. It paints many rendering anomalies, but overall puts in a good effort and largely delivers the continuous-scroll reflowing that Tagged PDF promises.

Because I love comedy, I tried a couple of SEO optimisation tools which are so scummy I won't even bother linking to them. You can imagine how helpful their advice was.

## Feedback

To the general concept of a PDF-based website, reactions have ranged from "love it" and "a compelling proposition", to "baffled", "don't", and "I hate reading that on mobile". The Lab 6 Corporate Voice is undeterred, and believes the experience will resonate more profoundly as the value of static publishing is rediscovered.


One bug has been discovered in Lab 6 Issue 0: a typo. Lab6 is not merely promotional material for Adobe, but is also a static, immutable doc-chain

<sup>3</sup> As long as they are set to open PDF files themselves, rather than download them.

<sup>4</sup> Bug number 56, created 13 years ago.



with each issue referring back to the published hashes of previous issues – so we can’t just edit the bug away. What we can do though, is patch it.

 Find **patch0.lab** on the coverdisk

Here’s how it works:

PDFs support incremental updates, meaning you can append new objects (and replacement versions of old objects) to the end, then write a new cross-reference table that identifies the new object and points to the previous cross-reference table – all without having to rewrite any of the original document. If you grab your trusty copy of Lab 6 Issue 0, you can apply the patch as follows:

```
$ cat patch0.lab >> 0.pdf
```

The new SHA256 hash is:

```
23960e5ad7c58b67f83ad6729e0c9be5af3822de726393da584c8ec7e12684b7
```

The patch was developed mostly manually, with some light automation; compiled PDF objects can be *very* unfriendly to read. Here’s an example of how the text “Find the disk as a PDF attachment!” is encoded by LibreOffice:

```
5.7 142.089 Td /F3 10 Tf[<0102>-2<03040506>7<0708>-6<050402>-2(\  
t)9(\n)-6<050b09050b050c0d>2<01050b06>-2<06>-2<0b0e>3<070f>2(\b)-  
6<0306>-2<10>]TJ
```

**Td**, **Tf**, and **TJ** are commands (set display position, select font, and write text, respectively), preceded by their arguments. **TJ**’s bracketed argument contains the text, encoded as the glyph number in the selected font. So **<0102>** represents the first and second glyph in the font **/F3**. Only hex numbers inside angle brackets are glyph numbers; the numbers outside the angle brackets are horizontal offsets used to [kern](#) letters. Bizarrely, when a glyph number has an ASCII representation, Libreoffice chooses to emit an ASCII literal in parentheses, such as **(\n)**. This is *not* a newline; it’s just glyph number 0x0A.

Glyph numbers are *not* Unicode code points. Nor are they the number of the glyph in an embedded font file, because PDFs typically don’t embed font files. Instead, the PDF creator software subsets fonts so that only the required glyphs are included, and in this case it numbers the glyphs in the order they are used, which is why you see a progression from 01, 02, 03, 04, 05, 06, 07, 08, and only back to 05 when we hit the first re-used glyph in the sentence, which in this case is a space. (It’s worth noting that this

example comes from the first use of this font in the document, so this really is the first time the glyph is encountered).

Having understood the syntax, we are still no closer to decoding the text. For this we must separately look up the font object `/F3` and find its `/ToUnicode` mapping, which finally gives us a way to translate glyph number into a Unicode code point. PDFs in general are not required to contain a `/ToUnicode` mapping, which means it's possible to include text that can be displayed, but not exported without the use of [OCR](#). But that's why we use PDF/A, where such mappings are mandatory.

To cut a long story short, the offending paragraph in Issue 0 was located with the aid of some brute force Python:

 Find **decodetj.py** on the coverdisk

From this point it was simply a matter of extracting the original paragraph, manually patching it with some extra characters, fixing up the length, then adding a new trailer with manually-recalculated offsets.

So, slightly harder than hitting the edit button on a Wordpress blog entry, but you'll surely agree thoroughly worth it.

## Further development

Now that PDF has proper audio support, what about video? Unfortunately, the most popular video container formats like Matroska, ASF and Ogg tend to have sane requirements like headers that must be located at the beginning of the file. MP4 (ISO base media format) *technically* allows its `ftyp` box to appear merely "as early as possible" in the file, but in practice media players like mpv and VLC are too picky to accept this.

Not to worry – video doesn't bring a lot to the accessibility table.

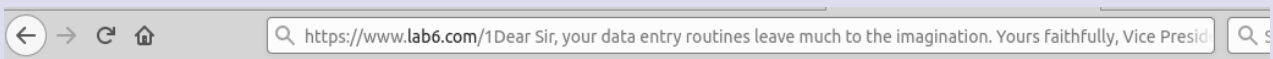
## Commentary

Feedback and comment forms are nice features on a website, but this isn't really an option in PDF/A. In regular cursed PDF<sup>5</sup>, you can create editable text fields and a submit button which POSTs to a URL, but we're not using regular cursed PDF here, so we're out of luck, right?

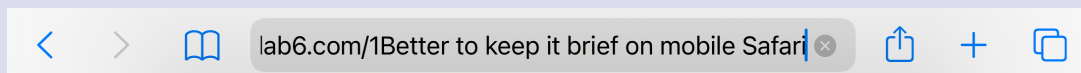
Wrong!

Browsers still, as of 2021, display a URL bar to users, and the URL bar is none other than a user-editable text field with a submit action!

So, to submit comments on this document, just enter your comment in the address bar after the URL, and it shall be grepped out of the web server logs. Here's an example of a comment on Issue 1 using Firefox:



And another on iOS:



To be clear:

`https://www.lab6.com/1<Your Comment Goes Here! Type anything, then press enter>`

The comment can only be a structureless character stream, so feel free to describe your own human-readable metadata in-band. The maximum length is a few thousand characters. Your browser should take care of encoding.

In the event that you are not using a browser to read this document, you can still use a browser to submit a comment – or indeed any device capable of issuing an HTTP request.

Just be sure to append your comment to the URL of the document, including the number, so that it is clear which issue you are commenting on.

Comments will then go into an eldritch limbo (“moderation queue”), existing only in the server logs, until a future Lab 6 editor remembers to run a command like the following:

```
sed -n -r 's/.*"GET \/[0-9]+(.+) HTTP\./\1/gp' access.log | uniq | python3 -c
"import sys, urllib.parse as ul; print(*[ul.unquote_plus(l) for l in sys.stdin],
end='\n')"
```

## The small web

Of course, PDFs tend to have a life independent of the web, and so Lab 6 is now also available over [Gemini](#):

<gemini://lab6.com>

(Indeed, this issue was published *first* over Gemini).

The Gemini protocol favours text documents written in its own lightweight markup format, but does not prevent hosting other file types.

Reading Gemini sites using Petr Vernigorov's Elaho on iOS is a delightful experience – and by virtue of being based on Firefox, it even handles opening PDF files natively!

Comment-submission-via-URL works over Gemini too, naturally.

## PDF in the News

The [PDF/A-4 standard](#) was approved in November 2020, and like most new technology, it appears to be a regression, due to allowing the presence of JavaScript in the file, although it is hard to verify this as the standard is kept behind the [ISO paywall](#). In order to keep your PDF/A lawn pristine, best to keep PDF/A-4 kids off it until this can be clarified.





## Fantasy Footbytes

Anyway, moving on from PDFs...

Just as [colornames.org](http://colornames.org) is doing sterling work filling out the namespace of 24-bit colours, Lab 6 is proud to do its bit for the English language by naming all the 8-bit bytes. There are 256 of these beauties, and while some of them have been named in character encodings, the more popular ones such as ASCII, UTF-8, and Windows Code Page 1252, leave several values undefined. Mac Roman does manage to assign a character, symbol, or control code name to every value, including the Apple logo *twice*. But these aren't *names*. Not really. Let's change that.

Grab your magnifying glass and check out the table on the next page, or read the file in text mode to find the data in a more convenient format.

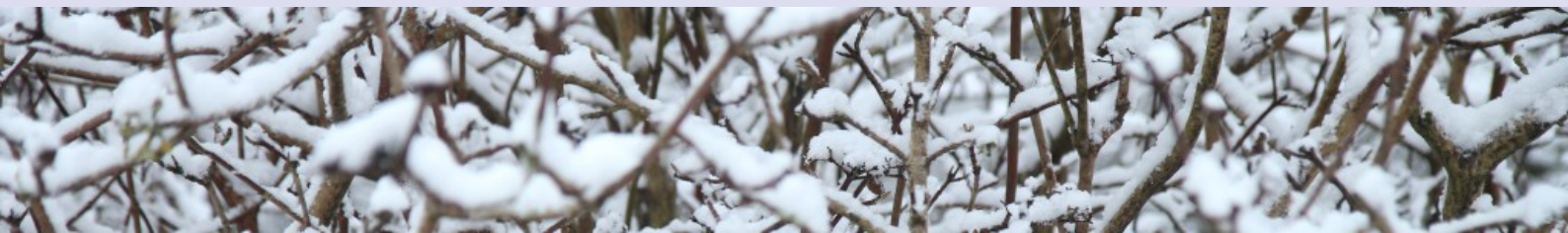
The scene is now set to destroy environmentally ruinous proof of work cryptocurrencies by neutralising them into a mechanism for running a fantasy football tournament:

- Pick a nice round Bitcoin block like  $S = 393216_{\text{DEC}} = 60000_{\text{HEX}}$ .
- Choose two teams  $T_1$  and  $T_2$  to compete against each other.
- Find the result of the match between team  $T_1$  and team  $T_2$  in block  $S + (256 \times T_1) + T_2$
- For example, in season  $60000_{\text{HEX}}$ , when Chownley play Openmillwall, the result can be found in block:

$$60000_{\text{HEX}} + (FF_{\text{HEX}} \times 8F_{\text{HEX}}) + 05_{\text{HEX}} = 68F05_{\text{HEX}}$$

- And what is the result? Whichever team has the highest [nonce](#) value wins. 65,536 blocks are needed to determine results for all local (home) matches, and another 65,536 blocks for remote (away) matches. Fixtures where a team plays themselves are always fogged off.

Dec	Hex	Team	Dec	Hex	Team	Dec	Hex	Team	Dec	Hex	Team
	0 0x00	Null City		64 0x40	AFK Bourneshell		128 0x80	Von Neucastle United		192 0xC0	TCP Porto
	1 0x01	Starthampton		65 0x41	JVham HotSpot		129 0x81	Examouth		193 0xC1	MAC Address Tel Aviv FC
	2 0x02	Writing		66 0x42	Done DD		130 0x82	Enverton		194 0xC2	Bayer Filter Munich
	3 0x03	Alvechurch-Turing		67 0x43	BSDtar Donetsk		131 0x83	Ballinabellard		195 0xC3	Olympique Lyon Estates
	4 0x04	Shellcode 04		68 0x44	Polyvillareal		132 0x84	Homotopy Cliftonville		196 0xC4	Inte8le
	5 0x05	Openmillwall		69 0x45	Nice		133 0x85	RAID Rovers		197 0xC5	Javantus
	6 0x06	Ackrington Stanley		70 0x46	OlympiarchOS		134 0x86	Less tr City		198 0xC6	Keygenoa
	7 0x07	St Synchronous Mirren		71 0x47	Static Kiev		135 0x87	Terminal Citty		199 0xC7	Botafo.go
	8 0x08	Cray Supervalley Paperless Mills		72 0x48	Kruskal Palace		136 0x88	Cursor Wanderers		200 0xC8	Sunny Cove Rangers
	9 0x09	GSV Einhorn Is Finkle		73 0x49	AST DOM Villa		137 0x89	Arstechnical		201 0xC9	Apache Fluminense
	10 0x0A	Zstandard Liège		74 0x4A	Bootsplash Arygle		138 0x8A	Altsport RFC		202 0xCA	Salisbury
	11 0x0B	Porting		75 0x4B	Airdrie Octonions		139 0x8B	Newton Raphson Aycliffe		203 0xCB	Foobarcelona
	12 0x0C	Middlesout		76 0x4C	Hallambda		140 0x8C	Run Korn Town		204 0xCC	Blyth Spartstations
	13 0x0D	Ryhope Corollary Warfare FC		77 0x4D	Greenthread Morton		141 0x8D	Guitarheroe		205 0xCD	Tilbury Containers
	14 0x0E	Beşiktaş Cult of Jimnastik Kubüntü		78 0x4E	N Queens Problem Rangers		142 0x8E	Dunwich Hamlet HP		206 0xCE	CLide
	15 0x0F	Grays Almanathletic		79 0x4F	Forth Athletic		143 0x8F	Chownley		207 0xCF	Inverurie Low Code Works
	16 0x10	Roswell First Corinthians 8:2		80 0x50	Tucowdenbeath		144 0x90	Nandwich Town		208 0xD0	Red Team
	17 0x11	Device County		81 0x51	CA OS/Asuna		145 0x91	Bill Gates' Head		209 0xD1	Blue Team
	18 0x12	Blinken 04 Lichten		82 0x52	Eternal Wednesday		146 0x92	Solderhell Moors		210 0xD2	Beaconsell Town
	19 0x13	FC SAN		83 0x53	Stirling Engine Albion		147 0x93	Tokenring Athletic		211 0xD3	DC Milan
	20 0x14	Cache Invalidation & Nîmes		84 0x54	Unsigned Long Eaton		148 0x94	Wizard's Cleeve		212 0xD4	Vasco Certified Net Associates
	21 0x15	NAK Dons		85 0x55	Midi Tower Hamlets		149 0x95	Whytehat		213 0xD5	GNUport County
	22 0x16	IaaX		86 0x56	V4L Wolfsburg		150 0x96	Malware & Xtree		214 0xD6	Nandwich Town
	23 0x17	Transmere End		87 0x57	Pentesttrith		151 0x97	Wardrivington Town		215 0xD7	Host Bromwich Adapter
	24 0x18	Bradfnord City		88 0x58	Keith Packard		152 0x98	Fractal Collieries		216 0xD8	Cryptal Palace
	25 0x19	Medium North End		89 0x59	Conwy's Game of Football		153 0x99	Kirby Marlinspike		217 0xD9	Blockpool NFT
	26 0x1A	Fenermicrobahçe		90 0x5A	LazyI/O		154 0x9A	Stallmanchester FS		218 0xDA	Architectural Lens
	27 0x1B	SSD Napoli		91 0x5B	Noetherwell		155 0x9B	Eibarbazquux		219 0xDB	Montpeltier
	28 0x1C	Sunderfile		92 0x5C	Eigencity		156 0x9C	Sunderland RSA		220 0xDC	Goole Fonts
	29 0x1D	Ignite & Hive Albion		93 0x5D	HELO Athletic		157 0x9D	TADScaster Albion		221 0xDD	Splunkirk
	30 0x1E	Intercal Milan		94 0x5E	Gala Fairydean Naming Things Is Hard		158 0x9E	Rovers Return Oriented Football		222 0xDE	Marlowverflow
	31 0x1F	Atalanparty		95 0x5F	Miningrigg Rose Athletic		159 0x9F	Object Orient		223 0xDF	Bashley
	32 0x20	Lunar Rovers		96 0x60	Losslessmouth		160 0xA0	Angsty Gnomads		224 0xE0	Hashford Town
	33 0x21	Winampdoria		97 0x61	DAG Algorithm and Redshift		161 0xA1	Chiptune Modbury Town		225 0xE1	FK Riemann Zeta
	34 0x22	Port Knock		98 0x62	East Fifo		162 0xA2	Disjoint United		226 0xE2	Genthub
	35 0x23	Oct*thorpe United		99 0x63	SparTeX MOSFET		163 0xA3	Rampishamware		227 0xE3	3D Chessterfield
	36 0x24	Hartleblackliver Mining Pool		100 0x64	Stran.rar		164 0xA4	Dunstable Multivibrator		228 0xE4	Xrangers
	37 0x25	Cagliarewriting		101 0x65	Dun k-line		165 0xA5	FLACwell Heath		229 0xE5	Ports Tree Vale
	38 0x26	Bologna/Lux		102 0x66	Irlambda		166 0xA6	NetAFC Hayes		230 0xE6	Sphere Everton
	39 0x27	SC Librebουργ		103 0x67	Cardijk City		167 0xA7	Trafford Shaping		231 0xE7	Ballardrat
	40 0x28	Cá.DIZ		104 0x68	Stack City		168 0xA8	Stenhouse more		232 0xE8	Seymour Cray Wanderers
	41 0x29	Apt-Getafe		105 0x69	Covertree City		169 0xA9	HANA Athletic		233 0xE9	Enterprise Civil Service Bus
	42 0x2A	Typecaster Rovers		106 0x6A	Acidburn Rovers		170 0xAA	Olapton		234 0xEA	Darwen XNU
	43 0x2B	Turbo Chelc++		107 0x6B	Napcester City		171 0xAB	Hashton & Backup United		235 0xEB	IMPLY Gate Priory
	44 0x2C	Zener St. Petersburg		108 0x6C	Wolfframalphton Wanderers		172 0xAC	Tunbridge Seekwells		236 0xEC	Abingdom Town
	45 0x2D	AFC St. Austelnet		109 0x6D	Andds County		173 0xAD	Coleshill Climbing Town		237 0xED	Halespwn Town
	46 0x2E	Hibernation		110 0x6E	Torqwrench United		174 0xAE	Glasgow Haskell Aloysians		238 0xEE	KirkMcKusickloch Rob Pike
	47 0x2F	Prescot Ribbon Cables		111 0x6F	Rexsham		175 0xAF	Linterhouse		239 0xEF	Portsknuth
	48 0x30	Sheffield Oday		112 0x70	Raspbury		176 0xB0	Valgrind of Clyde		240 0xF0	Great YARNmouth Town
	49 0x31	Heather St. JS		113 0x71	Future Crew Alexandra		177 0xB1	CRC7		241 0xF1	Lamport County
	50 0x32	BO2kRussia Dortmund		114 0x72	Vimbledon		178 0xB2	MOTD		242 0xF2	Backus-Stourport
	51 0x33	LaTeX Orient		115 0x73	MFC Linking City		179 0xB3	UART Ain't a Recursive Teamname		243 0xF3	Torvaldershot Town
	52 0x34	St Johnston Sans		116 0x74	Emacslesfield Town		180 0xB4	Queens Park Named Range		244 0xF4	Dense Boldface
	53 0x35	Lothlorien Thistle Hutchison 5G		117 0x75	Debian & Redmond		181 0xB5	Celtick-tock Refresh		245 0xF5	Dijon Postel
	54 0x36	IPv6switch Town		118 0x76	Shepton Malware		182 0xB6	PR Flamengo		246 0xF6	Bjarne Strasbourg
	55 0x37	Serialbus		119 0x77	Blackburn Roverflow		183 0xB7	Breakin City		247 0xF7	Bordeauxgecoin
	56 0x38	Full HAM8		120 0x78	Inverse Hamiltonian Thistle		184 0xB8	Wysiwigan		248 0xF8	Paris Saint Voidmain()
	57 0x39	Sigkillmarnock		121 0x79	Man(1)field Town		185 0xB9	Ipswitchboard Wanderers		249 0xF9	Touhouse
	58 0x3A	AJAX		122 0x7A	Mempool Town		186 0xBA	Maidentail United		250 0xFA	Aminets
	59 0x3B	Arpenden Town		123 0x7B	MK Nod		187 0xBB	Coaxfosters		251 0xFB	Eeproma
	60 0x3C	Linker City		124 0x7C	Xargsyle		188 0xBC	Merthyriston Town		252 0xFC	Saasuolo
	61 0x3D	Scihub Academicals		125 0x7D	Md5sumderland		189 0xBD	Market Driven Town		253 0xFD	Crontone
	62 0x3E	Balanced Forest		126 0x7E	Birminghamming City		190 0xBE	Carcdrilisle United		254 0xFE	CSV 1860 Munich
	63 0x3F	Heart of Central Tendency		127 0x7F	Galoisfield Town		191 0xBF	Complex Madrid		255 0xFF	Pagnell ≠ Newport Pagnell



## Colophon

This document was drafted in OneNote, productionised using LibreOffice Writer, post-processed in Audacity and hand-tuned using a derivation of qpdf, before being glued together with bash, sed, and python. The coverdisk is a 1.44MB FAT-12 formatted floppy disk image.

The document's primary URL is <https://lab6.com/1> – although [mirrors](#) are welcome.

The document may contain outrageous falsehoods and embarrassing mistakes, unretractable due to it being immutable. Corrections may be published in future issues.

All content is licensed under:



<https://creativecommons.org/licenses/by-sa/4.0/>

Contributions may be sent to the email address on the last page, or the Bitcoin address on the first page.

This document's hexadecimal SHA256 hash begins 0x01, identifying the issue number.

250e3f7d581acff115537ba38e89ad31 is a handy random 128-bit integer that will appear in every issue of Lab 6 and can be used to search for copies.

## Back Issues

[00](#) – All hail PDF, FORENSIC.ZIP

SHA256 00c411fee9419cd861d9850dc56d53b7e6a211e90df9a1ca953e021b0cf31a56

OFFICIAL SPONSOR OF [6](#)

## Tracking

<a href="#">OEIS entries</a>	344,647
<a href="#">English Wikipedia entries</a>	6,311,391
<a href="#">RC5-72 keys tested</a> (% complete)	365,427,233,059,946,627,072 (7.738%)
<a href="#">Bitcoin blocks</a> (hash)	686413 000000000000000000000005bbef80d32208e28d73e4adf775898554d3487b72a3bf
<a href="#">XKCDs</a>	2472
<a href="#">IPv6 adoption</a>	35.06%
<a href="#">Abe Vigoda Status Page Status</a>	Still up
<a href="#">Largest Prime</a>	$2^{82,589,933} - 1$
<a href="#">BTTF Movies</a>	3
<a href="#">World population according to the CIA</a>	7,772,850,805
<a href="#">Latest stable Linux kernel</a>	5.12.9
<a href="#">Color Names</a>	2,545,185
<a href="#">Latest number on tildeverse's #counting</a>	6,070
<a href="#">Count of University of Queensland pitch drops</a>	9
<a href="#">Project Gutenberg eBooks</a>	65,490
<a href="#">UNIX time rebaselined to the big bang</a>	434313793622924534
<a href="#">OCRemix Releases</a>	4,034

Did you enjoy these numbers? Send your feedback to [comment@input.lab6.com](mailto:comment@input.lab6.com) and it may be published in the next issue.

You may have just lost Schrödinger's game.