Fonts & Encoding

—

A Never Ending Problem?

Volker RW Schaa

GSI Helmholtzzentrum für
Schwerionenforschung GmbH
Darmstadt, Germany

JACoW Team Meeting
1 History of fonts and encodings
   - In pre-computer time
   - What changed in the computer era?
   - New standards
   - Problem of interpretation

2 Fonts in print (encoding and subsetting)
   - Encoding and naming
   - Font character maps
   - Font problems part 1 – Font not embedded
   - Font problems part 2 – Font embedded but characters missing

3 Conclusion
   - What can we do?
   - Thanks
In pre-computer times fonts never were a problem

Characters were local and known
In pre-computer times fonts never were a problem

Characters were local and known

In pre-computer time
- What changed in the computer era?
- New standards
- Problem of interpretation
In pre-computer times fonts never were a problem

Characters were local and known

You needed foreign characters — you took them
In pre-computer times fonts never were a problem

Characters were local and known

You needed foreign characters — you took them
In pre-computer times fonts never were a problem.

- **Characters were local and known**
  - In pre-computer times, fonts were not a problem.
  - Characters were local and known.

- **You needed foreign characters — you took them**
  - When you needed foreign characters, you took them.
In pre-computer time fonts never were a problem

Characters were placed in the typesetter’s tray
In pre-computer times fonts never were a problem.

Characters were placed in the typesetter’s tray by frequency of occurrence.
In pre-computer times fonts never were a problem

Characters were placed in the typesetter’s tray by frequency of occurrence

German, Swiss and other West-European languages
In pre-computer times fonts never were a problem

Characters were placed in the typesetter’s tray by frequency of occurrence.

German, Swiss and other West-European languages used the following distribution.
Computers introduced problems: Early days

- Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)
Computers introduced problems: Early days

- Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)

![USASCII code chart](chart.png)
Computers introduced problems: Early days

- Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)
- Introduction of EBCDIC by IBM in 1963 (8 Bit, 94 printable characters, 65 control codes, 3 invisible spaces, soffthyphen)
History of fonts and encodings

Font in print (encoding and subsetting)

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NUL</td>
<td>SOH</td>
<td>STX</td>
<td>ETX</td>
<td>FS</td>
<td>GS</td>
<td>RS</td>
<td>US</td>
<td>DOM</td>
</tr>
<tr>
<td>1</td>
<td>STR</td>
<td>DC1</td>
<td>DC2</td>
<td>DC3</td>
<td>QM</td>
<td>SC</td>
<td>CR</td>
<td>SO</td>
<td>SI</td>
</tr>
<tr>
<td>2</td>
<td>CAN</td>
<td>EM</td>
<td>ETB</td>
<td>ESC</td>
<td>DEL</td>
<td>HT</td>
<td>RX</td>
<td>FS</td>
<td>SP</td>
</tr>
<tr>
<td>3</td>
<td>NUL</td>
<td>DCS</td>
<td>PAD</td>
<td>SYN</td>
<td>STS</td>
<td>CHC</td>
<td>MW</td>
<td>SPA</td>
<td>EOT</td>
</tr>
<tr>
<td>4</td>
<td>DCS</td>
<td>PAD</td>
<td>DCS</td>
<td>PAD</td>
<td>DCS</td>
<td>PAD</td>
<td>DCS</td>
<td>SUB</td>
<td>EOT</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>DLE</td>
<td>DC4</td>
<td>NUL</td>
<td>DC1</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
</tr>
<tr>
<td>6</td>
<td>DC3</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
</tr>
<tr>
<td>7</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
<td>DLE</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0001</td>
<td>0002</td>
<td>0003</td>
<td>0004</td>
<td>0005</td>
<td>0006</td>
<td>0007</td>
<td>0008</td>
</tr>
<tr>
<td>1</td>
<td>0009</td>
<td>000A</td>
<td>000B</td>
<td>000C</td>
<td>000D</td>
<td>000E</td>
<td>000F</td>
<td>0010</td>
<td>0011</td>
</tr>
<tr>
<td>2</td>
<td>0012</td>
<td>0013</td>
<td>0014</td>
<td>0015</td>
<td>0016</td>
<td>0017</td>
<td>0018</td>
<td>0019</td>
<td>001A</td>
</tr>
<tr>
<td>3</td>
<td>001B</td>
<td>001C</td>
<td>001D</td>
<td>001E</td>
<td>001F</td>
<td>0020</td>
<td>0021</td>
<td>0022</td>
<td>0023</td>
</tr>
<tr>
<td>4</td>
<td>0024</td>
<td>0025</td>
<td>0026</td>
<td>0027</td>
<td>0028</td>
<td>0029</td>
<td>002A</td>
<td>002B</td>
<td>002C</td>
</tr>
<tr>
<td>5</td>
<td>002D</td>
<td>002E</td>
<td>002F</td>
<td>0030</td>
<td>0031</td>
<td>0032</td>
<td>0033</td>
<td>0034</td>
<td>0035</td>
</tr>
<tr>
<td>6</td>
<td>0036</td>
<td>0037</td>
<td>0038</td>
<td>0039</td>
<td>003A</td>
<td>003B</td>
<td>003C</td>
<td>003D</td>
<td>003E</td>
</tr>
<tr>
<td>7</td>
<td>003F</td>
<td>0040</td>
<td>0041</td>
<td>0042</td>
<td>0043</td>
<td>0044</td>
<td>0045</td>
<td>0046</td>
<td>0047</td>
</tr>
</tbody>
</table>

In pre-computer time

Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)

In computer times:

Introduction of EBCDIC

Today, IBM claims to be an open-systems company, but IBM's

Both widely spread 'standards' were implemented for US-English

characters, 33

4 printable

(softyphen)
Computers introduced problems: Early days

- Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)
- Introduction of EBCDIC by IBM in 1963 (8 Bit, 94 printable characters, 65 control codes, 3 invisible spaces, soffthyphen)

From the Jargon book:
Computers introduced problems: Early days

- Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)
- Introduction of EBCDIC by IBM in 1963 (8 Bit, 94 printable characters, 65 control codes, 3 invisible spaces, softhyphen)

From the Jargon book:
Today, IBM claims to be an open-systems company, but IBM’s own description of the EBCDIC variants and how to convert between them is still internally classified top-secret, burn-before-reading.
Computers introduced problems: Early days

- Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)
- Introduction of EBCDIC by IBM in 1963 (8 Bit, 94 printable characters, 65 control codes, 3 invisible spaces, soft hyphen)

A popular joke:
Computers introduced problems: Early days

- Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)
- Introduction of EBCDIC by IBM in 1963 (8 Bit, 94 printable characters, 65 control codes, 3 invisible spaces, soft hyphen)

A popular joke:
Professor: "So the American government went to IBM to come up with an encryption standard, and they came up with—"

---

From the Jargon book:
Instead of get used to reading and writing a German, French, or Swedish, etc., programmer had to code points encoded different characters in different countries. It did not make any additional codes available, so the same characters missing, Canada used an own version which supported French. Even ASCII wasn't usable for British-English because the '£' was own description of the EBCDIC variants and how to convert.
Computers introduced problems: Early days

- Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)
- Introduction of EBCDIC by IBM in 1963 (8 Bit, 94 printable characters, 65 control codes, 3 invisible spaces, soft hyphen)

A popular joke:
Professor: "So the American government went to IBM to come up with an encryption standard, and they came up with—"
Student: "EBCDIC!"
Computers introduced problems: Early days

- Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)
- Introduction of EBCDIC by IBM in 1963 (8 Bit, 94 printable characters, 65 control codes, 3 invisible spaces, softyphen)
- Both widely spread ‘standards’ were implemented for US-English only
Computers introduced problems: Early days

- Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)
- Introduction of EBCDIC by IBM in 1963 (8 Bit, 94 printable characters, 65 control codes, 3 invisible spaces, soft hyphen)
- Both widely spread ‘standards’ were implemented for US-English only
- Even ASCII wasn’t usable for British-English because the ‘£’ was missing, Canada used an own version which supported French characters
Computers introduced problems: Early days

- Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)
- Introduction of EBCDIC by IBM in 1963 (8 Bit, 94 printable characters, 65 control codes, 3 invisible spaces, softhyphen)
- Both widely spread ‘standards’ were implemented for US-English only
- Even ASCII wasn’t usable for British-English because the ‘£’ was missing, Canada used an own version which supported French characters
- The later standard ISO/IEC 646, like ASCII, was a 7-bit character set. It did not make any additional codes available, so the same code points encoded different characters in different countries.
Computers introduced problems: Early days

- Introduction of ASCII in 1963 (7 Bit, 94 printable characters, 33 control codes, invisible space)
- Introduction of EBCDIC by IBM in 1963 (8 Bit, 94 printable characters, 65 control codes, 3 invisible spaces, soft hyphen)
- Both widely spread ‘standards’ were implemented for US-English only
- Even ASCII wasn’t usable for British-English because the ‘£’ was missing, Canada used an own version which supported French characters
- The later standard ISO/IEC 646, like ASCII, was a 7-bit character set. It did not make any additional codes available, so the same code points encoded different characters in different countries.
- A German, French, or Swedish, etc., programmer had to get used to reading and writing
  
  \[
  \text{ä} \quad \text{aÄiÜ}=\text{"Ö"n'}; \quad \text{ü}
  \]
  
  instead of
  
  \[
  \{ \text{a[i]=\text{"\n"; } } \}
  \]

From the Jargon book: "burn-before-reading. Between them is still internally classified top-secret, and they came up with—"
In April 1984 an eight-bit standard was adopted as ECMA-94 (later as ISO/IEC 8859). This derived from the character sets DEC-MCS and Mac OS Roman, developed as true extensions of ASCII. It leaves the original character-mapping intact, and adds additional character definitions after the first 128 (i.e., 7 bit) characters.
Computers introduced problems:

- In April 1984 an eight-bit standard was adopted as ECMA-94 (later as ISO/IEC 8859). This derived from the character sets DEC-MCS and Mac OS Roman, developed as true extensions of ASCII. It leaves the original character-mapping intact, and adds additional character definitions after the first 128 (i.e., 7 bit).

- With ECMA-94 (ISO/IEC 8859-1/-2/-3/-4) we are now able to write and encode the following European languages: Albanian, Catalan, Czech, Danish, Dutch, English, Estonian, Faeroese, Finnish, French, Galician, German, Greenlandic, Hungarian, Icelandic, Irish, Italian, Lappish, Latvian, Lithuanian, Maltese, Norwegian, Polish, Portuguese, Rumanian, Serbo-croatian, Slovak, Slovene, Spanish, Swedish, and Turkish (plus Afrikaans and Esperanto).

- However, we now have ways of defining different characters in character sets, but...
Computers introduced problems: Later times

- In April 1984 an eight-bit standard was adopted as ECMA-94 (later as ISO/IEC 8859). This derived from the character sets DEC-MCS and Mac OS Roman, developed as true extensions of ASCII. It leaves the original character-mapping intact, and adds additional character definitions after the first 128 (i.e., 7 bit) characters.

- With ECMA-94 (ISO/IEC 8859-1/-2/-3/-4) we are now able to write and encode the following European languages: Albanian, Catalan, Czech, Danish, Dutch, English, Estonian, Faeroese, Finnish, French, Galician, German, Greenlandic, Hungarian, Icelandic, Irish, Italian, Lappish, Latvian, Lithuanian, Maltese, Norwegian, Polish, Portuguese, Rumanian, Serbo-croatian, Slovak, Slovene, Spanish, Swedish, and Turkish (plus Afrikaans and Esperanto).
In April 1984 an eight-bit standard was adopted as ECMA-94 (later as ISO/IEC 8859). This derived from the character sets DEC-MCS and Mac OS Roman, developed as true extensions of ASCII. It leaves the original character-mapping intact, and adds additional character definitions after the first 128 (i.e., 7 bit) characters.

With ECMA-94 (ISO/IEC 8859-1/-2/-3/-4) we are now able to write and encode the following European languages: Albanian, Catalan, Czech, Danish, Dutch, English, Estonian, Faeroese, Finnish, French, Galician, German, Greenlandic, Hungarian, Icelandic, Irish, Italian, Lappish, Latvian, Lithuanian, Maltese, Norwegian, Polish, Portuguese, Rumanian, Serbo-croatian, Slovak, Slovene, Spanish, Swedish, and Turkish (plus Afrikaans and Esperanto).

So now we have ways of defining different characters in character sets, but...
New standards: Problems solved?

Now imagine, somebody types (or even better cuts/pastes) an ‘Abstract’ text into the submission form of SPMS for the conference where you are the editor.
New standards: Problems solved?

- Now imagine, somebody types (or even better cuts/pastes) an ‘Abstract’ text into the submission form of SPMS for the conference where you are the editor.
- Now we looking at the input text as the computer does. What do we (or the computer) see?

Perfect! But what does it mean?

We have (or the computer has) no idea where this might come from. We only know that we accept 8 Bit values. No idea whether it’s a Chinese scientist at CERN using his laptop or a French who is sitting at KEK, or . . .

No knowledge about used character set or character encoding. If the computer switches to a different interpretation of the input, it might look like this.

It helps a bit, but we still have no idea about the characters not being ASCII (or in the range of 32. . . 127 or x'20'...'7e') . . .
In pre-computer time
What changed in the computer era?
New standards
Problem of interpretation

New standards: Problems solved?

Now imagine, somebody types (or even better cuts/pastes) an
‘Abstract’ text into the submission form of SPMS for the
conference where you are the editor.

Now we looking at the input text as the computer does. What do
we (or the computer) see?

Perfect! But what does it mean?

We have (or the computer has) no idea where this might come
from. We only know that we accept 8 Bit values. No idea whether
it’s a Chinese scientist at CERN using his laptop or a French
who is sitting at KEK, or . . .

No knowledge about used character set or character encoding.
If the computer switches to a different interpretation of the input,
it might look like this.

It helps a bit, but we still have no idea about the characters not
being ASCII (or in the range of 32…7e). . .
Now imagine, somebody types (or even better cuts/pastes) an ‘Abstract’ text into the submission form of SPMS for the conference where you are the editor.

Now we looking at the input text as the computer does. What do we (or the computer) see?

Perfect! But what does it mean?
New standards: Problems solved?

- Now imagine, somebody types (or even better cuts/pastes) an ‘Abstract’ text into the submission form of SPMS for the conference where you are the editor.
- Now we looking at the input text as the computer does. What do we (or the computer) see?
- Perfect! But what does it mean?
- We have (or the computer has) no idea where this might come from. We only know that we accept 8 Bit values. No idea whether it’s a Chinese scientist at CERN using his laptop or a French who is sitting at KEK, or . . .
New standards: Problems solved?

- Now imagine, somebody types (or even better cuts/pastes) an ‘Abstract’ text into the submission form of SPMS for the conference where you are the editor.
- Now we looking at the input text as the computer does. What do we (or the computer) see?
- Perfect! But what does it mean?
- We have (or the computer has) no idea where this might come from. We only know that we accept 8 Bit values. No idea whether it’s a Chinese scientist at CERN using his laptop or a French who is sitting at KEK, or . . .
- No knowledge about used character set or character encoding.
### New standards: Problems solved?

- Now imagine, somebody types (or even better cuts/pastes) an ‘Abstract’ text into the submission form of SPMS for the conference where you are the editor.
- Now we looking at the input text as the computer does. What do we (or the computer) see?
- Perfect! But what does it mean?
- We have (or the computer has) no idea where this might come from. We only know that we accept 8 Bit values. No idea whether it’s a Chinese scientist at CERN using his laptop or a French who is sitting at KEK, or . . .
- No knowledge about used **character set** or **character encoding**.
- If the computer switches to a different interpretation of the input, it might look like this.
In pre-computer time
What changed in the computer era?
New standards
Problem of interpretation

New standards: Problems solved?

now imagine somebody types (or even better cuts/pastes) an 'abstract' text into the submission form of SPMS for the conference where you are the editor.

now we looking at the input text as the computer does. what do we (or the computer) see?

perfect! but what does it mean?

we have (or the computer has) no idea where this might come from. we only know that we accept 8 bit values. no idea whether it's a chinese scientist at CERN using his laptop or a french who is sitting at KEK, or . . .

no knowledge about used character set or character encoding.

if the computer switches to a different interpretation of the input, it might look like this.

It helps a bit, but we still have no idea about the characters not being ASCII (or in the range of 32...127 or \x'20'...'7e'). . . .
Now imagine, somebody types (or even better cuts/pastes) an ‘Abstract’ text into the submission form of SPMS for the conference where you are the editor.

Now we looking at the input text as the computer does. What do we (or the computer) see?

Perfect! But what does it mean?

We have (or the computer has) no idea where this might come from. We only know that we accept 8 Bit values. No idea whether it’s a Chinese scientist at CERN using his laptop or a French who is sitting at KEK, or . . .

No knowledge about used character set or character encoding.

If the computer switches to a different interpretation of the input, it might look like this.

It helps a bit, but we still have no idea about the characters not being ASCII (or in the range of 32…127 or x’20’…’7e’) . . .
Let's try to uncover the secret

- To get an idea about what it could mean we switch to West-European languages and select ISO8859-1:
Let’s try to uncover the secret

Let’s try to uncover the secret

Volker RW Schaa
Fonts & Encoding
Let’s try to uncover the secret

- To get an idea about what it could mean we switch to West-European languages and select ISO8859-1:

  ❌ Not a West-European language!
Let’s try to uncover the secret

- To get an idea about what it could mean we switch to the Greek language and select ISO8859-7:
Let's try to uncover the secret

ΠΟΔΑΣΤΣΡ ἸΕΝΜΡ ΣΕΜΨΘΟΪ ΞΑΪΞΑΗΞΙΡ
ΟΤ ΣΟΒΣΤΧΕΞΕΙΛΑ σζοمواد

υήαστλι: 24 ΗΑ, 732 ΗΑ, 1290 ΗΑ, 1361 ΗΑ
γέςα: 8000 ΥΥΜΕΚ ἸΑ ἩΑ (80 ΥΥΜΕΚ/ΣΟΤΛΑ) τοζη

υήαστλι χωδεμεζω, σχιδετεμψτχα υζζο πομυήεζω.
ζαζπομοφεζε: υμψρεοχελαρ οβμαζτζ, ἱζἰεζελικ ἄκοζ.
δολυνεζω πο ἱαπζοζυ.
λοεταλτζ: +7 (916) 99ώ-Ϊ6-61, 8052511(a)gmail.com
ναλζιν
Let’s try to uncover the secret

- To get an idea about what it could mean we switch to the Greek language and select ISO8859-7:

❌ Doesn’t look Greek to me!
Let's try to uncover the secret

- To get an idea about what it could mean we switch to the Baltic languages and select ISO8859-13:
Let’s try to uncover the secret

ŠNĻĀIÂDÔÔN ŠÂKĢN ÔÂGUČĽS ÎÎŚĪŢÂIĒN
Â ŌLÂÂDÔ×ÂĪĪĒÈĪ ŌŅĻŽĪL

ēŽIÔÔÉÈ: 24 Ėī, 732 Ėī, 1290 Ėī, 1361 Ėī
ĒÂĪĪ: 8000 ŅÕÂÂGÂŽ ŮÌ Ėī (80 ŅÕÂÂGÂŽ/ŌLÕÈĪ) Ūłňē

ēŽIÔÔÉÈ ×ĻÂÂGÂĪL, Ő×ĒÂÂÅÂĜŲÕÔ×Î Őęņò ŠĽÂÂÔÂĪL.
ŅĪŌŚĽÂÂ ÔIĒÂ: ÔGUŅIĻ×ÔĒĪŅ LÂÇIÔOŲ, ĖIŚÂIÔÈÈŽ ŇÎŽĪ.".

ÂLĒÂÔÂÌÔ ÔÌ ŠÎŠŅĪÔÔ.
ĒLĪĪÎÊÔÔL: +7 (916) 992-66-61, 8052511(@)gmail.com
KIĒÕÈK
Let’s try to uncover the secret

- To get an idea about what it could mean we switch to the Baltic languages and select ISO8859-13:

  Not a Baltic language!
Let’s try to uncover the secret

To get an idea about what it could mean, let’s try the Thai language and select ISO8859-11:
Let’s try to uncover the secret

To get an idea about what it could mean, let’s try

Z

the Thai language and select ISO8859-11:

Looks funny but perhaps not Thai! (See question marks in black diamonds?)

Volker RW Schaa

Let’s try to uncover the secret

To get an idea about what it could mean, let’s try

Z

the Thai language and select ISO8859-11:

Looks funny but perhaps not Thai! (See question marks in black diamonds?)

Volker RW Schaa
Let's try to uncover the secret

To get an idea about what it could mean, let's try the Thai language and select ISO8859-11:

Looks funny but perhaps not Thai! (See question marks in black diamonds?)
Let's try to uncover the secret

- To get an idea about what it could mean we switch to the Armenian language and select ISO8859-16:
Let’s try to uncover the secret

To get an idea about what it could mean we switch to the Armenian language and select ISO8859-16:

Looks good, but is it Armenian?

Volker RW Schaa
Fonts & Encoding
Let’s try to uncover the secret

- To get an idea about what it could mean we switch to the Armenian language and select ISO8859-16:

Looks good, but is it Armenian?
Let’s try to uncover the secret

- To get an idea about what it could mean, let’s stay for a moment in Europe and try Romanian and select MAC-Romanian:
In pre-computer time
What changed in the computer era?
New standards
Problem of interpretation

Let's try to uncover the secret

?“œf j≈‘”– /≈ Œ– “≈ Œy”œ/Œ j/Œ j Œ≈Œ–œ’ “œN” œ≈ŒE…À j ÙÚÔ ŒÔ

ɉTj ”´À…: 24 ≈ j, 732 ≈ j, 1290 ≈ j, 1361 ≈ j
”≈Œ j: 8000 ”’ÑÃ≈ /j ≈ j (80 ”’ÑÃ≈ /”œ’À j) ÙÔÚÁ

ɉTj ”´À… φy≈ Ã≈ŒY, ”φ…f≈ “≈ Œy”œ/ 1ÉÕÔ –œÃ’ Œ≈ŒY.
Új ”œÃœ≈≈Œ…≈: 1Ãy–Œœ/”À j–œÑÃ j”œj “y, ÈŒ∥≈Œ”À… “j œŒ.
%œÀ’Ø≈Œ’Y –œ /j–“œ”’.
êœŒjÀ’Y: +7 (9I6) 99 ç – ’6-61, 8052511(a)gmail.com
ÌjÀ”…Ø

Volker RW Schaa  Fonts & Encoding
Let’s try to uncover the secret

- To get an idea about what it could mean, let’s stay for a moment in Europe and try Romanian and select MAC-Romanian:

  ✗ Surely not Romanian (even on a MAC)
Let's try to uncover the secret

- To get an idea about what it could mean, let's try setting the text in Arabic and switch to ISO8859-6:
Let’s try to uncover the secret

زدؤء إشسر غإحجر سإجظبدغ غء غغء إخة ر

دش سدآشسطإخخث ء؟؟؟؟

ءششتة: ٢٤ ا، ٧٩٢ ا، ١٣٩٠ ا، ١٣٨١ ا

كإخء: ٠٠٠٨ زسآجإت غء ا (٠٨ زسآجات/سدوحة) ء؟؟؟

ءششتة طؤؤإجإخع، سطة ؤؤشإجظسطء ء؟ ء؟ ء؟ ء؟ ذدجس ء؟ إخع.

ءششتة طؤؤإجإخع، سطة ؤؤشإجظسطء ء؟ ء؟ ء؟ ء؟ ذدجس ء؟ إخع.


ه6-61, 8052511(a)gmail.com

Volker RW Schaa  Fonts & Encoding
Let's try to uncover the secret

- To get an idea about what it could mean, let's try
  - Arabic and switch to ISO8859-6:
  - Too many black diamonds! Not Arabic!
Let’s try to uncover the secret

To get an idea about what it could mean, let us assume it’s our Chinese colleague at CERN,

so switch to Chinese-Simplified (ISO-2022-CN):
Let's try to uncover the secret

Still many black diamonds! Not simple Chinese!

To get an idea about what it could mean, let us assume it's our Chinese colleague at CERN, Z so switch to Chinese-Simplified (ISO-2022-CN):

磔劣运 24 椒, 732 椒, 1290 椞, 1361 椞
问瘟: 8000 艺绿攀 谰 椞 (80 艺绿攀/西运) 鼩蜱
磔劣运 踌呐膛鸨, 幼赡旁盘阘宰 蹿蝮 邢陶介<UnityEngine:Arial,Normal> Brittatical
鹧有咸现盼膳: 踌匿蜗子肆 下塘秏, 榄谂牦松 伊氏.

湎苏团卧 邢 谰幸嫌.
臌卧了再: +7 (916) 99-1, 8052511(a)gmail.com
碳缴赏
Let’s try to uncover the secret

- To get an idea about what it could mean, let us assume it’s our Chinese colleague at CERN,

  so switch to Chinese-Simplified (ISO-2022-CN):

  **Still many black diamonds! Not simple Chinese!**
Let's try to uncover the secret

To get an idea about what it could mean, what about traditional Chinese, let's try Chinese-Traditional:
Let's try to uncover the secret
Let's try to uncover the secret

- To get an idea about what it could mean, what about traditional Chinese, let's try Chinese-Traditional:

  🔄 Different but still wrong!
Let's try to uncover the secret

- To get an idea about what it could mean, what about Farsi? We use Mac-Farsi:
Let’s try to uncover the secret

زدؤء إشسر فإحجر سإحظبدغ خء خء إخرة رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء رةء الر

چ•ء ششتة: ۲۴ ا، ۲۳ ا، ۱۳۹۰ ا، ۱۳۶۱ ا،
کاخ: ۸۰۰۰ زصاباجات غء اة (۸۰ زصاباجات/شدشة)
چ•ء ششتة طعوا إجاخ، سطة وشاءش بششة، حن• ب ذدجم إجاخ.
۸۰ سددمضاخة إ: چجطرخدششة، ر داجء سشش، یخغازشسشات زءتدح.
لدتضاهاخشع ذذ غء ذذدسن.
۹/۶-۶۱، ۶۵۰۰۸۸۶۶ (I• دخء تخشع: +۹۸ ز-۶۱-۶۵۰۰۸۸۶۶ (a)gmail.com

Volker RW Schaa  Fonts & Encoding
Let's try to uncover the secret

- To get an idea about what it could mean, what about Farsi? We use Mac-Farsi:

  Still no idea what it means!
Let’s try to uncover the secret

- To get an idea about what it could mean, what about the fifth most spoken language?

Hindi? We switch to Mac-Hindi:
Let's try to uncover the secret

Let's try to uncover the secret to get an idea about what it could mean, what about the fifth most spoken language Hindi? We switch to Mac-Hindi:

Still black diamonds!

Let's try to uncover the secret

Volker RW Schaa

Font & Encoding
Let’s try to uncover the secret

- To get an idea about what it could mean, what about the fifth most spoken language
  
  Hindi? We switch to Mac-Hindi:

  ❌ Still black diamonds!
Let’s try to uncover the secret

- To get an idea about what it could mean, what about our Japanese colleagues?
- Let’s try Japanese in Shift_JIS encoding:
Let’s try to uncover the secret

Anybody can read this?

Let’s try Japanese in Shift_JIS encoding:

?ちもやひ：２４ すず、７３２ すず、１２９０ すず、１３６１ すず

蝯形：８０００ もえつふとな べち すず（８０ もえつふとな/もまやひ）??

?ちもやひ ラルトナフナホル、モラホトナフナフリモラヒチ ?? ミマフェゴナホル。

?もミマフマヨナホノナ：？リムホマラモヒチム マツフチモワリ、鰤レナホモヒホ メチハマホ。

?まヒュヘナホホアル キマ レチミメマエ。

・ホマチヒアル：+7 (916) 99?-・-61, 8052511(a)gmail.com

喜ヒモノ
Let’s try to uncover the secret

- To get an idea about what it could mean, what about our Japanese colleagues?

Let’s try Japanese in Shift_JIS encoding:

- Anybody can read this?
Let's try to uncover the secret

- To get an idea about what it could mean, we now try Unicode
  Let's switch to UTF-16:
Let's try to uncover the secret

- To get an idea about what it could mean, we now try Unicode

Let's switch to UTF-16:
Let’s try to uncover the secret

- To get an idea about what it could mean, we now try Unicode

- Let’s switch to UTF-16:

- Too many question marks and daggers!
Let’s try to uncover the secret

OK, we left out a language which is the eighth most spoken one: Russian! Let’s switch to Cyrillic in ISO8859-5 encoding:
Let’s try to uncover the secret

Продается земля сельхоз назначения
от собственника СРОЧНО

Участки: 24 га, 732 га, 1290 га, 1361 га
Цена: 8000 рублей за га (80 рублей/сотка) ТОРГ

Участки выделены, свидетельства УФРС получены.
Расположение: Ульяновская область, Инженский район.
Документы по запросу.
Контакты: +7 (916) 99ч-36-61, 8052511(a)gmail.com
Максим
Let’s try to uncover the secret

- OK, we left out a language which is the eighth most spoken one: Russian! Let’s switch to Cyrillic in ISO8859-5 encoding:
- That look nice, even the line feeds are correct. Makzim??
Let’s try to uncover the secret

- So in the end it’s not an ‘Abstract’ but some SPAM message.
- And this is the English translation (OK, mostly English):
The ground сельхоз purposes is on sale
From the proprietor URGENTLY
Sites: 24 hectares, 732 hectares, 1290 hectares, 1361 hectares
The price: 8000 roubles for hectares (80 roubles/сотка) the TENDER
Sites are allocated, certificates УФПС are received.
Arrangement: the Ulyanovsk area, area Inzensky.
Documents by inquiry.
Contacts: 7 (916) 994-36-61, 8052511 (a) gmail.com
Maxim
We now have understood that **Encoding** is important as it points to the right character: `x‘a4’ + ISO8859-15 ⇒ €`
Encoding

- We now have understood that **Encoding** is important as it points to the right character:  `x‘a4’ + ISO8859-15 ⇒ €
- If we would have used:  `x‘a4’ + ISO8859-14 ⇒ Ć
Encoding

- We now have understood that **Encoding** is important as it points to the right character: \text{x`a4`} + \text{IS08859-15} \Rightarrow \text{€}
- If we would have used: \text{x`a4`} + \text{IS08859-14} \Rightarrow \text{Ć}
- But how are **Symbol** fonts encoded? Is there a unique way to define an encoding for them?
We now have understood that **Encoding** is important as it points to the right character: \( \text{x‘a4’ + ISO8859-15} \Rightarrow \€ \)

If we would have used: \( \text{x‘a4’ + ISO8859-14} \Rightarrow Ć \)

But how are **Symbol** fonts encoded? Is there a unique way to define an encoding for them?

💡 Short answer: before Unicode some had names (PostScript Level 3 named \( \approx 180 \) including Greek letters but nothing specific for Math Symbols)
Encoding

- We now have understood that **Encoding** is important as it points to the right character: \x{a4} + ISO8859-15 ⇒ €
- If we would have used: \x{a4} + ISO8859-14 ⇒ Ć
- But how are **Symbol** fonts encoded? Is there a unique way to define an encoding for them?

⚠️ Short answer: before Unicode some had names (PostScript Level 3 named ≈180 including Greek letters but nothing specific for Math Symbols)

⚙️ In **Unicode** it look like this:
Encoding

We now have understood that Encoding is important as it points to the right character: \( \text{x‘a4'} + \text{ISO8859-15} \Rightarrow \€ \)

If we would have used: \( \text{x‘a4'} + \text{ISO8859-14} \Rightarrow ˙C \)

But how are Symbol fonts encoded? Is there a unique way to do it?

### Multiplication and division sign operators

<table>
<thead>
<tr>
<th>Code</th>
<th>Unicode</th>
</tr>
</thead>
<tbody>
<tr>
<td>2A2F</td>
<td>VECTOR OR CROSS PRODUCT</td>
</tr>
<tr>
<td></td>
<td>( \rightarrow 00D7 \times ) multiplication sign</td>
</tr>
<tr>
<td>2A30</td>
<td>MULTIPLICATION SIGN WITH DOT ABOVE</td>
</tr>
<tr>
<td>2A31</td>
<td>MULTIPLICATION SIGN WITH UNDERBAR</td>
</tr>
<tr>
<td>2A32</td>
<td>SEMIDIRECT PRODUCT WITH BOTTOM CLOSED</td>
</tr>
<tr>
<td>2A33</td>
<td>SMASH PRODUCT</td>
</tr>
<tr>
<td>2A34</td>
<td>MULTIPLICATION SIGN IN LEFT HALF CIRCLE</td>
</tr>
<tr>
<td>2A35</td>
<td>MULTIPLICATION SIGN IN RIGHT HALF CIRCLE</td>
</tr>
<tr>
<td>2A36</td>
<td>CIRCLED MULTIPLICATION SIGN WITH CIRCUMFLEX ACCENT</td>
</tr>
<tr>
<td>2A37</td>
<td>MULTIPLICATION SIGN IN DOUBLE CIRCLE</td>
</tr>
<tr>
<td>2A38</td>
<td>CIRCLED DIVISION SIGN</td>
</tr>
</tbody>
</table>
Mapping

- A typical font (Type 1, TrueType and OpenType) contains a map with character names and its location in the font.
Mapping

- A typical font (Type 1, TrueType and OpenType) contains a map with character names and its location in the font.
- Nowadays fonts have far more than the possible 192 characters (standard is 250, but there are fonts with more 2 000 characters).
Mapping

- A typical font (Type 1, TrueType and OpenType) contains a map with character names and its location in the font.
- Nowadays fonts have far more than the possible 192 characters (standard is 250, but there are fonts with more 2,000 characters).
- The PostScript (printer) driver looks up the requested font and
  - accesses the character map table of this fonts,
  - it scans the text to typeset and finds out about the requested Encoding and
  - all different characters that appear in the text.
Mapping

- A typical font (Type 1, TrueType and OpenType) contains a map with character names and its location in the font.

- Nowadays fonts have far more than the possible 192 characters (standard is 250, but there are fonts with more 2,000 characters).

- The PostScript (printer) driver looks up the requested font and
  - accesses the character map table of this fonts,
  - it scans the text to typeset and finds out about the requested Encoding and
  - all different characters that appear in the text.

- Now it copies the needed Encoding vector and, depending on the parameters, copies either the whole set of characters, just the needed ones or nothing (Embedded, Embedded Subset, nada).
Mapping

- A typical font (Type 1, TrueType and OpenType) contains a map with character names and its location in the font.
- Nowadays fonts have far more than the possible 192 characters (standard is 250, but there are fonts with more 2 000 characters).
- The PostScript (printer) driver looks up the requested font and accesses the character map table of this fonts, it scans the text to typeset and finds out about the requested Encoding and all different characters that appear in the text.
- Now it copies the needed Encoding vector and, depending on the parameters, copies either the whole set of characters, just the needed ones or nothing (Embedded, Embedded Subset, nada).
- If it finds characters which do not fit into one Encoding vector, the driver may copy a second map and the corresponding characters.
Mapping and character copying

- How does a Encoding map looks like?
- An example of a Custom map which encodes more than the standard ISOLatin1Encoding by using some of the code point between x‘00’ and x‘1f’.
Mapping and character copying

```
/Base1Encoding [
  % 0x00 (encoded characters from Adobe Standard not in Windows 3.1)
  /.notdef /dotaccent /fi /fl /fraction /hungarumlaut /lslash /lslash
  /ogonek /ring /.notdef /breve /minus /.notdef /Zcaron /zcaron
  % 0x10
  /caron /dotlessi /dotlessj /ff /ffi /ffl /.notdef /.notdef
  /.notdef /.notdef /.notdef /.notdef /.notdef /.notdef /grave /quotesingle
  % 0x20 (ASCII begins)
  /space /exclam /quotedbl /numbersign /dollar /percent /ampersand /quoteright
  /parenleft /parenright /asterisk /plus /comma /hyphen /period /slash
  % 0x30
  /zero /one /two /three /four /five /six /seven
  /eight /nine /colon /semicolon /less /equal /greater /question
  % 0x40
  /at /A /B /C /D /E /F /G /H /I /J /K /L /M /N /O
  % 0x50
  /P /Q /R /S /T /U /V /W
  /X /Y /Z /bracketleft /backslash /bracketright /asciicircum /underscore
  % 0x60
  /quoteleft /a /b /c /d /e /f /g /h /i /j /k /l /m /n /o
  % 0x70
  /p /q /r /s /t /u /v /w
  /x /y /z /bracketleft /bar /braceright /asciitilde /.notdef
  % 0x80
  /Euro /.notdef /quotesinglebase /florin
  /quotedblbase /ellipsis /dagger /daggerdbl
  /circumflex /perthousand /Scaron /guilsingleleft
  /OE /.notdef /.notdef /.notdef
```
Mapping and character copying

- How does a **Encoding** map looks like?
  An example of a **Custom** map which encodes more than the standard **ISOLatin1Encoding** by using some of the code point between x‘00’ and x‘1f’.
- The code which makes up a font **Subset** may look like this:
Mapping and character copying

```
/TimesNewRoman findfont /Encoding get
dup 72 /H put
dup 101 /e put
dup 108 /l put
dup 111 /o put
dup 32 /space put
dup 87 /W put
dup 114 /r put
dup 100 /d put
dup 33 /exclam put
pop
/N15 11.2898 Tf
(Hello World!)
```
Mapping and character copying

- How does a **Encoding** map looks like?
  An example of a **Custom** map which encodes more than the standard **ISOLatin1Encoding** by using some of the code point between x‘00’ and x‘1f’.

- The code which makes up a font **Subset** may look like this:

- So this all looks fine, but why do we face problems so often?
Mapping and character copying

- How does a **Encoding** map looks like?
  An example of a **Custom** map which encodes more than the standard **ISOLatin1Encoding** by using some of the code point between x‘00’ and x‘1f’.

- The code which makes up a font **Subset** may look like this:

- So this all looks fine, but why do we face problems so often?

- Let us look into the **Document Properties** of a typical PDF file:
Mapping and character copying

How does a Encoding map looks like?

An example of a Custom map which encodes more than the standard ISOLatin1Encoding by using some of the code point between x'00' and x'1f'.

The code which makes up a font Subset may look like this:

So this all looks fine, but why do we face problems so often?

Let us look into the Document Properties of a typical PDF file:

We find 3 different Encodings (Ansi, Identity-H, and Roman) for one font, the 2 pairs of character sub-maps (Identity-H, Ansi) are surely not the same, and there are 3 different font format embedded: Type 1, TrueType, and TrueType (CID).

Positive: all fonts are embedded as subsets!

But: we surely will have problems to change or edit such a file with Pitstop or Acrobat!
Mapping and character copying

- How does a **Encoding** map looks like?
  An example of a **Custom** map which encodes more than the standard **ISOLatin1Encoding** by using some of the code point between x‘00’ and x‘1f’.

- The code which makes up a font **Subset** may look like this:

- **So this all looks fine, but why do we face problems so often?**

- Let us look into the **Document Properties** of a typical PDF file:

- **We find**
  - 3 different **Encodings** (Ansi, Identity-H, and Roman) for one font,
  - the 2 pairs of character sub-maps (**Identity-H, Ansi**) are surely not the same,
  - and there are 3 different font format embedded: Type 1, TrueType, and TrueType (CID)
Mapping and character copying

- How does a Encoding map looks like? An example of a Custom map which encodes more than the standard ISOLatin1Encoding by using some of the code point between x`00` and x`1f`.

- The code which makes up a font Subset may look like this:

- So this all looks fine, but why do we face problems so often?

- Let us look into the Document Properties of a typical PDF file:

- We find
  - 3 different Encodings (Ansi, Identity-H, and Roman) for one font,
  - the 2 pairs of character sub-maps (Identity-H, Ansi) are surely not the same,
  - and there are 3 different font format embedded: Type 1, TrueType, and TrueType (CID)

- Positive: all fonts are embedded as subsets!

- But: we surely will have problems to change or edit such a file with Pitstop or Acrobat!
Why is a font not embedded?

Just for the statistics: in ≈35% of the PDF files uploaded for a JACoW conference one or more fonts are missing.

- A font was requested which is one of the 15 base fonts and the switch **always embed fonts** had not been set in the PostScript produced by the author.
Why is a font not embedded?
Just for the statistics: in $\approx 35\%$ of the PDF files uploaded for a JACoW conference one or more fonts are missing.

- A font was requested which is one of the 15 base fonts and the switch **always embed fonts** had not been set in the PostScript produced by the author.

- In the PostScript driver setup the option for TrueType fonts was set as “Substitute with Device Font".
A font was requested in the PostScript file which is not installed on the editor's computer, therefore a substitute font is used for display and PDF generation, but the font will not be included in the final PDF.

A TrueType, CID or OpenType font was used that contained the flag `/FSType <x> def` with an x-value, 0 (i.e. 2 [restricted license embedding], 4 [embedding allowed, but not editable], 256 [no subset embedding], 512 [Bitmap embedding only]).
Why is a font not embedded?
Just for the statistics: in ≈35% of the PDF files uploaded for a JACoW conference one or more fonts are missing.

- A font was requested which is one of the 15 base fonts and the switch `always embed fonts` had not been set in the PostScript produced by the author.
- In the PostScript driver setup the option for TrueType fonts was set as "Substitute with Device Font".
- A font was requested in the PostScript file which is not installed on the editor's computer, therefore a substitute font is used for display and PDF generation, but the font will not be included in the final PDF.
Why is a font not embedded?

Just for the statistics: in ≈35% of the PDF files uploaded for a JACoW conference one or more fonts are missing.

- A font was requested which is one of the 15 base fonts and the switch always embed fonts had not been set in the PostScript produced by the author.

- In the PostScript driver setup the option for TrueType fonts was set as “Substitute with Device Font”.

- A font was requested in the PostScript file which is not installed on the editor’s computer, therefore a substitute font is used for display and PDF generation, but the font will not be included in the final PDF.

- A TrueType, CID or OpenType font was used that contained the flag `/FSType <x> def` with an x-value ≠ 0 (i.e. 2 [restricted license embedding], 4 [embedding allowed, but not editable], 256 [no subset embedding], 512 [Bitmap embedding only])
Why are characters missing (or shown as □)?

- A font was requested with a high version (newer release date containing more characters) than the font installed on the editor’s computer (therefore missing in display and/or print) or on the printer (missing in print).

[Like during PAC’09 on Mac computers]
Why are characters missing (or shown as □)?

- A font was requested with a high version (newer release date containing more characters) than the font installed on the editor’s computer (therefore missing in display and/or print) or on the printer (missing in print).

- **Missing Characters in PDF** in

  http://americanprinter.com/mag/printing_best_pdfs/

  On the Mac, as any user knows, fonts can be found just about anywhere and it’s very easy to have duplicate fonts. To make matters worse, OS X includes Apple’s new dfonts, which are simply the Mac version of the most commonly used fonts, like Helvetica and Times. When it comes to PDF creation, this variability can result in the wrong version of a font being embedded in a PDF file with the possible outcome of different spacing, kerning or even inclusion of a glyph that is entirely different from the one originally set in the page layout.

  [Like during PAC’09 on Mac computers]
Why are characters missing (or shown as □) cont.?

- In the PostScript driver setup the option for TrueType fonts was set as “Substitute with Device Font” but the printer device has a font with different layout or less characters.
What can we do for the editors to make life easier?

- Make sure when installing Word on PC or Mac that the newest fonts are installed and that there is only one version on the computer.
What can we do for the editors to make life easier?

- Make sure when installing Word on PC or Mac that the newest fonts are installed and that there is only one version on the computer.
- Make sure that the newest firmware is loaded on printers used in the editorial office.
What can we do for the editors to make life easier?

- Make sure when installing Word on PC or Mac that the newest fonts are installed and that there is only one version on the computer.
- Make sure that the newest firmware is loaded on printers used in the editorial office.
- The editors should check twice that fonts are embedded.
What can we do for the editors to make life easier?

- Make sure when installing Word on PC or Mac that the newest fonts are installed and that there is only one version on the computer.
- Make sure that the newest firmware is loaded on printers used in the editorial office.
- The editors should check twice that fonts are embedded.
- We should provide a set of fonts (on JACoW) which are not distributed with the OS any longer (i.e. original Times family) for installation on computers in the editorial office.
What can we do for the editors to make life easier?

- Make sure when installing Word on PC or Mac that the newest fonts are installed and that there is only one version on the computer.
- Make sure that the newest firmware is loaded on printers used in the editorial office.
- The editors should check twice that fonts are embedded.
- We should provide a set of fonts (on JACoW) which are not distributed with the OS any longer (i.e. original Times family) for installation on computers in the editorial office.
- Ease the configuration of Tools & Utilities
What can we do for the editors to make life easier?

- Make sure when installing Word on PC or Mac that the newest fonts are installed and that there is only one version on the computer.
- Make sure that the newest firmware is loaded on printers used in the editorial office.
- The editors should check twice that fonts are embedded.
- We should provide a set of fonts (on JACoW) which are not distributed with the OS any longer (i.e. original Times family) for installation on computers in the editorial office.
- Ease the configuration of Tools & Utilities

Listen to Raphael’s talk about a new step to ease configuration woes with a “Generic PostScript Printer for JACoW”!
Thanks for your attention!