Fonts For Forensics

by Fred Cohen, Ph.D.

Abstract:

Like other latent evidence that cannot be directly perceived by people, bit sequences have to be presented through tools. Presentations of digital forensic evidence often involve the presentation of text versions of bit sequences representing traces of events that took place within digital systems. This paper is about creating fonts for the examination and presentation of particular classes of bit sequences presented in particular ways in legal situations. Unlike fonts used for other purposes, fonts for forensics are less about the beauty of the presentation and more about the tradeoff between readability and being definitive about what is present. In other words, what you see is what you get, rather than what you see is what looks nice.

Background

The presentation of trace evidence for legal purposes has substantially different requirements than for other purposes for several reasons. This includes, without limit, (1) legal mandates may restrict page formats (e.g., require the use of pleading paper for certain submissions), (2) subtle differences in presentation may be important for bringing clarity to the information presented (e.g., the difference between several spaces and a tab character may be vital to the issues at hand), (3) challenges may be brought based on what is unclear (e.g., how can we tell from what is on this page that what you are claiming about this text is in fact true?), and (4) what is visible in many fonts may not properly reveal what is in fact present in the digital forensic data, leading to errors and omissions.

The history of fonts in non-legal uses has evolved over the last 60 years. In the early days of computing, there were two main technologies for presentation of digital data in readable form; dots and lines. The line technology used continuous media, such as deflections of a cathode ray in a cathode ray tube (CRT) or movement and up/down motion of a mechanical pen in two dimensions (the pen plotter). The dot technology consisted largely of light emitting diodes, lamps of various sorts, displays with fixed shape elements that were on or off at any given moment, and eventually the cathode ray tube with fixed scan patterns.

The fonts for plotters and line drawing CRTs originally consisted of sequences of line segments drawn one after another with pen up and down movements to break line continuity. Font designers created an array of different fonts and used coding schemes for representation, such as the American Standard Code for Information Interchange (ASCII)[1], and Extended Binary Coded Decimal Interchange Code (EBCDIC)[2]. The dot matrix font designers started using fixed height and width font elements with dots on or off within the matrix of a single symbol location. Each ASCII, EBCDIC, or other coded symbol was assigned a font element for display purposes, with the exception of some special characters, such as <backspace>, <space>, <tab>, and <carriage-return> used for location control within a line, and characters such as <line-feed> and <form-feed> for movement within the page and movement from page to page. Other byte vales were often unused.

Over time, the dot technology largely won out, with pen plotters remaining still today. As display and printer technology improved, fonts became far more complex, involving more than on/off values for each location, variable hight and width, and a wide array of different symbol sets placed within font families. Boldface, underlines, and similar things were added to reflect the printer methods of using carriage return or backspace and printing over the same location again and again to produce similar effects, fonts were developed for multiple languages, and ultimately unicode, a 2-byte coding scheme came about to help handle the explosion in the number of symbols desired within a font. While there are many other codings for bits in widespread use, the present discussion will be largely restricted to 8-bit fonts representing the ASCII character set. This is for convenience of space, but most of the results presented apply equally to other coding schemes and can be extended, with minimal difficulty, to larger and smaller symbol sets and other similar schema.

There have been a number of different software packages over the years that have provided different representations of what normally appears on the screen. For example, and without limit; the "vi" visual and "emacs" editors display control characters (e.g., control-A) as two or more characters next to each other; Wordstar, and later Microsoft Word and many other document editors have had presentation modes that show many non-printing characters; the program "hexdump" and other similar programs provide presentations of hex, octal, binary or other coding details, in some cases along side of a display of the printable representations of many of the byte values; and many display systems such as packet analyzers show and allow the expansion and contraction of records, fields within records, and various representations of content in different windows. However, to date, we have found none of these that meet the requirements of a font for forensics, in that they all fail in one way or the other to precisely and accurately present what is present for all byte values and with the basis details together in the same symbol with the presentation. Many of these packages also tend to have severe limits on the sequences they can sensibly present (e.g., the protocol analyzers make assumptions about interpretation that make them better presentations for what they are intended to project, but not for what they are not intended to project, and document formatters tend to not deal well with arbitrary binary files), don't allow flexibility in the alignment of content, and make underlying assumptions about the coding scheme that lead to interpretation difficulties by the examiner seeking to understand what is present.

Requirements of a font for forensics

As a general rule, it is highly desirable that displayed symbols from a defined symbol set used for legal purposes be precise, accurate, and unique. Precision and accuracy of representation are well understood in the legal community and, for the presentation of scientific and technical evidence, have been highly supported by legal rulings. The uniqueness property is highly desirable to avoid confusion and allow definitive answers to be given to specific questions that may arise. As a first attempt to characterize a set of rules and basis for those rules when devising fonts for use in forensic examination and presentation, the following criteria are identified and the rational explained:

- Each symbol should be clearly different from all other symbols
    - This allows for clarity about which one is identified. If this is not true, then there will be confusion both for the examiner and for those who review the results, including the lawyers, judges, juries, clerks, and public. Legal documents are often printed, scanned, reprinted, and go through other similar machinations. While it is impossible to always preserve all of the characteristics of what was originally present, it is important to provide enough of a difference between symbols so that these differences are likely to survive multi-generation copying, scanning, and a wide range of displays.
- Each symbol should be of the same width and height
    - This allows symbols to be compared to other symbols around them for location. While this may destroy the appearance of tab characters and other similar presentation values, it provides clarity around issues like spaces, columns, helps with fixed width fields, such as databases, and allows the column and row to be clearly seen and specified verbally, which is vital for providing accurate testimony in legal matters.

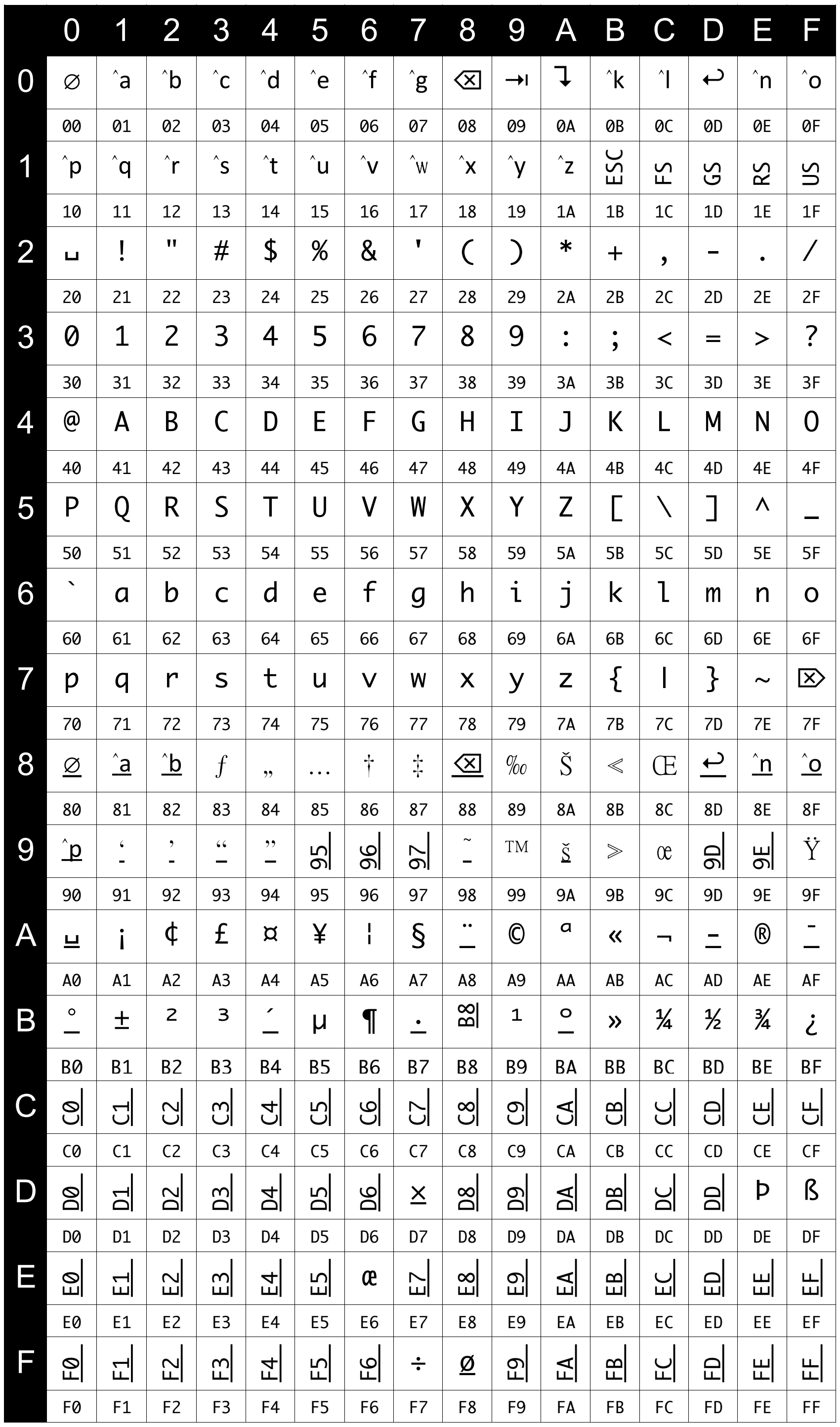
- Each symbol should be familiar, with minimal added interpretation, so that it looks similar to what might appear on a display of the same symbol on a screen or printer.
  - The word "help" should still be readable as such by someone who could read it in the normal display mode, or the font will create more confusion that it removes. Thus the font for EBCDIC will have to reflect EBCDIC coding, the one for ASCII will have to reflect ASCII coding, etc. There are limits to this today since existing fonts do not do this very well. For example, there are a wide range of different representations for the upper 128 symbols in the byte-values of the ASCII character set because, when defined, it was a 7-bit coding. The variations include a range of different accented characters, symbols used to make boxes and other graphical components, mathematical symbols, etc.. This proliferation of fonts and enormous variety of presentations is ultimately problematic, and can only be solved to a limited extent by the production of any particular font for forensics. For that reason, a forensic version of each font may ultimately be developed to allow the other properties to be met while producing a closer approximation to this one.
- Each symbol must be printable so that a <space>, <tab>, <carriage-return>, <backspace>, <escape>, and other "non-printable" characters can be clearly seen on the printed page.
  - This is necessary because, in many cases the issue in dispute is the non-printable characters, and even when they are not in dispute, it makes interpretation far easier when the non-printing characters are clearly revealed rather than being hidden. While many fonts provide presentation forms for most of the first 128 symbols within the symbol set, most of the ones observed in this study have shown a large portion of the codes from 128-255 as non-printing, essentially unassigned. There are many different options for the presentation of non-printing characters, and the presentation is a function of the utility for the examiner.
- Each symbol should self-indicate the underlying bit pattern that produced it so that it can be traced back to its original value.
  - This provides a trace back to the origin of the data that produced the symbol, and allows the individual examining it to definitively know the basis for the display provided. For example, the byte code 03 may mean different things in different contexts. By providing the code with the display, the interpretation for a different context can be made, while without it, there would be no direct way for the examiner to know the basis for the interpretation applied. This also serves to remind the viewer that these are traces of latent evidence.
- Another challenge is that the upper half (the first bit 1) of the ASCII and many other character sets, have a wide range of different presentations. The font designer must identify a way to make these meaningful across a broad context of uses.
  - For ease of understanding, except in cases where there is a clearly differentiable symbol for the upper half of the code space, the decision was taken to represent these symbols with emphasis marks (underlines) or coding indicators (e.g., FE). This may sacrifice a great deal in meaning because it does not display what actually may have "appeared" on a display or printer, but this can be resolved by the use of images of displays for cases where this is probative.

A side effect of these criteria is that the font will take up more space on a page than the normal font would take up for the same level of readability, and it will have some differences from the fonts commonly used for other purposes, such as a more distinct difference between 0 and O, 1 and l, I and |, and so forth.

A sample font for forensics

As a starting point, a forensic font for ASCII was developed, and a tool was implemented to convert any input byte sequence into a display using this font. A table depicting this font is included in Figure 1.

Figure 1 covers ASCII codes ranging from hex 00 through hex FF, with each code corresponding to a printable fixed width and fixed height depiction. The depiction includes both the area with the commonly used symbol (e.g., A, B, C, etc.), the horizontal line, and the HEX value for that symbol that is just below the common printing. It does not include the vertical lines between symbols or the horizontal separator lines. Thus the representation for HEX code 5A is a "Z", with a horizontal line under it, and the small digits 5A under that line.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	∅ 00	^a 01	^b 02	^c 03	^d 04	^e 05	^f 06	^g 07	⌫ 08	⇥ 09	↴ 0A	^k 0B	^l 0C	↩ 0D	^n 0E	^o 0F
1	^p 10	^q 11	^r 12	^s 13	^t 14	^u 15	^v 16	^w 17	^x 18	^y 19	^z 1A	ESC 1B	FS 1C	GS 1D	RS 1E	US 1F
2	␣ 20	! 21	" 22	# 23	$ 24	% 25	& 26	' 27	(28	) 29	* 2A	+ 2B	, 2C	- 2D	. 2E	/ 2F
3	0 30	1 31	2 32	3 33	4 34	5 35	6 36	7 37	8 38	9 39	: 3A	; 3B	< 3C	= 3D	> 3E	? 3F
4	@ 40	A 41	B 42	C 43	D 44	E 45	F 46	G 47	H 48	I 49	J 4A	K 4B	L 4C	M 4D	N 4E	O 4F
5	P 50	Q 51	R 52	S 53	T 54	U 55	V 56	W 57	X 58	Y 59	Z 5A	[5B	\ 5C	] 5D	^ 5E	_ 5F
6	` 60	a 61	b 62	c 63	d 64	e 65	f 66	g 67	h 68	i 69	j 6A	k 6B	l 6C	m 6D	n 6E	o 6F
7	p 70	q 71	r 72	s 73	t 74	u 75	v 76	w 77	x 78	y 79	z 7A	{ 7B	| 7C	} 7D	~ 7E	⌦ 7F
8	∅ 80	^a 81	^b 82	ƒ 83	„ 84	… 85	† 86	‡ 87	⌫ 88	‰ 89	Š 8A	‹ 8B	Œ 8C	↩ 8D	^n 8E	^o 8F
9	^p 90	‘ 91	’ 92	“ 93	” 94	95 95	96 96	97 97	˜ 98	™ 99	š 9A	› 9B	œ 9C	9D 9D	9E 9E	Ÿ 9F
A	␣ A0	¡ A1	¢ A2	£ A3	¤ A4	¥ A5	¦ A6	§ A7	¨ A8	© A9	ª AA	« AB	¬ AC	- AD	® AE	¯ AF
B	° B0	± B1	² B2	³ B3	´ B4	µ B5	¶ B6	· B7	B8 B8	¹ B9	º BA	» BB	¼ BC	½ BD	¾ BE	¿ BF
C	C0 C0	C1 C1	C2 C2	C3 C3	C4 C4	C5 C5	C6 C6	C7 C7	C8 C8	C9 C9	CA CA	CB CB	CC CC	CD CD	CE CE	CF CF
D	D0 D0	D1 D1	D2 D2	D3 D3	D4 D4	D5 D5	D6 D6	× D7	D8 D8	D9 D9	DA DA	DB DB	DC DC	DD DD	Þ DE	ß DF
E	E0 E0	E1 E1	E2 E2	E3 E3	E4 E4	E5 E5	æ E6	E7 E7	E8 E8	E9 E9	EA EA	EB EB	EC EC	ED ED	EE EE	EF EF
F	F0 F0	F1 F1	F2 F2	F3 F3	F4 F4	F5 F5	F6 F6	÷ F7	ø F8	F9 F9	FA FA	FB FB	FC FC	FD FD	FE FE	FF FF

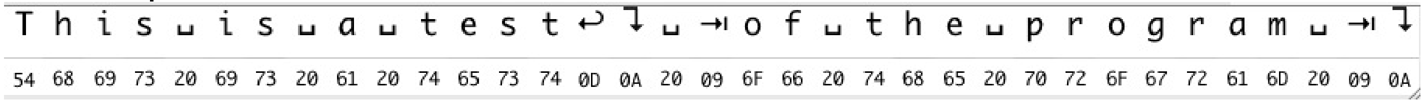
Figure 1 - A sample forensic font for the US ASCII character set

This table is converted into a JPEG output file, and using the freely available and widely used "convert" program, a JPEG file corresponding to each ASCII code is extracted, with each placed in a file named with either an F or an S followed by the two character HEX value followed by the extension ".jpg".

A simple conversion for display may be done by a program that extracts the hex value for each byte in the input file, and produces an HTML output file consisting of a sequence of image tags, with each image tag corresponding to the JPEG file associated with the ASCII code of the byte value. A simple version of such a program is provided as a Unix shell script in Figure 10 later in this paper.

The sample conversion program also provides for specifying a width and height of the displayed output by using HTML tags, and provides for the addition of new lines after user-defined end of line characters (e.g., 0A). For example, the conversion in Figure 2 is demonstrative (commands in bold).

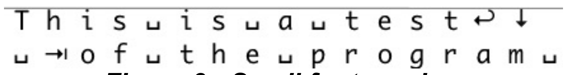
```
>echo "This is a test^M
      of the program"
This is a test
        of the program
>!! | ff
```



```
T h i s ␣ i s ␣ a ␣ t e s t ↩ ↴ ␣ ⇥ o f ␣ t h e ␣ p r o g r a m ␣ ⇥ ↴
54 68 69 73 20 69 73 20 61 20 74 65 73 74 0D 0A 20 09 6F 66 20 74 68 65 20 70 72 6F 67 72 61 6D 20 09 0A
```

Figure 2 - A depiction using a forensic font.

Figure 2 demonstrates the resulting output display, indicating the presence of the space prior to the tab, and the presence of the carriage return prior to the newline. A "small" version of the font is also usable (Figure 3) to depict all of the characters without including the HEX digits. All HEX values are still displayed and differentiable, and the content is relatively readable. In some cases, this presentation is more useful when readability is more important that the byte values involved.



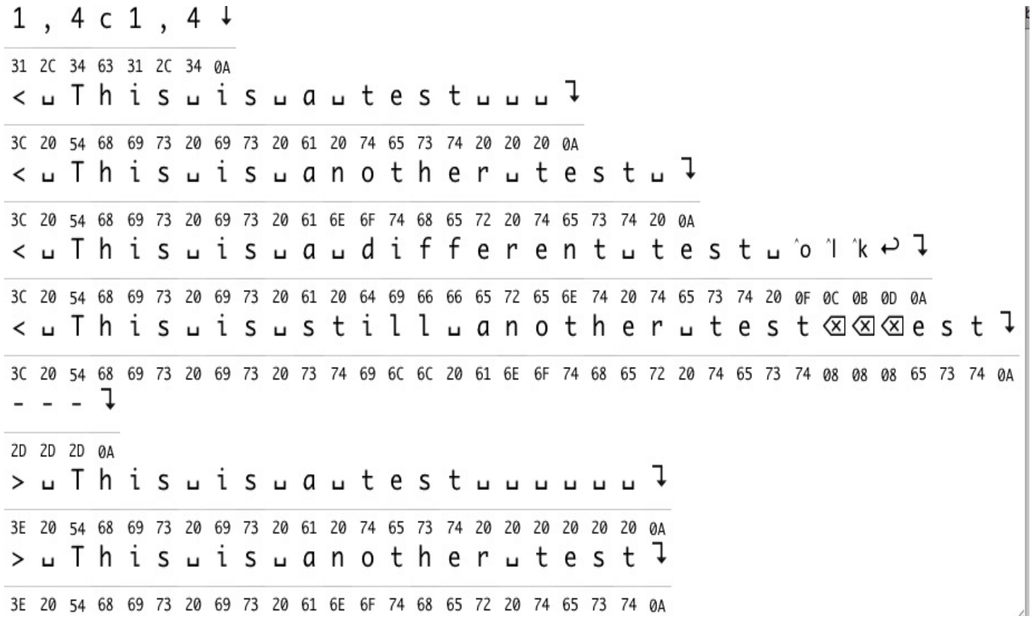
```
T h i s ␣ i s ␣ a ␣ t e s t ↩ ↓
␣ ⇥ o f ␣ t h e ␣ p r o g r a m ␣
```

Figure 3 - Small font version

Output of a "diff" command

As a case example, a recent forensics case involved the use of the diff command between two sequences. The output of this command demonstrated that there was a difference in lines of the sequences that appeared to be identical on the output display. In the actual case, no forensic phone was available, and it wasn't immediately obvious what the differences were. As a result, some time and effort were wasted, and in a less careful examination, portions of the results might have been missed. By applying the forensic font, the difference becomes immediately obvious. The sample shown in Figure 4 demonstrates such a difference (commands are indicated in boldface). In this case, the displayed output is presented next to the input, and typically appears in a separate windows (since the display is not available in the command terminal window.

```
FF>diff test1 test2

1,4c1,4
< This is a test
< This is another test
< This is a different test

< This is still another test
---
> This is a test
> This is another test
> This is a different test
> This is still another test
FF>diff test1 test2 | ff
```



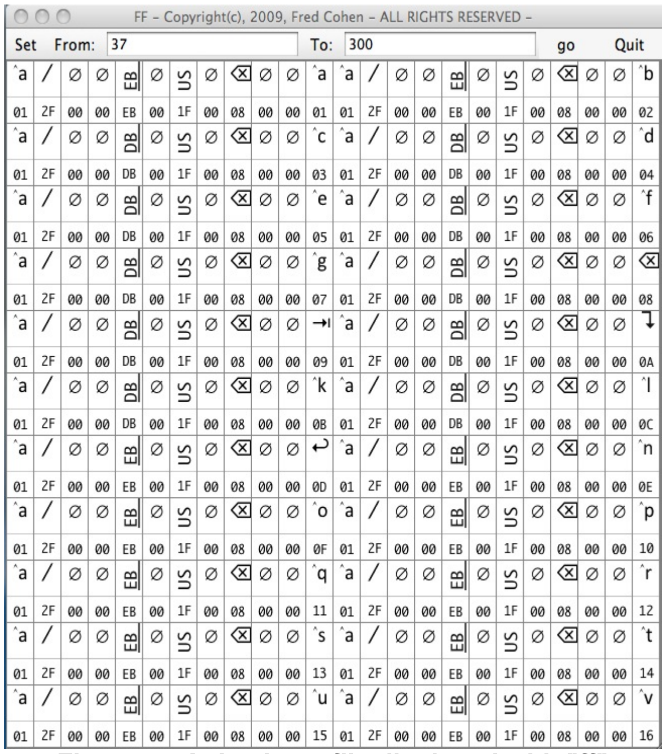
```
1 , 4 c 1 , 4 ↓
31 2C 34 63 31 2C 34 0A
< ␣ T h i s ␣ i s ␣ a ␣ t e s t ␣ ␣ ␣ ↴
3C 20 54 68 69 73 20 69 73 20 61 20 74 65 73 74 20 20 20 0A
< ␣ T h i s ␣ i s ␣ a n o t h e r ␣ t e s t ␣ ↴
3C 20 54 68 69 73 20 69 73 20 61 6E 6F 74 68 65 72 20 74 65 73 74 20 0A
< ␣ T h i s ␣ i s ␣ a ␣ d i f f e r e n t ␣ t e s t ␣ ^O ^L ^K ↩ ↴
3C 20 54 68 69 73 20 69 73 20 61 20 64 69 66 66 65 72 65 6E 74 20 74 65 73 74 20 0F 0C 0B 0D 0A
< ␣ T h i s ␣ i s ␣ s t i l l ␣ a n o t h e r ␣ t e s t ⌫ ⌫ ⌫ e s t ↴
3C 20 54 68 69 73 20 69 73 20 73 74 69 6C 6C 20 61 6E 6F 74 68 65 72 20 74 65 73 74 08 08 08 65 73 74 0A
- - - ↴
2D 2D 2D 0A
> ␣ T h i s ␣ i s ␣ a ␣ t e s t ␣ ␣ ␣ ␣ ␣ ␣ ↴
3E 20 54 68 69 73 20 69 73 20 61 20 74 65 73 74 20 20 20 20 20 20 0A
> ␣ T h i s ␣ i s ␣ a n o t h e r ␣ t e s t ↴
3E 20 54 68 69 73 20 69 73 20 61 6E 6F 74 68 65 72 20 74 65 73 74 0A
```

Figure 4 - the output of a "diff" command using a forensic font

Using the forensic font depiction, the differences are immediately obvious; test1 has 3 spaces before the end of line, while test2 has 6; in the next line, test1 has an extra space before the end of line; in the 3rd line, there are control characters in test1; and the last line of test1 has 3 backspaces causing e, s, and t, to be overwritten with identical characters in test1. Only the presence of differences in the third line are demonstrated in the normal output of the diff command.

An example from a database file

Another example where forensics font is useful, is in depicting and reviewing the contents of a fixed width file, or other similar database content. For example, in examining a binary file is part of the storage of the OSX Spotlight system, a system used to allow contents of the Macintosh to be rapidly searched by the user, the file format is not immediately apparent, and there are various binary characters present between strings. By presenting these results using a forensics font, the database structure can be seen with some additional clarity, albeit the use of such a font does not eliminate all uncertainty. Figure 5 provides an example of the depiction of a WK4 formatted file (a worksheet from a spreadsheet program). In this example, the width of the display was adjusted until the characters aligned. The result is that using the visual capacity of the human observer, and structure alignment of fields within this file can be readily ascertained in a matter of seconds. Different portions of the file might have different periodicity, and further adjustments can be made on a region by region basis to gain insight into the content, and to allow further examination to proceed. Using this tool, it's often easy to find the limiters, and with those limiters, the tool can be used specifying a delimiter to be replaced by new lines. In this instance, the fixed width of the forensic font is particularly helpful in finding underlying data structure. However, when there is flexible data structure, additional tools may be helpful. As an example, using a table depiction may allow table areas to be differentiated by the presence of characters within the input file, with table rows differentiated by different symbols. Again, using the ability to change widths dynamically, and the forensics fonts fixed width and display of all characters and character codes, the examiner can rapidly detect structure, to pick that structure, and undertake more detailed examination.



```
From: 37                To: 300
^a  /  ∅  ∅  EB ∅  US ∅  ⌫  ∅  ∅  ^a ^a  /  ∅  ∅  EB ∅  US ∅  ⌫  ∅  ∅  ^b
01  2F 00 00 EB 00 1F 00 08 00 00 01 01  2F 00 00 EB 00 1F 00 08 00 00 02
^a  /  ∅  ∅  DB ∅  US ∅  ⌫  ∅  ∅  ^c ^a  /  ∅  ∅  DB ∅  US ∅  ⌫  ∅  ∅  ^d
01  2F 00 00 DB 00 1F 00 08 00 00 03 01  2F 00 00 DB 00 1F 00 08 00 00 04
^a  /  ∅  ∅  DB ∅  US ∅  ⌫  ∅  ∅  ^e ^a  /  ∅  ∅  DB ∅  US ∅  ⌫  ∅  ∅  ^f
01  2F 00 00 DB 00 1F 00 08 00 00 05 01  2F 00 00 DB 00 1F 00 08 00 00 06
^a  /  ∅  ∅  DB ∅  US ∅  ⌫  ∅  ∅  ^g ^a  /  ∅  ∅  DB ∅  US ∅  ⌫  ∅  ∅  ⌫
01  2F 00 00 DB 00 1F 00 08 00 00 07 01  2F 00 00 DB 00 1F 00 08 00 00 08
^a  /  ∅  ∅  DB ∅  US ∅  ⌫  ∅  ∅  ⇥  ^a  /  ∅  ∅  DB ∅  US ∅  ⌫  ∅  ∅  ↴
01  2F 00 00 DB 00 1F 00 08 00 00 09 01  2F 00 00 DB 00 1F 00 08 00 00 0A
^a  /  ∅  ∅  DB ∅  US ∅  ⌫  ∅  ∅  ^k ^a  /  ∅  ∅  DB ∅  US ∅  ⌫  ∅  ∅  ^l
01  2F 00 00 DB 00 1F 00 08 00 00 0B 01  2F 00 00 DB 00 1F 00 08 00 00 0C
^a  /  ∅  ∅  EB ∅  US ∅  ⌫  ∅  ∅  ↩  ^a  /  ∅  ∅  EB ∅  US ∅  ⌫  ∅  ∅  ^n
01  2F 00 00 EB 00 1F 00 08 00 00 0D 01  2F 00 00 EB 00 1F 00 08 00 00 0E
^a  /  ∅  ∅  EB ∅  US ∅  ⌫  ∅  ∅  ^o ^a  /  ∅  ∅  EB ∅  US ∅  ⌫  ∅  ∅  ^p
01  2F 00 00 EB 00 1F 00 08 00 00 0F 01  2F 00 00 EB 00 1F 00 08 00 00 10
^a  /  ∅  ∅  EB ∅  US ∅  ⌫  ∅  ∅  ^q ^a  /  ∅  ∅  EB ∅  US ∅  ⌫  ∅  ∅  ^r
01  2F 00 00 EB 00 1F 00 08 00 00 11 01  2F 00 00 EB 00 1F 00 08 00 00 12
^a  /  ∅  ∅  EB ∅  US ∅  ⌫  ∅  ∅  ^s ^a  /  ∅  ∅  EB ∅  US ∅  ⌫  ∅  ∅  ^t
01  2F 00 00 EB 00 1F 00 08 00 00 13 01  2F 00 00 EB 00 1F 00 08 00 00 14
^a  /  ∅  ∅  EB ∅  US ∅  ⌫  ∅  ∅  ^u ^a  /  ∅  ∅  EB ∅  US ∅  ⌫  ∅  ∅  ^v
01  2F 00 00 EB 00 1F 00 08 00 00 15 01  2F 00 00 EB 00 1F 00 08 00 00 16
```

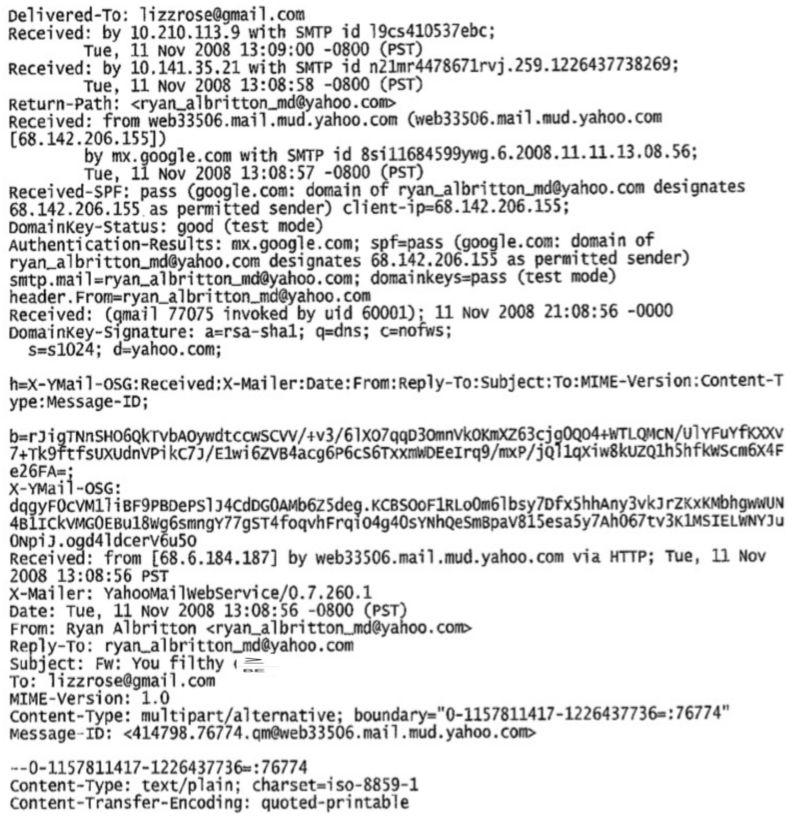
Figure 5 - A database file displayed with "ff"

An example from a printout

Another quite common example occurs when attempting to print. Many printing mechanisms perform alterations on the original input in order to depict it within the output media, as well as for the purpose of pleasant appearance. While this is certainly useful for many purposes, in presenting content of forensic value, it is often more important for the output to precisely and accurately reflect the input, and far less important that the output look pleasing. Among the more common problems faced by the examiner looking at a printout, are the misalignment of tab stops; the kerning of characters and spaces so as to make the actual number or presence of spaces unclear, and the alignment of characters from line to line uneven; the removal of individual or consolidation of multiple blank lines; the continuation of characters from one line into subsequent lines when the depiction is not wide enough to fully display the characters in a line within a single line (wrap around); and the removal, or different uses of nonprinting characters.

Each of these issues produce difficulties for the examiner, both in understanding and explaining the printout, and difficulties in the trier of fact and legal counsel in understanding the testimony. Figure 6 (slightly redacted) contains the output as presented in testimony in a legal matter.[3]

Output like this is problematic in terms of identifying precisely what byte sequences were present within the original content, and more specifically, it makes it difficult to determine where the header of the email sent stops and where the body of that same email begins.



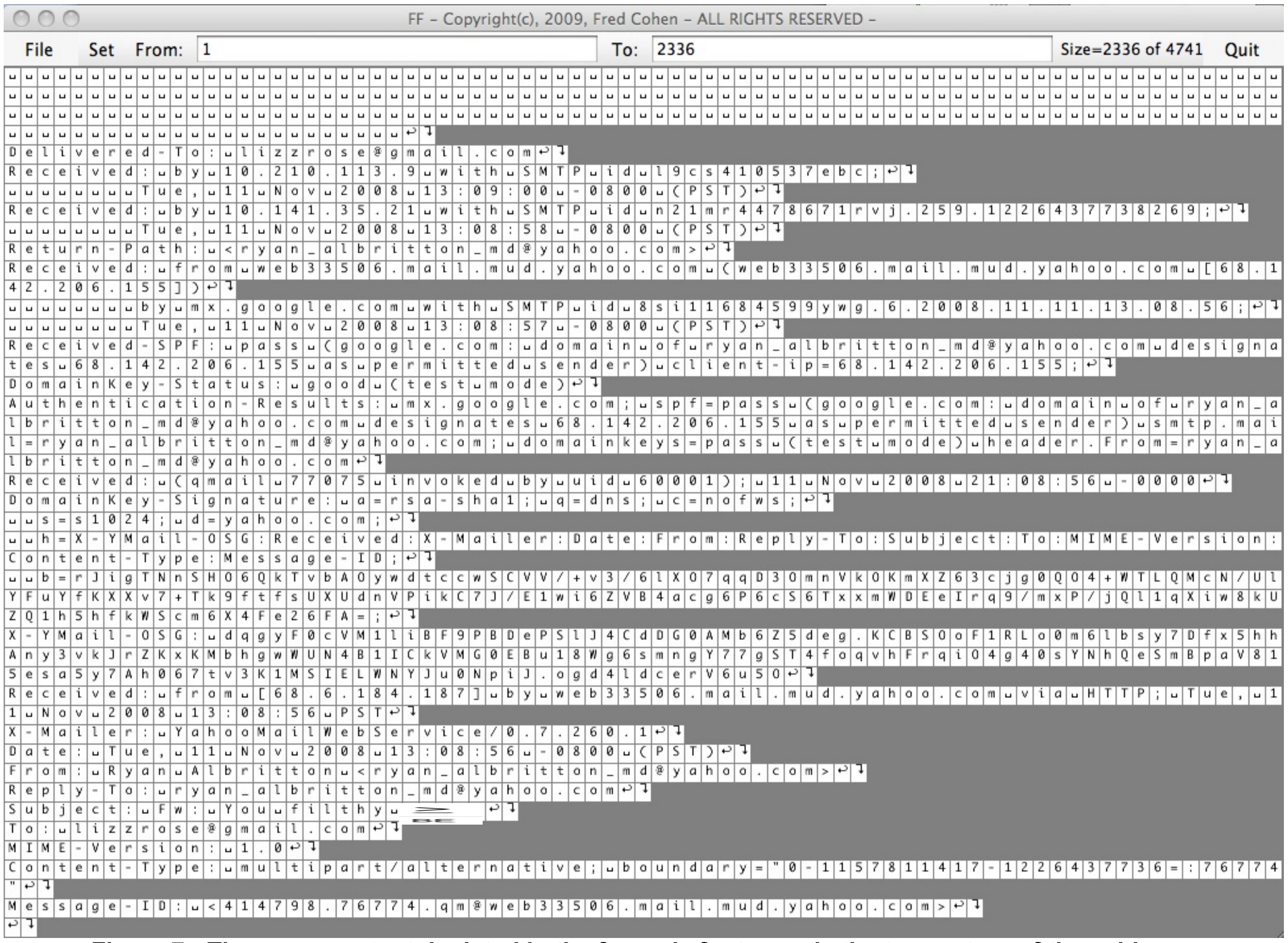
```
Delivered-To: lizzrose@gmail.com
Received: by 10.210.113.9 with SMTP id l9cs410537ebc;
        Tue, 11 Nov 2008 13:09:00 -0800 (PST)
Received: by 10.141.35.21 with SMTP id n21mr4478671rvj.259.1226437738269;
        Tue, 11 Nov 2008 13:08:58 -0800 (PST)
Return-Path: <ryan_albritton_md@yahoo.com>
Received: from web33506.mail.mud.yahoo.com (web33506.mail.mud.yahoo.com
[68.142.206.155])
        by mx.google.com with SMTP id 8si11684599ywg.6.2008.11.11.13.08.56;
        Tue, 11 Nov 2008 13:08:57 -0800 (PST)
Received-SPF: pass (google.com: domain of ryan_albritton_md@yahoo.com designates
68.142.206.155 as permitted sender) client-ip=68.142.206.155;
DomainKey-Status: good (test mode)
Authentication-Results: mx.google.com; spf=pass (google.com: domain of
ryan_albritton_md@yahoo.com designates 68.142.206.155 as permitted sender)
smtp.mail=ryan_albritton_md@yahoo.com; domainkeys=pass (test mode)
header.From=ryan_albritton_md@yahoo.com
Received: (qmail 77075 invoked by uid 60001); 11 Nov 2008 21:08:56 -0000
DomainKey-Signature: a=rsa-sha1; q=dns; c=nofws;
  s=s1024; d=yahoo.com;

h=X-YMail-OSG:Received:X-Mailer:Date:From:Reply-To:Subject:To:MIME-Version:Content-T
ype:Message-ID;

b=rJigTNnSHO6QkTvbAOywdtccwSCVV/+v3/6lXO7qqD3OmnVkOKmXZ63cjg0Q04+WTLQMCN/UlYFuYfKXXv
7+Tk9ftfsUXUdnVPikC7J/E1wi6ZVB4acg6P6cS6TxxmWDEeIrq9/mxP/jQl1qXiw8kUZQ1h5hfKWScm6X4F
e26FA=;
X-YMail-OSG:
dqgyF0cVM1liBF9PBDePSlJ4CdDG0AMb6Z5deg.KCBSOoF1RLoOm6lbsy7Dfx5hhAny3vkJrZKxKMbhgwWUN
4B1ICkVMG0EBu18Wg6smngY77gST4foqvhFrqiO4g4OsYNhQeSmBpaV815esa5y7Ah067tv3K1MSIELWNYJu
ONpiJ.ogd4ldcerV6u5O
Received: from [68.6.184.187] by web33506.mail.mud.yahoo.com via HTTP; Tue, 11 Nov
2008 13:08:56 PST
X-Mailer: YahooMailWebService/0.7.260.1
Date: Tue, 11 Nov 2008 13:08:56 -0800 (PST)
From: Ryan Albritton <ryan_albritton_md@yahoo.com>
Reply-To: ryan_albritton_md@yahoo.com
Subject: Fw: You filthy [redacted]
To: lizzrose@gmail.com
MIME-Version: 1.0
Content-Type: multipart/alternative; boundary="0-1157811417-1226437736=:76774"
Message-ID: <414798.76774.qm@web33506.mail.mud.yahoo.com>

--0-1157811417-1226437736=:76774
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: quoted-printable
```

Figure 6 - PDF printer formatted output

In particular, and without limit; there is what appears to be a blank line after the line that appears to end in "d=yahoo.com", and another such appearance after the line that appears to end in "Message-ID;", and another such appearance after the line that appears to end in "yahoo.com>"; and there are a number of lines that appear to start with sequences of characters with no leading blanks, and that do not have a ":" prior to the first blank area in their lines, indicating that they are not continuation lines from a previous header, and that they are not themselves valid headers for an email (e.g., the apparent line that appears to read "[68.142.206.155])").

Many readers may shrug as this example, because, to them it may be obvious that these are cases where wrap-around and printer attempts to not have words wrap around from line to line cased these depictions. But these are assumptions that, in forensics matters, may be critical issues, and these assumptions may not in fact be true. In fact, in this particular matter, the output from Figure 6 was placed in front of an examiner who could not definitively authenticate what was actually present. One of the parties was asserting that this particular claimed email was a forgery. One of the claims of forgery surrounded the non-printable symbols contained in the messages and formats of headers.

Figure 7 presents the header portion of the actual file (slightly redacted) using the forensic font (short form). In this case, it is immediately clear that there are a series of leading space characters followed by the content of the header, that <CR><LF> ends the lines, and that the apparent end of the header area in the PDF depiction from Figure 6 does not accurately reflect the lack of a second linefeed at that point in the original content. In testifying with regard to the output partially depicted in Figure 6, the expert had to indicate that the depiction was unclear and that recollection alone would have to be used to identify what was present in the header portion of this message. Figure 7 is definitive.



FF – Copyright(c), 2009, Fred Cohen – ALL RIGHTS RESERVED –

File Set From: 1 To: 2336 Size=2336 of 4741 Quit

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣↩↴
Delivered-To:␣lizzrose@gmail.com↩↴
Received:␣by␣10.210.113.9␣with␣SMTP␣id␣l9cs410537ebc;↩↴
␣␣␣␣␣␣␣␣Tue,␣11␣Nov␣2008␣13:09:00␣-0800␣(PST)↩↴
Received:␣by␣10.141.35.21␣with␣SMTP␣id␣n21mr4478671rvj.259.1226437738269;↩↴
␣␣␣␣␣␣␣␣Tue,␣11␣Nov␣2008␣13:08:58␣-0800␣(PST)↩↴
Return-Path:␣<ryan_albritton_md@yahoo.com>↩↴
Received:␣from␣web33506.mail.mud.yahoo.com␣(web33506.mail.mud.yahoo.com␣[68.1
42.206.155])↩↴
␣␣␣␣␣␣␣␣by␣mx.google.com␣with␣SMTP␣id␣8si1168459ywg.6.2008.11.11.13.08.56;↩↴
␣␣␣␣␣␣␣␣Tue,␣11␣Nov␣2008␣13:08:57␣-0800␣(PST)↩↴
Received-SPF:␣pass␣(google.com:␣domain␣of␣ryan_albritton_md@yahoo.com␣designa
tes␣68.142.206.155␣as␣permitted␣sender)␣client-ip=68.142.206.155;↩↴
DomainKey-Status:␣good␣(test␣mode)↩↴
Authentication-Results:␣mx.google.com;␣spf=pass␣(google.com:␣domain␣of␣ryan_a
lbritton_md@yahoo.com␣designates␣68.142.206.155␣as␣permitted␣sender)␣smtp.mai
l=ryan_albritton_md@yahoo.com;␣domainkeys=pass␣(test␣mode)␣header.From=ryan_a
lbritton_md@yahoo.com↩↴
Received:␣(qmail␣77075␣invoked␣by␣uid␣60001);␣11␣Nov␣2008␣21:08:56␣-0000↩↴
DomainKey-Signature:␣a=rsa-sha1;␣q=dns;␣c=nofws;↩↴
␣␣s=s1024;␣d=yahoo.com;↩↴
␣␣h=X-YMail-OSG:Received:X-Mailer:Date:From:Reply-To:Subject:To:MIME-Version:
Content-Type:Message-ID;↩↴
␣␣b=rJigTNnSH06QkTvbAOywdtccwSCVV/+v3/6lX07qqD30mnVkOKmXZ63cjg0Q04+WTLQMcN/Ul
YFuYfKXXv7+Tk9ftfsUXUdnVPikC7J/E1wi6ZVB4acg6P6cS6TxxmWDEeIrq9/mxP/jQl1qXiw8kU
ZQ1h5hfkWScm6X4Fe26FA=;↩↴
X-YMail-OSG:␣dqgyF0cVM1liBF9PBDePSlJ4CdDG0AMb6Z5deg.KCBSOoF1RLo0m6lbsy7Dfx5hh
Any3vkJrZKxKMbhgwWUN4B1ICkVMG0EBu18Wg6smngY77gST4foqvhFrqi04g40sYNhQeSmBpaV81
5esa5y7Ah067tv3K1MSIELWNYJu0NpiJ.ogd4ldcerV6u50↩↴
Received:␣from␣[68.6.184.187]␣by␣web33506.mail.mud.yahoo.com␣via␣HTTP;␣Tue,␣1
1␣Nov␣2008␣13:08:56␣PST↩↴
X-Mailer:␣YahooMailWebService/0.7.260.1↩↴
Date:␣Tue,␣11␣Nov␣2008␣13:08:56␣-0800␣(PST)↩↴
From:␣Ryan␣Albritton␣<ryan_albritton_md@yahoo.com>↩↴
Reply-To:␣ryan_albritton_md@yahoo.com↩↴
Subject:␣Fw:␣You␣filthy␣[illegible]↩↴
To:␣lizzrose@gmail.com↩↴
MIME-Version:␣1.0↩↴
Content-Type:␣multipart/alternative;␣boundary="0-1157811417-1226437736=:76774
"↩↴
Message-ID:␣<414798.76774.qm@web33506.mail.mud.yahoo.com>↩↴
↩↴
```

Figure 7 - The same content depicted in the forensic font reveals the true nature of the evidence

In Figure 7, spacing between depicted font elements is used to provide clarity around symbol separations. The fixed width font combined with the depiction of all bytes provides clarity around what is actually present, even when the output results are wrap-around, and even when, as in large figures placed in smaller printouts, the display is degraded.

By placing depictions such as Figure 7 in evidence, optionally along with other depictions that provide better human readability, the problem of the forensic examiner making guesses about what might have happened is largely eliminated and precise results become immediately evident. Of course this does not eliminate all of the potential misinterpretations or assumptions that might be made by the examiner, but it certainly may eliminate those related to the identified sorts of errors.

A tool to apply forensic fonts

A tool was developed to experiment with forensic fonts, and in particular, for use in matters such as those identified herein. The name of the tool is "FF", and it is available for free download from http://all.net/. This particular implementation was written in java and is provided as a "jar" file with executable and the files associated with the defined fonts identified herein. It provides a graphical interface to allow the user to display a file in the forensic font, and to manipulate the depictions for the sorts of examples described herein. This includes, without limit:

- The presentation of a unique depiction for each of the eight bit ASCII character codes.
- The ability to resize and reshape the output window for ease of alignment.
- The ability to display bytes starting at any location within the input.
- The ability to define end of line characters for the depiction.
- The ability to select between full fonts and smaller versions of that font without the hex digits.
- The ability to resize the font as depicted.

Using this tool, all of the depictions shown in this paper were generated, screenshots were taken, and those screenshots were used for the presentations.

Validation of the tool

The key properties of key tool for forensic fonts include, without limit:

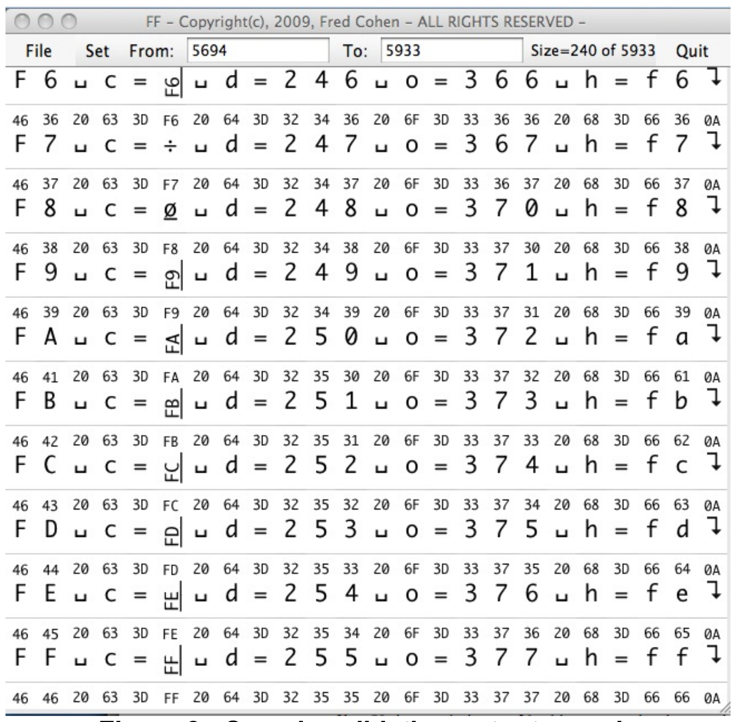
1. The tool must not alter original writing.
2. Displayed information must be complete.
3. Displayed information must be accurate.

While additional properties might be desired from a user standpoint, these are the key properties that must be verified and validated for forensic soundness.

Property 1 is true of the current tool because it takes only input from the standard input, files and the user, and produces output only to the screen. It has no code that does any file output.

Property 2 cannot be proven in the same sense as Property 1. In fact, we know that in general, Property 2 cannot be true because the input is of indeterminate length, and the design of this tool is such that it must store everything it is able to display. Thus, for some input length, the displayed information will not be complete. The normal behavior is an error message indicating inadequate memory, and the program does not continue to operate normally in the sense of responding to user input. This has been validated by testing.

Property 3 also cannot be proven in the same sense as Property 1. While we can validate that for provided inputs, outputs accurately reflect the desired behavior, it is infeasible to test all possible input sequences and to verify the output in all such cases. To do so would require, at a minimum, a validation of the underlying operating environment, and this is infeasible. Thus, the best we can do, is to take a testing approach, and validate the available code to the extent feasible for the purpose.



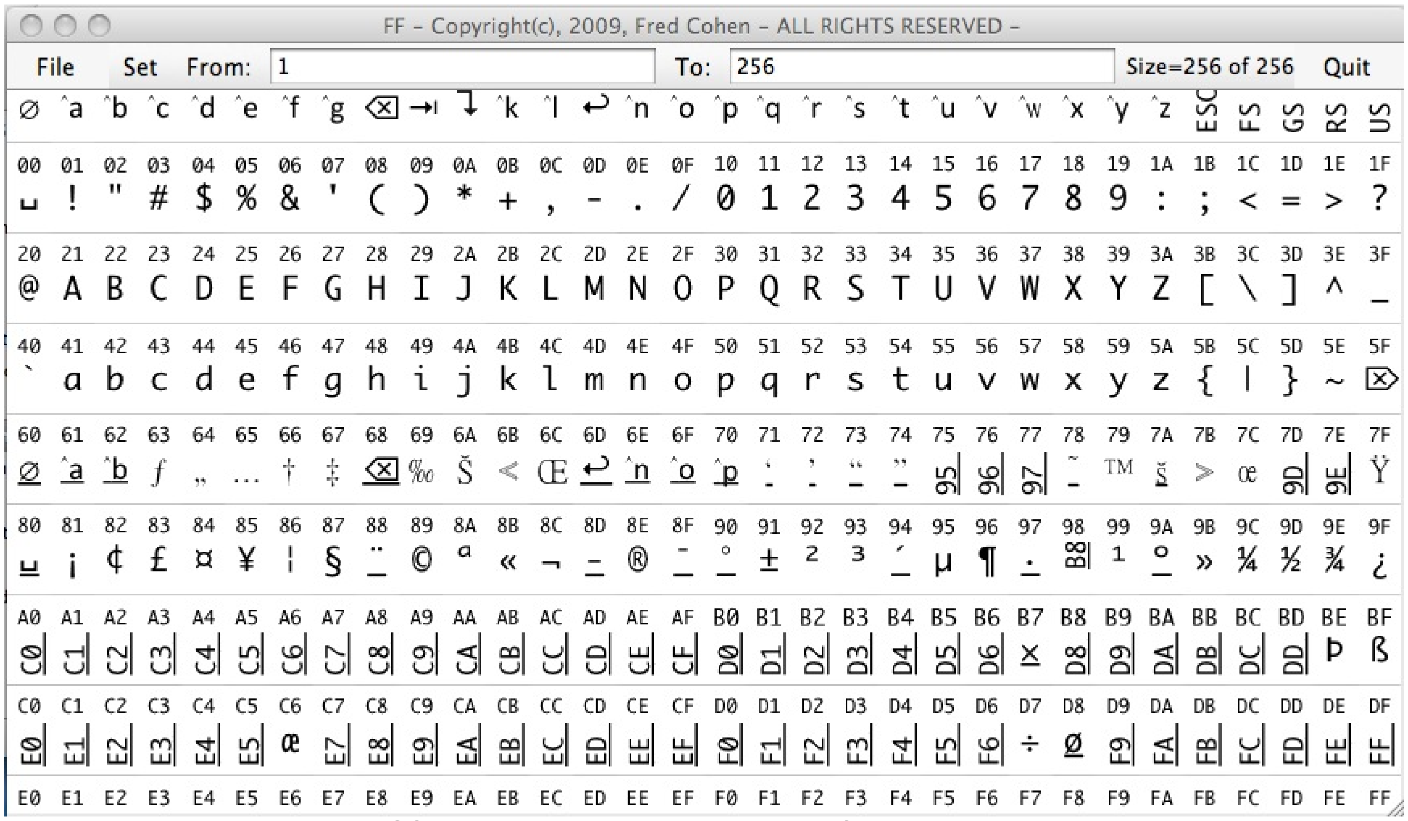
```
From: 5694    To: 5933    Size=240 of 5933
F  6  ␣  c  =  F6 ␣  d  =  2  4  6  ␣  o  =  3  6  6  ␣  h  =  f  6  ↴
46 36 20 63 3D F6 20 64 3D 32 34 36 20 6F 3D 33 36 36 20 68 3D 66 36 0A
F  7  ␣  c  =  ÷  ␣  d  =  2  4  7  ␣  o  =  3  6  7  ␣  h  =  f  7  ↴
46 37 20 63 3D F7 20 64 3D 32 34 37 20 6F 3D 33 36 37 20 68 3D 66 37 0A
F  8  ␣  c  =  ø  ␣  d  =  2  4  8  ␣  o  =  3  7  0  ␣  h  =  f  8  ↴
46 38 20 63 3D F8 20 64 3D 32 34 38 20 6F 3D 33 37 30 20 68 3D 66 38 0A
F  9  ␣  c  =  F9 ␣  d  =  2  4  9  ␣  o  =  3  7  1  ␣  h  =  f  9  ↴
46 39 20 63 3D F9 20 64 3D 32 34 39 20 6F 3D 33 37 31 20 68 3D 66 39 0A
F  A  ␣  c  =  FA ␣  d  =  2  5  0  ␣  o  =  3  7  2  ␣  h  =  f  a  ↴
46 41 20 63 3D FA 20 64 3D 32 35 30 20 6F 3D 33 37 32 20 68 3D 66 61 0A
F  B  ␣  c  =  FB ␣  d  =  2  5  1  ␣  o  =  3  7  3  ␣  h  =  f  b  ↴
46 42 20 63 3D FB 20 64 3D 32 35 31 20 6F 3D 33 37 33 20 68 3D 66 62 0A
F  C  ␣  c  =  FC ␣  d  =  2  5  2  ␣  o  =  3  7  4  ␣  h  =  f  c  ↴
46 43 20 63 3D FC 20 64 3D 32 35 32 20 6F 3D 33 37 34 20 68 3D 66 63 0A
F  D  ␣  c  =  FD ␣  d  =  2  5  3  ␣  o  =  3  7  5  ␣  h  =  f  d  ↴
46 44 20 63 3D FD 20 64 3D 32 35 33 20 6F 3D 33 37 35 20 68 3D 66 64 0A
F  E  ␣  c  =  FE ␣  d  =  2  5  4  ␣  o  =  3  7  6  ␣  h  =  f  e  ↴
46 45 20 63 3D FE 20 64 3D 32 35 34 20 6F 3D 33 37 36 20 68 3D 66 65 0A
F  F  ␣  c  =  FF ␣  d  =  2  5  5  ␣  o  =  3  7  7  ␣  h  =  f  f  ↴
46 46 20 63 3D FF 20 64 3D 32 35 35 20 6F 3D 33 37 37 20 68 3D 66 66 0A
```

Figure 8 - Sample validation output sample

Several tests were made based on systematic generation of known byte sequences and verification that the displayed sequences matched the generated ones. Because of various peculiarities of the "java" language (i.e., not having unsigned byte values, various display limitations, storage limits, and rendering approaches) the validation runs produced a number of anomalies that were corrected.

The tool is depicted in the screenshot provided in Figure 8 and throughout this paper. Figure 8 shows the output of a file generated for depicting the ASCII character codes, beginning with the 2-character HEX value for the byte, a space, the string c= followed by the printed character, a space, the decimal value of the byte, a space, the octal value of the byte, a space, the HEX value of the byte, and an end of line character. This depiction begins a new line every time the end of line character is seen. Validation of the locations of characters within the file, the size of the file, the accuracy of the depictions to the table, and the hex values shown have been made. A similar input sequence containing all bytes in increasing order has been used to generate tables similar to the one depicted in Table 2, and to produce different alignments of this and other tables. This was produced by the C program below and is shown in Figure 9:

```
main(){char i;int j;for (j=0;j<256;j++) {i=j;write(1,&j,1);}
```



```
∅  ^a ^b ^c ^d ^e ^f ^g ⌫  ⇥  ↧  ^k ^l ↩  ^n ^o ^p ^q ^r ^s ^t ^u ^v ^w ^x ^y ^z ESC FS GS RS US
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
␣  !  "  #  $  %  &  '  (  )  *  +  ,  -  .  /  0  1  2  3  4  5  6  7  8  9  :  ;  <  =  >  ?
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
@  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z  [  \  ]  ^  _
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
`  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z  {  |  }  ~  ⌦
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
∅  ^a ^b ƒ  „  …  †  ‡  ⌫  ‰  Š  ‹  Œ  ↩  ^n ^o ^p ‘  ’  “  ”  95 96 97 ˜  ™  š  ›  œ  9D 9E Ÿ
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
␣  ¡  ¢  £  ¤  ¥  ¦  §  ¨  ©  ª  «  ¬  -  ®  ¯  °  ±  ²  ³  ´  µ  ¶  ·  B8 ¹  º  »  ¼  ½  ¾  ¿
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 ×  D8 D9 DA DB DC DD Þ  ß
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 æ  E7 E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 ÷  ø  F9 FA FB FC FD FE FF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
```

Figure 9 - The ASCII character set generated with a C program as displayed by FF

Accuracy is, of course, relative to specifications. Note that the "From:" "To:" fields are inclusive (i.e. the display in Figure 9 goes "From" the "1st" byte and "To" the "256th" byte. Thus the number of bytes displayed should be the "To" value less the "From" value plus 1 (i.e., 256). It is also possible to request a display of 0 bytes by inputting a "To" value less than the "From" value, and in such cases, no bytes will be displayed and the "To" value will be one less than the "From" value. It is also possible to request regions of the input outside of the actual range of the input. In such cases, the program corrects, displaying only from the start or to the end of the input sequence. These have also all been tested in the current implementation.

Validation of visualization tools are problematic in several ways. In particular, the thing being validated about them is the cognitive properties associated with the human observer of the output as it relates to the actuality of the latent input. Furthermore, the examination of results requires that the human examiner view and interpret the validation results. Validation may work on one system with one sort of display and fail on a different system with a different display because of properties that make for better or worse readability. Different font sizes may conceal details that end up misinforming the examiner because it is too hard to differentiate between appearances, and proper performance under test conditions does not guarantee proper performance under real world operation. Every tool has its limits, and the "FF" tool provided as an example is limited in the size of input file it can use, the size of display it can handle, and its performance is poor for large displays with small font sizes. Nonetheless, it has proven useful in an increasing number of cases where accurate depiction is meaningful to the matter at hand.

Producing results

Production of results from forensic fonts may come, without limit, in the form of pieces of paper, presentations using a computer display screen, or in files provided in digital form. A major goal of the use of forensic fonts is that they be equally informative in all such forms. The production of digital forensic evidence, which is latent by nature, requires that the tools used to produce it are reliable and suited for the purpose, that the methodology meets the requirements of scientific rigor, and that it is properly applied.[4] If the production is done with inadequate resolution to make the fonts readable or if the presentation method fails to properly display all of the symbols in proper sequence and placement, the use of forensic fonts will not alter those conditions. However, because the forensic font is self-indicating as to the underlying bit sequence present in the trace, the content should be clear to the properly skilled observer.

An alternative software approach for presenting and printing larger volumes of material in the forensic font uses a Web browser and the hypertext markup language (HTML). Figure10 shows the code for this approach. This example shell script uses the "hexdump" program and text replacement to create an HTML file that is about 20-30 times as large as the original content and that causes the Web browser to display the result by rendering a series of graphical files aligned so as to depict the desired results. This particular script has not been verified as to the criteria above.

```
echo "fixer [FS] S0A.jpg test 0A 12 32"
echo "  Full or Small - input file - output file (.html) break [width [height]]"
Font=$1;shift;InFile=$1;shift;OutFile=$1;shift;Break=$1;shift
if test "X$1" == "X"; then WL="";
  else WL=" width=\"$1\"";shift
  if test "X$1" == "X"; then WL="$WL"; else WL="$WL height=\"$1\""; shift; fi;fi
for i in `hexdump -v $InFile | toupper |
        while read a b; do echo $b;done`; do
                echo -n "<img src=$Font$i.jpg hspace=\"0\" vspace=\"0\"$WL>";
        if test "$i" == "$Break"; then echo "<br>"; fi; done > $OutFile.html
```

Figure 10 - A simple forensics font tool for use with a Web browser

Web browsers typically have print functions, and on some platforms, these functions allow the output to be printed or saved as portable document format (pdf) files or postsript (ps) files. In these systems, this process may be used to produce a printable version that is relatively portable and can be sent electronically from place to place. PDF files are commonly used in legal productions within the United States today, and as a result, this may be the most useful solution for the time being.

Summary, conclusions, further work

There are clearly limitations associated with forensic fonts. These include, without limit, coping with alignment such as tab stops, differences in appearance between the forensic font display in the normal display seen by user, the additional information present in the forensic font that makes cognition of content somewhat slower, and the fact that one size simply does not fit all. Nonetheless, it appears that forensic fonts are a valuable tool for more rapidly understanding the byte sequences present, for reducing content miss errors in the interpretation process, and for presenting forensic output to others when the underlying bit sequences are relevant to the issues in the case.

In the current implementation, the mechanisms used to transform a display a rudimentary, and were designed simply to demonstrate the concept. While they are useful in working on cases, substantial improvements in both the fonts and the tools will both be helpful in bringing this technology to more widespread use. In addition, the creation of fonts that are directly usable within browsers, terminal windows, document editors, and throughout the forensic process and tools, will substantially improve the outlook, both for accurate examination, and for improvements in the presentation of digital forensic evidence. The creation of forensic fonts for other character sets will also be helpful. For example, the implementation of EBCDIC and SIXBIT fonts were a simple matter, and they are helpful in examining exchanges and stored information in those representations. Similarly, other common representations, such as Unicode, seven bit ASCII with parity, etc. would be useful, and more generally, the capacity to create fonts sets for a wide range of different situations with relatively little effort would be helpful so that examiners can create font sets on-the-fly for presentation of specific data for specific uses.

Database interpretation of other sorts would also be quite helpful, as would fonts for depicting data values such as integers, stored timestamps, and other similar representations of multibyte values. While some of these might be more prudently implemented in other ways, the ultimate underlying question of formats for accurate and precise presentation of digital forensic data remains a worthy challenge, and this effort with regard to forensic fonts is only the beginning.

References

[1] The American Standard Code for Information Interchange (ASCII), standard X3.4-1963, American Standards Association, June 17, 1963

[2] Extended Binary Coded Decimal Interchange Code (EBCDIC), for details see the Web site at URL: http://www-01.ibm.com/software/globalization/cdra/appendix_a.jsp

[3] Rose v. Albritton, Superior Court of the County of San Francisco, Case No.: FDV-09-806677, July 14, 2009

[4] Daubert v. Merrell Dow Pharmaceuticals, Inc., 509 U.S. 579, 125 L. Ed. 2d 469, 113 S. Ct. 2786 (1993) and subsequent rulings dominate in US Federal cases. Frye (Frye v. United States, 293 F 1013 D.C. Cir, 1923) may apply in many states for non-Federal cases.