

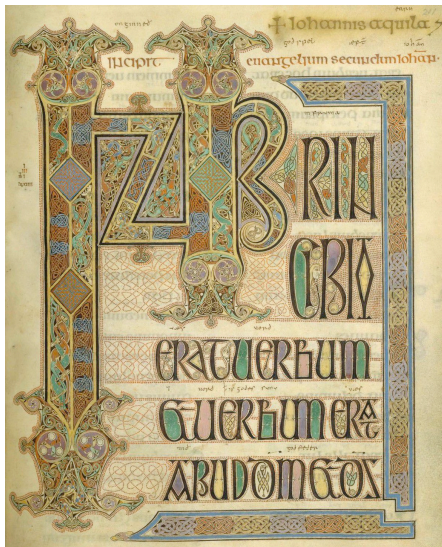
Fonts & Encoding

A Never Ending Problem?

Volker RW Schaa


GSI Helmholtzzentrum für
Schwerionenforschung GmbH
Darmstadt, Germany

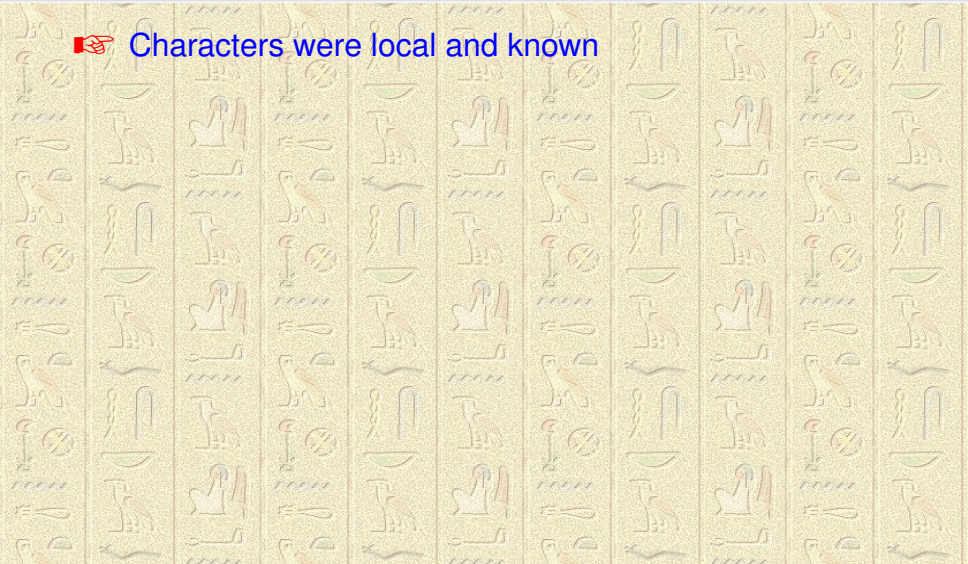
JACoW Team Meeting 2009
DESY@Hamburg, Germany
November, 2009



- 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 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

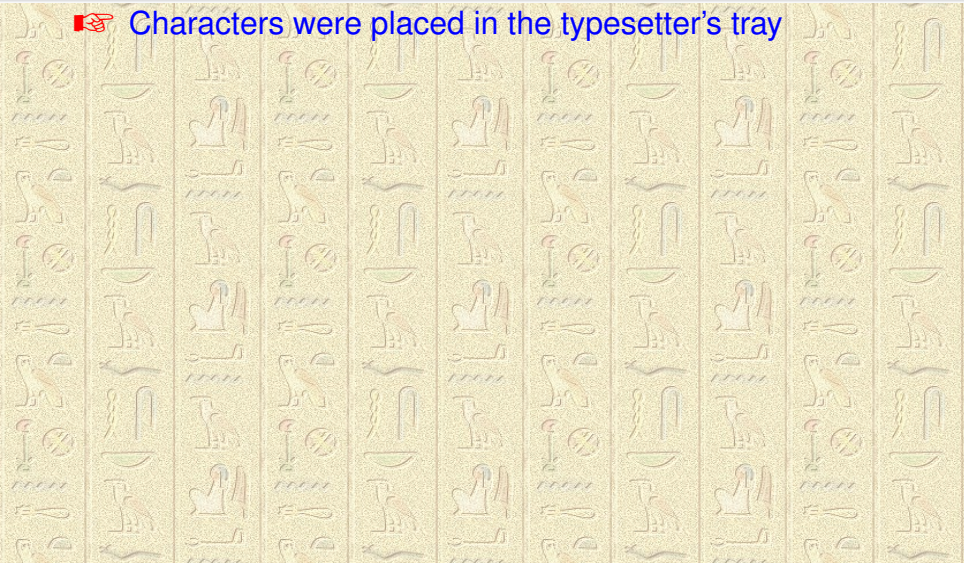


☞ You needed foreign characters — you took them



In pre-computer times 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

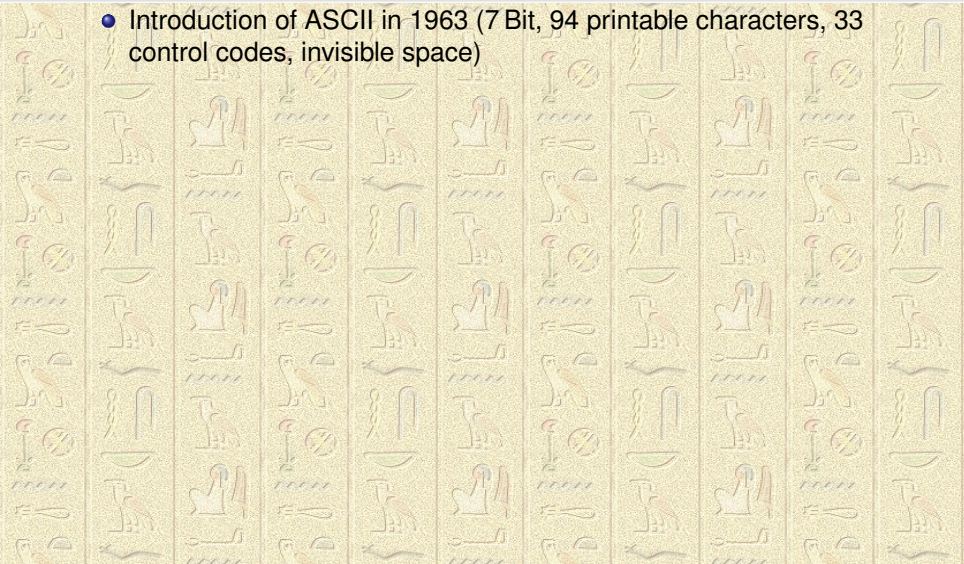


☞ German, Swiss and other West-European languages used the following distribution

A	B	C	D	E	F	G	H	I	K
L	M	N	O	P	Q	R	S	T	U
1	2	3	4	5	6	7	8	9	0
-	J	V	W	X	Y	Z	&		
à	â	á	Ä	ß		ä	ö	ü	«
»	„	“	+	§					
é	ê	è	ë			x	y	z	j
()	[]	!	?			v	w	-	:
;				t	u	r			
í	î	ï	ï	s				q	.
ó	ô	ò	Ö	h					
ú	û	ü	Ü	l	m		n	o	p
									,
									■
Æ	É	É	k	ck	c			fi	fl
ft									
œ	ç	ç	ch	b	a		e	d	f
									ff
									g
									■

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

					0	0	0	0	1	1	1	1
					0	0	0	1	0	0	1	1
Row	0	1	2	3	4	5	6	7				
0 0 0 0	0	NUL	DLE	SP	0	@	P	\	p			
0 0 0 1	1	SOH	DC1	!	1	A	Q	o	q			
0 0 1 0	2	STX	DC2	"	2	B	R	b	r			
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s			
0 1 0 0	4	EOT	DC4	\$	4	D	T	d	t			
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u			
0 1 1 0	6	ACK	SYN	^	6	F	V	f	v			
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w			
1 0 0 0	8	BS	CAN	(8	H	X	h	x			
1 0 0 1	9	HT	EM)	9	I	Y	i	y			
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z			
1 0 1 1	11	VT	ESC	+	;	K	[k	{			
1 1 0 0	12	FF	FS	,	<	L	\	l				
1 1 0 1	13	CR	GS	-	=	M]	m	}			
1 1 1 0	14	SO	RS	.	>	N	^	n	~			
1 1 1 1	15	SI	US	/	?	O	_	o	DEL			

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)

Conclusion

Computers into

- Intro
- con
- Intro
- chara

		UTF-8BCD1C															
		-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	NUL	SOH	STX	ETX	ST	HT	SSA	DEL	EPA	RI	SS2	VT	FF	CR	SO	SI	
	0000	0001	0002	0003	0009c	0009	0086	007f	0097	008d	008e	000c	000e	000d	000e	000f	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1-	DLE	DC1	DC2	DC3	ESC	LF	BS	ESA	CAN	EM	PU2	SS3	FS	GS	RS	US	
	0010	0011	0012	0013	009d	000a	0008	0087	0018	0019	0092	008f	001c	001d	001e	001f	
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
2-	PAD	HOP	BPH	NBH	IND	NEL	ETB	ESC	HTS	HTJ	VTS	PLD	PLU	ENQ	ACK	BEL	
	0080	0081	0082	0083	0084	0085	0017	001b	0088	0089	008a	008b	008c	0005	0006	0007	
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	
3-	DCS	PUL	SYN	STS	CCH	MW	SPA	EOT	SOS	SGCI	SCI	CSI	DC4	NAK	FM	SUB	
	0090	0091	0016	0093	0094	0095	0096	0004	0098	0099	009a	009b	0014	0015	009c	001a	
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	
4-	SP											.	<	(+		
	0020											002e	003c	0028	002b	007c	
	64											75	76	77	78	79	
5-	€											!	\$	*)	;	^
	0026											0021	0024	002a	0029	003b	005e
	80											90	91	92	93	94	95
6-	-	/										?	€	>	?		
	002d	002f										002c	0025	007e	003e	003f	
	96	97										107	108	109	110	111	
7-												~	:	#	@	'	"
												0060	003a	123	124	125	126
												121	122	123	124	125	127
8-		a	b	c	d	e	f	g	h	i							
		0061	0062	0063	0064	0065	0066	0067	0068	0069							
		129	130	131	132	133	134	135	136	137							
9-		j	k	l	m	n	o	p	q	r							
		006a	006b	006c	006d	006e	006f	0070	0071	0072							
		145	146	147	148	149	150	151	152	153							
A-		~	s	t	u	v	w	x	y	z				[
		007e	0073	0074	0075	0076	0077	0078	0079	007a				005b	173		
		161	162	163	164	165	166	167	168	169							
B-]			
														005d	189		
C-	{	A	B	C	D	E	F	G	H	I							
	007b	0041	0042	0043	0044	0045	0046	0047	0048	0049							
	192	193	194	195	196	197	198	199	200	201							
D-	}	J	K	L	M	N	O	P	Q	R							
	007d	004a	004b	004c	004d	004e	004f	0050	0051	0052							
	208	209	210	211	212	213	214	215	216	217							
E-	\	S	T	U	V	W	X	Y	Z								
	005c		0053	0054	0055	0056	0057	0058	0059	005a							
	224		225	226	227	228	229	230	231	232							
F-	0	1	2	3	4	5	6	7	8	9							APC
	0030	0031	0032	0033	0034	0035	0036	0037	0038	0039							003f
	240	241	242	243	244	245	246	247	248	249							255
	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F	

characters, 33
 4 printable
 softlyphen)

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:

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, softhyphen)



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, softhyphen)



A popular joke:

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

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)



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, softhyphen)
- 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, 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

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, 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.
- a German, French, or Swedish, etc., programmer had to get used to reading and writing

ä aÄiÛ='Ön'; ü

instead of

```
{ a[i]='\\n'; }
```


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.

Computers into

- In Apr (late DEC ASCII addition character)

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
b ₇	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	1	1	1	1
b ₆	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1
b ₅	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
b ₄	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₃	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₂	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₁	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₀	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₇	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₆	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₅	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₄	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₃	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₂	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₁	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₀	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₇	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₆	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₅	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₄	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₃	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₂	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₁	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b ₀	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MA-94
 ter sets
 tions of
 and adds
 7 bit)

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).

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).
- 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?

New standards: Problems solved?

- Now 'Abs confi
- Now we (

```
fe ff 20 20 20 04 1f 04 40 04 3e 04 34 04 30 04 35 04 42 04
41 04 4f 2004 37 04 35 04 3c 04 3b 04 4f 20 04 41 04 35 04
3b 04 4c 04 45 04 3e 04 37 2004 3d 04 30 04 37 04 3d 04 30
04 47 04 35 04 3d 04 38 04 4f 20 20 20 20 04 3e 04 42 20 04
41 04 3e 04 31 04 41 04 42 04 32 04 35 04 3d 04 3d 04 38 04
3a 04 30 20 04 21 04 20 04 1e 04 27 04 1d 04 1e 0a 0c 0a 0c
04 23 04 47 04 30 04 41 04 42 04 3a 04 38 3a 32 34 04 33 04
30 02 62 20 37 33 32 30 04 33 04 30 2b 31 32 39 30 04 33 04
30 2b 31 33 36 31 04 33 04 30 04 26 04 35 04 3d 04 30 3a 38
30 30 30 04 40 04 43 04 31 04 3b 04 35 04 39 20 04 37 04 30
20 04 33 04 30 20 32 38 38 30 20 04 40 04 43 04 31 04 3b 04
35 04 39 2f 04 41 04 3e 04 42 04 3a 04 30 29 04 22 04 1e 04
20 04 13 0a 0c 0a 0c 04 23 04 47 04 30 04 41 04 42 04 3a 04
38 20 04 32 04 4b 04 34 04 35 04 3b 04 35 04 3d 04 4b 2b 20
04 41 04 32 04 38 04 34 04 35 04 42 04 35 04 3b 04 4c 04 41
04 42 04 32 04 30 20 04 23 04 24 04 20 04 21 20 04 3f 04 3e
04 3b 04 43 04 47 04 35 04 3d 04 4b 1b 20 04 20 04 30 04 41
04 3f 04 3e 04 3b 04 3e 04 36 04 35 04 3d 04 38 04 35 3a 20
04 23 04 3b 04 4c 04 4f 04 3d 04 3e 04 32 04 41 04 3a 04 30
04 4f 20 04 3e 04 31 04 3b 04 30 04 41 04 42 04 4c 2b 04 18
04 3d 04 37 04 35 04 3d 04 41 04 3a 04 38 04 39 20 04 40 04
30 04 39 04 3e 04 3d 1b 0a 0c 04 14 04 3e 04 3a 04 43 04 3c
04 35 04 3d 04 42 04 4b 20 04 3f 04 3e 20 04 37 04 30 04 3f
04 40 04 3e 04 41 04 43 1e 20 0a 0c 04 1a 04 3e 04 3d 04 42
04 30 04 3a 04 42 04 4b 3a 20 20 2b 20 37 20 32 38 20 39 49
36 32 39 20 39 39 04 27 2d 04 17 20 36 0a 0c 2d 36 31 2b 20
38 30 35 32 35 31 31 32 38 30 61 32 39 20 67 6d 61 69 6b 2e
63 6f 6d 0a 0c 04 1c 04 30 04 3a 04 41 04 38 04 3c 20 0a 0c
```

an
at do

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?

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.

New standards: Problems solved?

`\ufeff \u041f\u0440\u043e\u0434\u0430\u0430\u0435\u0442\u0441\u044f \u0437\u043c
\u043c\u043b\u044f \u0441\u0435\u043b\u044c\u0445\u044e\u0437 \u043d\u0430\u0430\u0437
\u043d\u0430\u0430\u0447\u0435\u043d\u0438\u044f
 \u043e\u0442 \u0441\u0435\u0431\u0430\u0441\u0442\u0432\u0435\u043d\u043d
\u0438\u0430\u0430 \u0421 \u0422 \u0415 \u0427 \u0414 \u0415`

`\u0423\u0447\u0430\u0441\u0442 \u0442 \u0430 \u0438: 24 \u0433 \u0430, 732 \u0433 \u0430,
1290 \u0433 \u0430, 1361 \u0433 \u0430`

`\u0426\u0435\u043d\u0430\u0430: 8000 \u0440\u0443\u0431\u043b\u0435\u0439 \u0437\u0430
\u0430 (80 \u0440\u0443\u0443\u0431\u043b\u0435\u0439/\u0441\u0435\u0442\u0430
\u0430) \u0422 \u0415 \u0420 \u0413`

`\u0423\u0447\u0430\u0430\u0441\u0442 \u0430 \u0438 \u0432 \u0430 \u0435 \u043b \u0435
\u043d \u044b, \u0441 \u0432 \u0438 \u0434 \u0435 \u0442 \u0435 \u043b \u044c \u0441
\u0432 \u0430 \u0423 \u0424 \u0420 \u0421 \u043f \u0435 \u043b \u0443 \u0447 \u0435 \u043d
\u044b.`

`\u0420 \u0430 \u0441 \u0441 \u043f \u0435 \u043b \u0435 \u0436 \u0435 \u043d \u0438 \u0435:
\u043b \u044c \u044f \u043d \u0435 \u0432 \u0441 \u0430 \u0430 \u044f \u0435 \u0431
\u0430 \u0441 \u0442 \u044c, \u0418 \u043d \u0437 \u0437 \u0435 \u043d \u0441 \u0430 \u0438
\u0440 \u0430 \u0439 \u0435 \u043d.`

`\u0414 \u0435 \u0430 \u0443 \u043c \u0435 \u043d \u0442 \u044b \u043f \u0435 \u0437
\u043f \u0440 \u0435 \u0441 \u0443.`

`\u0410 \u0435 \u043d \u0442 \u0430 \u0442 \u0430 \u0442 \u044b: +7 (916) 99 \u0427-\u04176-61,
8052511(a)gmail.com`

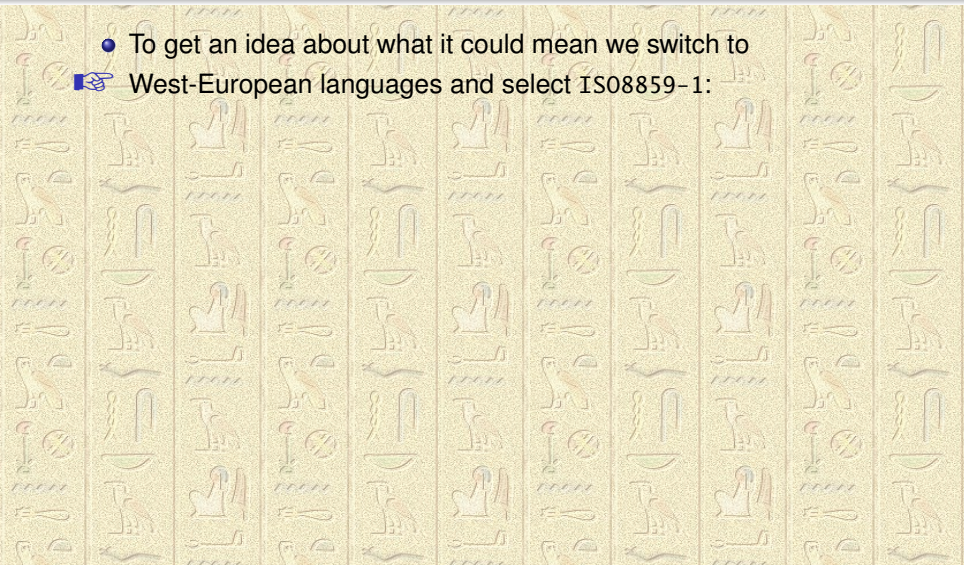
`\u041c \u0430 \u0441 \u0441 \u0438 \u043c`

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')...

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

δὸϊἰἄἄἄοὐὐ ὕἄἰἰἢ ὀἄἰϕεἰῦ ἱἄῦἱἄἱἄἱἔἢ
ἰἰ ὀἰἄἄἄ×ἄἱἱἔἔἄ ὀδἰἰἱ

δἰἄἄἄἄἄἄ: 24 ςἄ, 732 ςἄ, 1290 ςἄ, 1361 ςἄ
ἄἄἱἄ: 8000 ὀἄἱἄἄἄ ὕἄ ςἄ (80 ὀἄἱἄἄἄ/ὀἰἄἄἄ) ὀἰδς

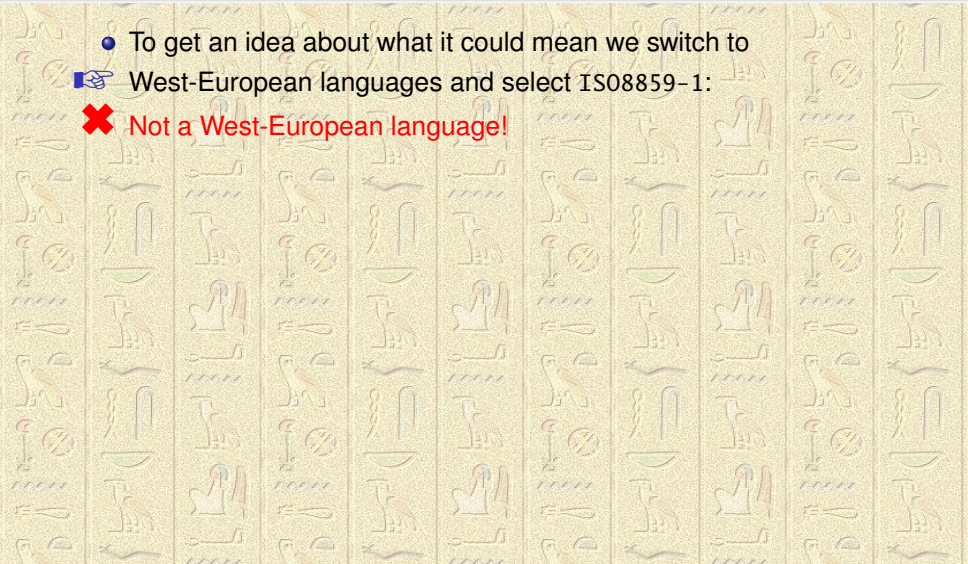
δἰἄἄἄἄἄἄ ×ὑἄἄἱἄἱῦ, ὀ×ἔἄἄἄἄἄἄἄἄ×ἄ ὀδδδ δἰἰἄἄἄἄἄῦ.
δἄἄἄἄἄἄἄἄἄἄ: δἰϕἢἱ×ὀἔἄἢ ἱἄἱἄἄἄἄ, εἰῦἄἱἄἄἄἄἄἄ ὀἄἄἄἄἄ.

ἄἱἄἄἄἄἄἄἄῦ δἰ ὕἄἄἄἄἄἄῦ.


ἄἱἄἄἄἄἄἄῦ: +7 (916) 99ἰ-ὑ6-61, 8052511(a)gmail.com
ἱἄἄἄἄἄἄ

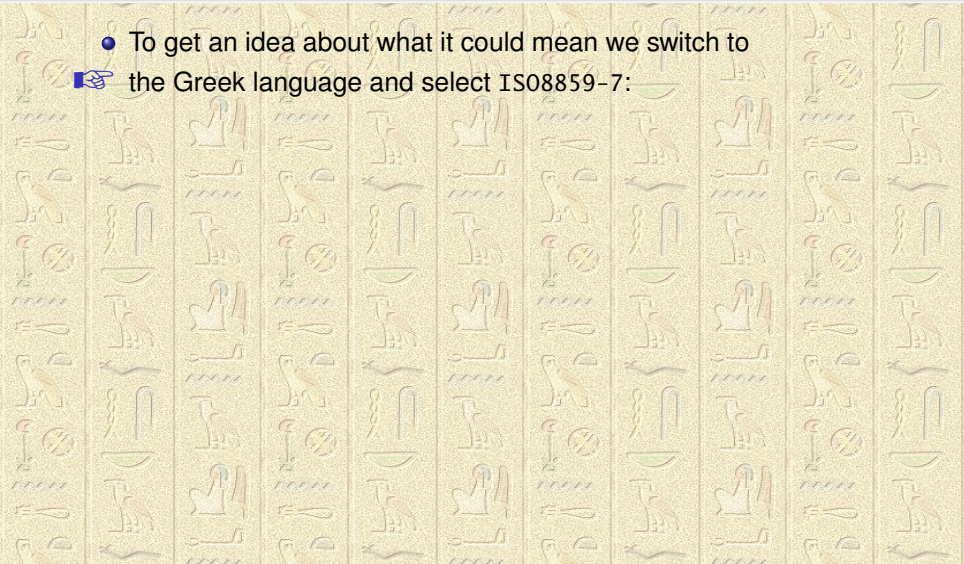
Let's try to uncover the secret

- 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

πϙΟΔΑΕΤΣΡ ïΕΝΜΡ ΣΕΜΨΘΟÏ ΣΑÏΣΑήΕΞΙΡ
ΟΤ ΣΟΒΣΤΧΕΞΞΙΛΑ σςοώξο

υήΑΣΤΛΙ: 24 ΗΑ, 732 ΗΑ, 1290 ΗΑ, 1361 ΗΑ

γΕΞΑ: 8000 ϙΥΒΜΕΚ ïΑ ΗΑ (80 ϙΥΒΜΕΚ/ΣΟΤΛΑ) τοςη

υήΑΣΤΛΙ ΧΩΔΕΜΕΞΩ, ΣΧΙΔΕΤΕΜΨΕΤΧΑ υζςσ ΠΟΜΥήΕΞΩ.

ςΑΣΠΟΜΟΦΕΞΙΕ: υΜΨΡΕΟΧΣΛΑΡ ΟΒΜΑΣΤΨ, ιΞÏΕΞΣΛΙΚ ϙΑΚΟΞ.

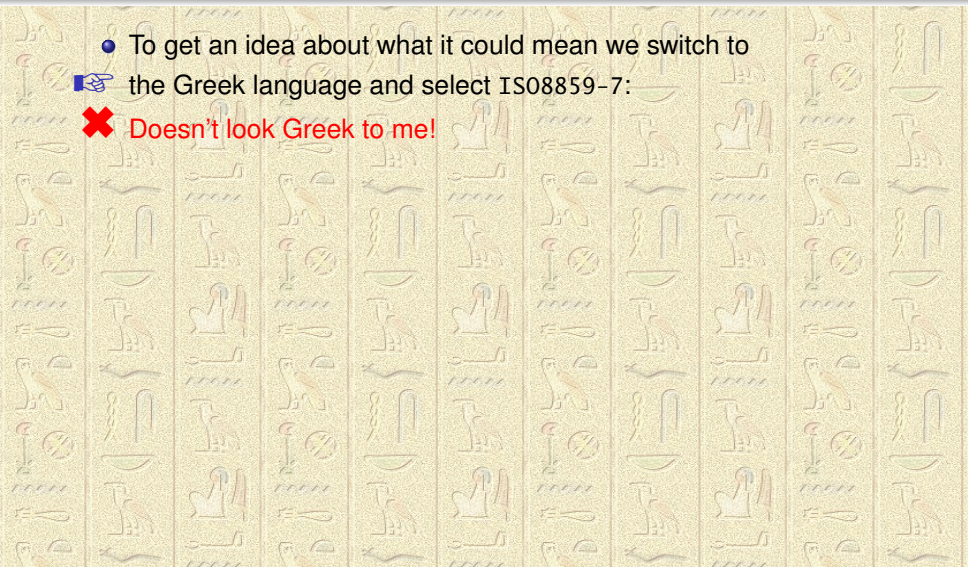
ΔΟΛΥΝΕΣΤΩ ΠΟ ïΑΠϙΟΣΥ.

ΛΟΣΤΑΛΤΩ: +7 (916) 99ώ-ι6-61, 8052511(a)gmail.com


υΑΛΣΙΝ

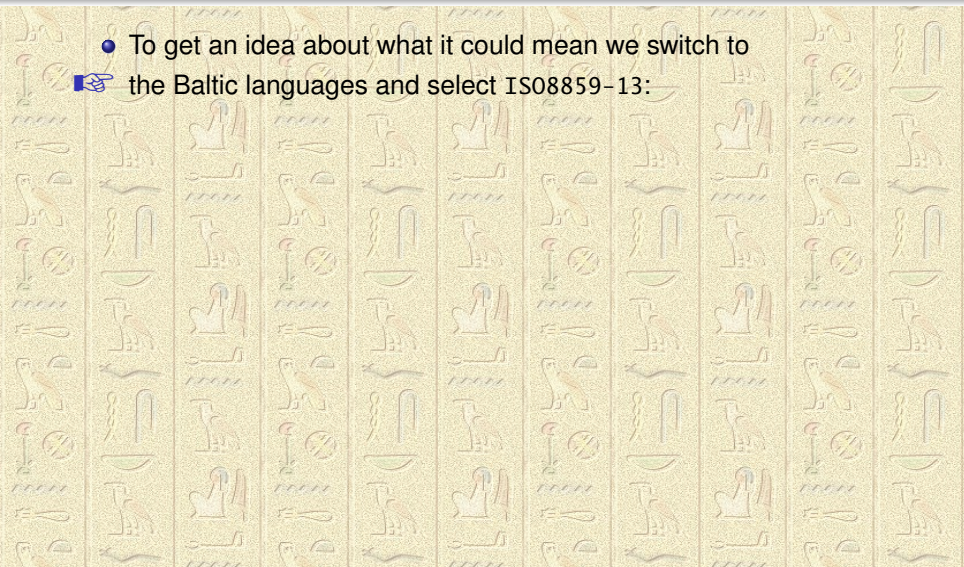
Let's try to uncover the secret

- 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

ŠŇĹĀİĀŌŌŃ ŠĀŖĜŇ ÓĀĜŮĹŚ İİŚİİŽĀİÉN
ĹŌ ÓĹĀÓŌ×ĀİİÉEİ óŋłżił

ŏŽİŌŌÉEÉ: 24 Ęİ, 732 Ęİ, 1290 Ęİ, 1361 Ęİ
ĊĀİİ: 8000 ŃŌĀĜĀŽ Śİ Ęİ (80 ŃŌĀĜĀŽ/ÓĹŌÉEİ) ōłŋē

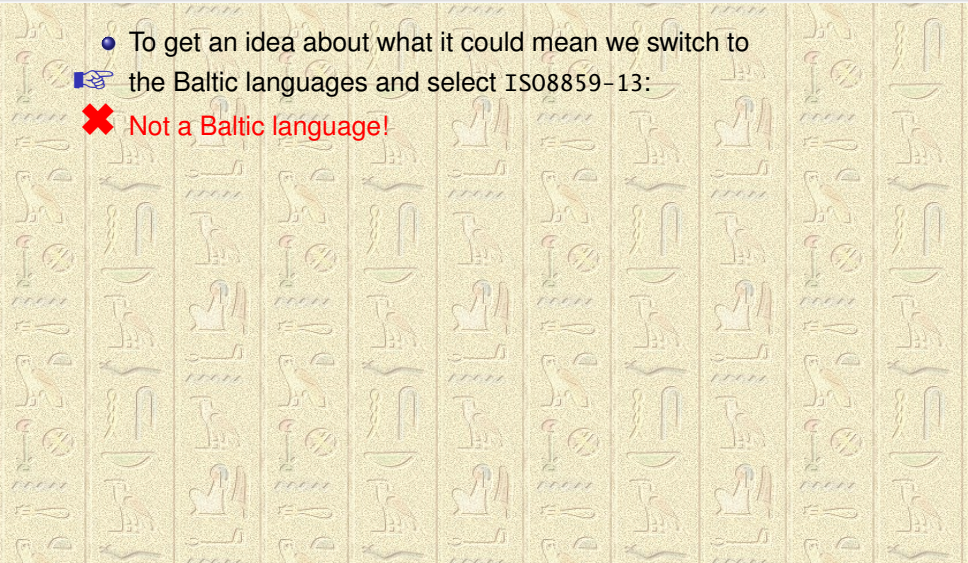
ŏŽİŌŌÉEÉ ×ĹĀĀĜĀİĹ, Ó×ÉĀĀŌĀĜŮŌŌ×İ ōęŋŏ ŠĹĜŌŽĀİĹ.
ŋİŌŚĹĸĹŌĀİÉĀ: ōĜŮŇİĹ×ŌÉEİŇ ĹĀĜİŌŌŮ, éİŚĀİŌÉEÉŽ ŇİŽĹİ.

āĹĘŌĶĀİŌĹ ŚĹ ŚİŠŇĹŌŌ.


èĹİŌİÉŌĹ: +7 (9İ6) 99ž-ś6-61, 8052511(a)gmail.com
ķİÉÓÉĶ

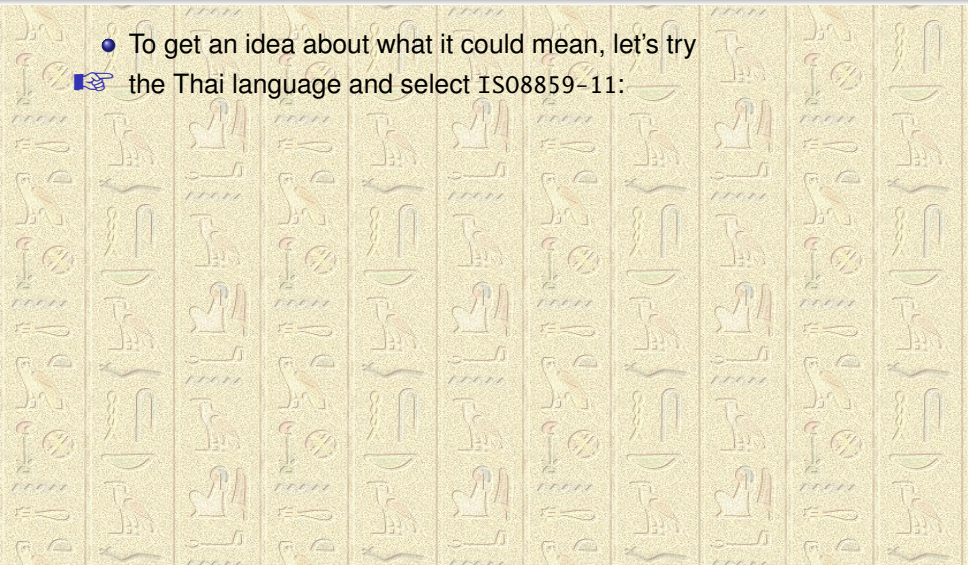
Let's try to uncover the secret

- 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

๐๑๒๓๔๕๖๗๘๙ ๐๑๒๓๔๕๖๗๘๙ ๐๑๒๓๔๕๖๗๘๙
๐๑๒๓๔๕๖๗๘๙ ๐๑๒๓๔๕๖๗๘๙ ๐๑๒๓๔๕๖๗๘๙

๕๐๖๗๘๙: 24 บม, 732 บม, 1290 บม, 1361 บม
ไลสม: 8000 ็ยฬลส ม บม (80 ็ยฬลส/าฬล) ๕๐๖๗

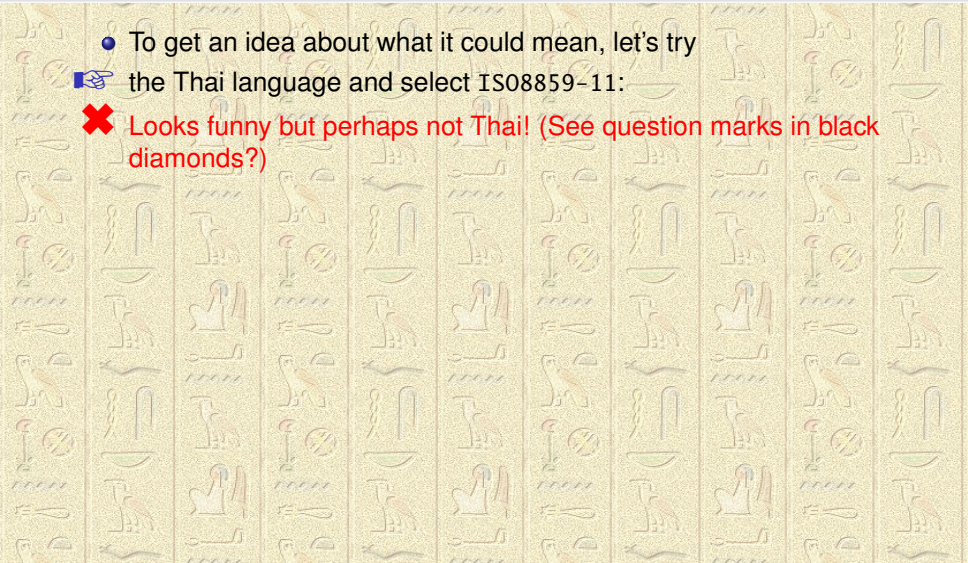
๕๐๖๗๘๙ ็ลฬลฬล, ็ลฬลฬลฬล ๕๐๖๗ ๕๐๖๗๘๙.
๒๓๔๕๖๗๘๙: ๕๐๖๗๘๙ ๕๐๖๗๘๙, ็ลฬลฬลฬล ๕๐๖๗๘๙.

๕๐๖๗๘๙ ๕๐๖๗๘๙.


๕๐๖๗๘๙: +7 (916) 99๐-๕๐๖๗-๘๙, 8052511(a)gmail.com
๕๐๖๗๘๙

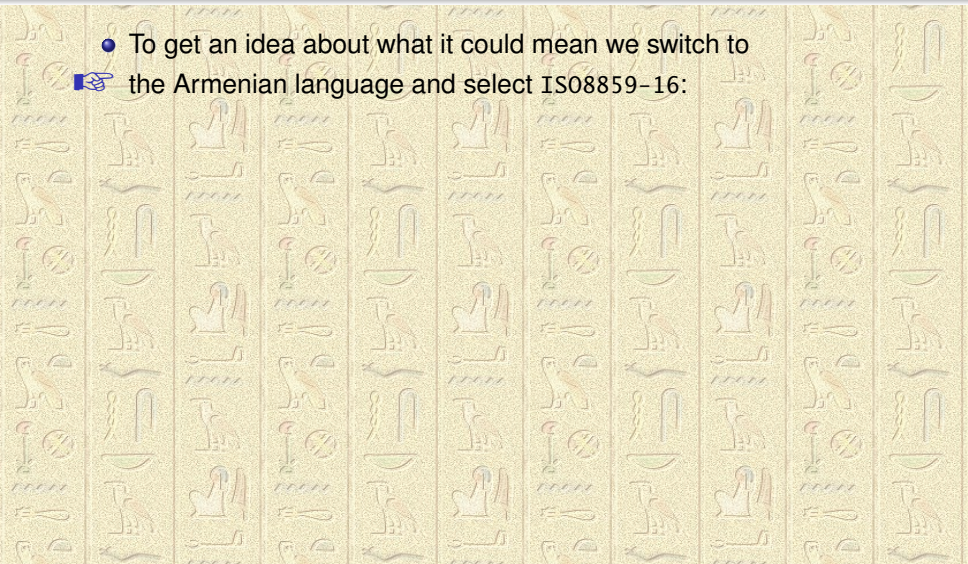
Let's try to uncover the secret

- 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

ᐱᑦᑲᑦᑲᑦᑲᑦᑲᑦ ᑲᑦᑲᑦᑲᑦᑲᑦ ᑲᑦᑲᑦᑲᑦᑲᑦ ᑲᑦᑲᑦᑲᑦᑲᑦ
ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ ᑲᑦᑲᑦᑲᑦᑲᑦ

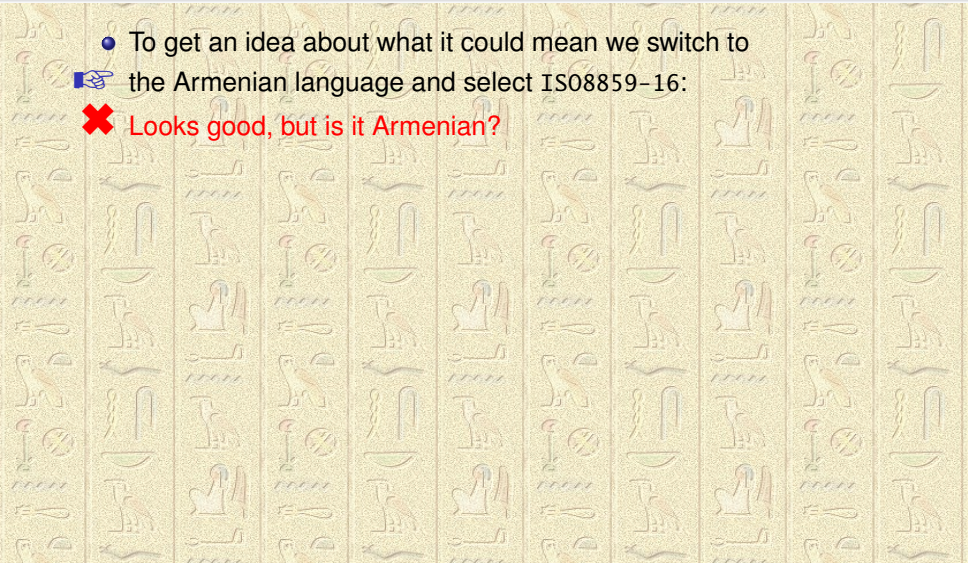
ᐱᑦᑲᑦᑲᑦᑲᑦ: 24 ᑲᑦᑲᑦ, 732 ᑲᑦᑲᑦ, 1290 ᑲᑦᑲᑦ, 1361 ᑲᑦᑲᑦ
ᑲᑦᑲᑦᑲᑦ: 8000 ᑲᑦᑲᑦᑲᑦᑲᑦ ᑲᑦᑲᑦ ᑲᑦᑲᑦ (80 ᑲᑦᑲᑦᑲᑦᑲᑦ/ᑲᑦᑲᑦᑲᑦᑲᑦ) ᑲᑦᑲᑦᑲᑦ

ᐱᑦᑲᑦᑲᑦᑲᑦ ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ, ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ ᐱᑦᑲᑦᑲᑦ ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ.
ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ: ᐱᑦᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ, ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ.


ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ ᑲᑦᑲᑦ ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ.
ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ: +7 (916) 99'-06-61, 8052511(a)gmail.com
ᑲᑦᑲᑦᑲᑦᑲᑦᑲᑦ

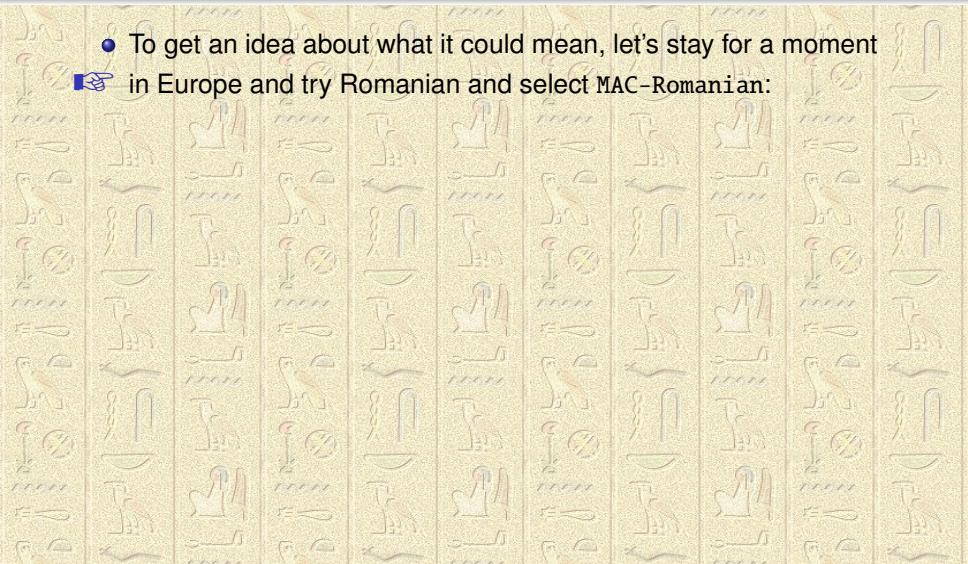
Let's try to uncover the secret

- 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:



Let's try to uncover the secret

? "æ f j ≈ ' " - / ≈ Õ Ã - " ≈ Ã ÿ » ce / E i / E i T ≈ E ... -
ce ' " ce Ñ " " ◊ ≈ E E ... À i U Ú Ô , Ó Ô

1 T i " " À ... : 24 « i , 732 « i , 1290 « i , 1361 « i
" ≈ E i : 8000 " " Ñ Ñ ≈ / i « i (80 " " Ñ Ñ ≈ / " ce ' À i) U Ô Ú Á

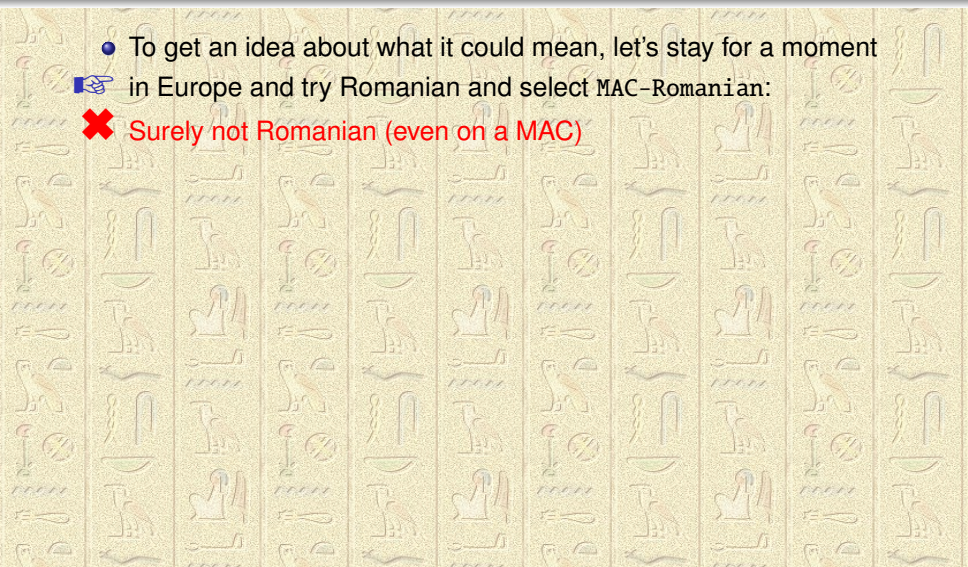
1 T i " " À ... ◊ ÿ f ≈ Ã ≈ E ÿ , " ◊ ... f ≈ ' ≈ Ã ÿ " " ◊ i 1 E Ú Ú - ce Ã ' T ≈ E ÿ .
Ú i " - ce Ã ce ÷ ≈ E ... ≈ : 1 Ã ÿ - E ce ◊ " À i - ce Ñ Ñ i " " ÿ , È E / ≈ E " À ... " i ce E .

% ce À ' Õ ≈ E ' ÿ - ce / i - " ce " " .

Î ce E ' i À ' ÿ : +7 (916) 99 , - ' 6 - 61 , 8052511(a)gmail.com
Ï i À " ... Õ

Let's try to uncover the secret

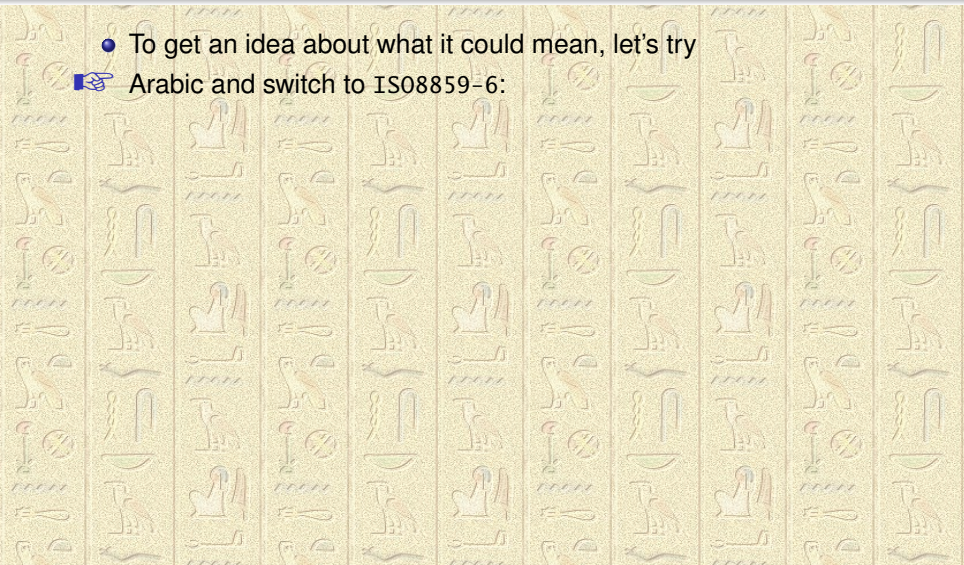
- To get an idea about what it could mean, let's stay for a moment in Europe and try Romanian and select MAC-Romanian:
 - ✗ Surely not Romanian (even on a MAC)



Let's try to uncover the secret

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

☞ Arabic and switch to ISO8859-6:



Let's try to uncover the secret

زدؤء إسرر غإجر ساإظبدغ ءغء ءغء إخر
 دش سدا سشطإخرة ء

ءشءة: ٢٤ ء، ٧٣٢ ء، ١٢٩٠ ء، ١٣٦١ ء
 كإء: ٨٠٠٠ زماجات ء ء (٨٠ زماجات/سءشءة) ء

ءشءة طءؤ إءإء؁ سطة ؤإشاإظسشءة ن ء ذءءص إءء.
 ءذءءءإءة إ: ءظرخءطسشءر ءآء ءشظ؁ ءءءإءشءة زءءءء.

لءءصءإءشء ذء ءغ ءذءءص.

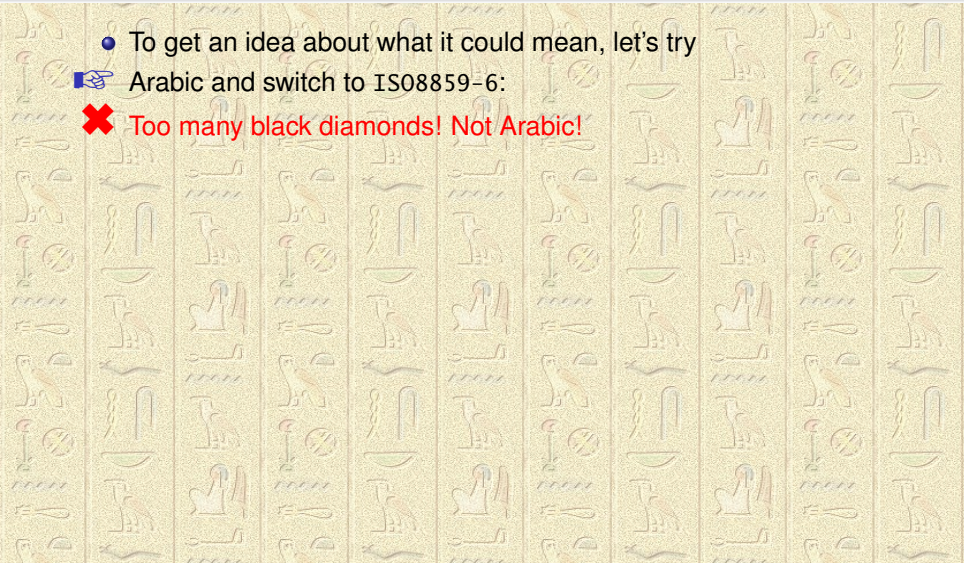
9) ٧+ : ءشءة ءشء: 99-6-61, 8052511(a)gmail.com
 ءشءءء.

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

鸚夏僚杂 讵吞 优特认 瘟悞赁盼裳
显 酉掠宰盼紊肆 篁稔辑

蹠劣运: 24 橐, 732 橐, 1290 橐, 1361 橐
闷瘟: 8000 艺绿攀 讵 橐 (80 艺绿攀/酉运) 麸蟀

蹠劣运 踪呐膛钨, 幼贍旁盘蹠宰 蹠蝮 邢陶乔钨.
姑有咸现盼膳: 跽匱蜗子肆 下塘釉, 榭讵幹松 伊氏.

洒苏团卧 邢 讵幸嫌.
脾卧了再: +7 (9I6) 99-1, 8052511(a)gmail.com
破擻赏


Let's try to uncover the secret

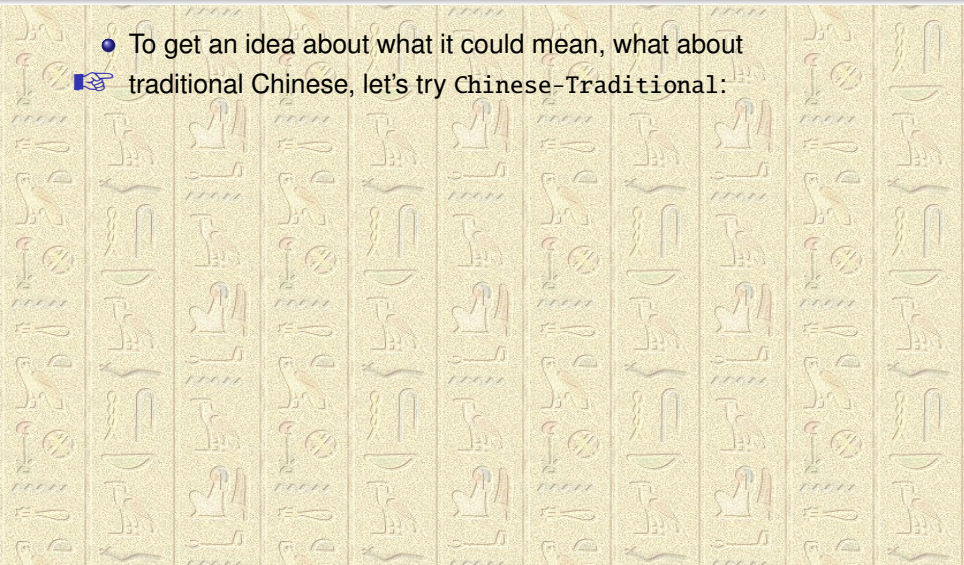
- To get an idea about what it could mean, let us assume it's our Chinese colleague at CERN,

☞ 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

導花 唐 情性 苑珂缶 氛悸 欠啞
近 衍 哥欠沽芒 檔骷複

輾剛: 24 共, 732 共, 1290 共, 1361 共
齋氛: 8000 皆日 惋 共 (80 皆日/衍剛) 縹錄

輾剛 耿千咀治, 負呆氏月迸哥 邈雕 垠咎喜治.
謂衫芷邸欠呢: 謠追泓耄芒 芟呵訂, 漣情況豕 狡抑.

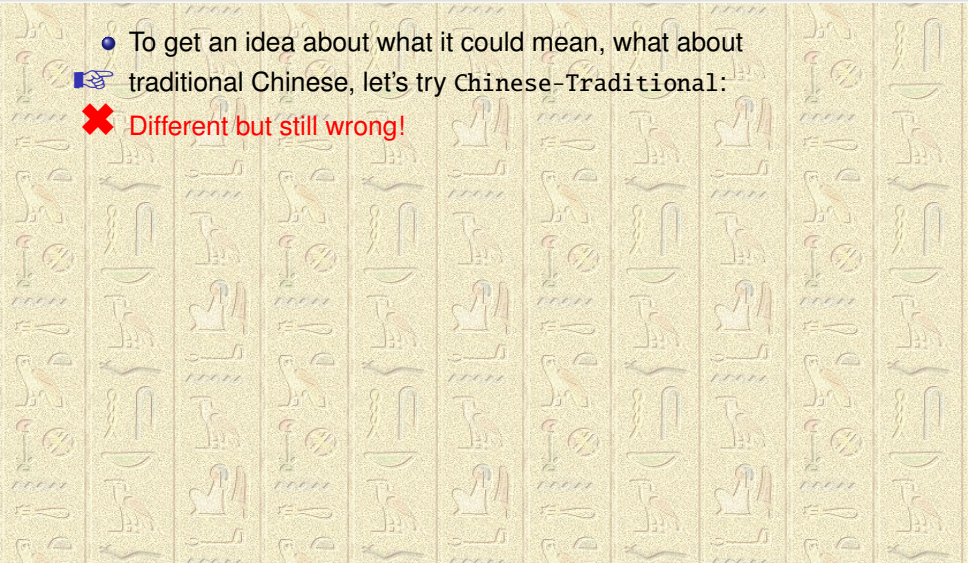
積迄怔沮 垠 惋城返.

鉸沮 唆: +7 (9I6) 99-6-61, 8052511(a)gmail.com
縱迤吻

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:

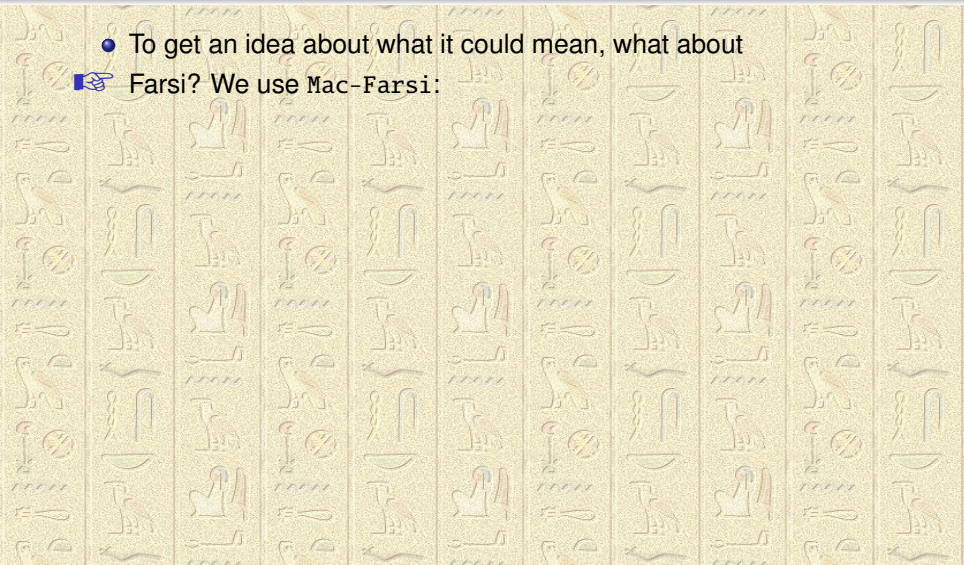
✘ Different but still wrong!



Let's try to uncover the secret

- To get an idea about what it could mean, what about

👉 Farsi? We use Mac-Farsi:

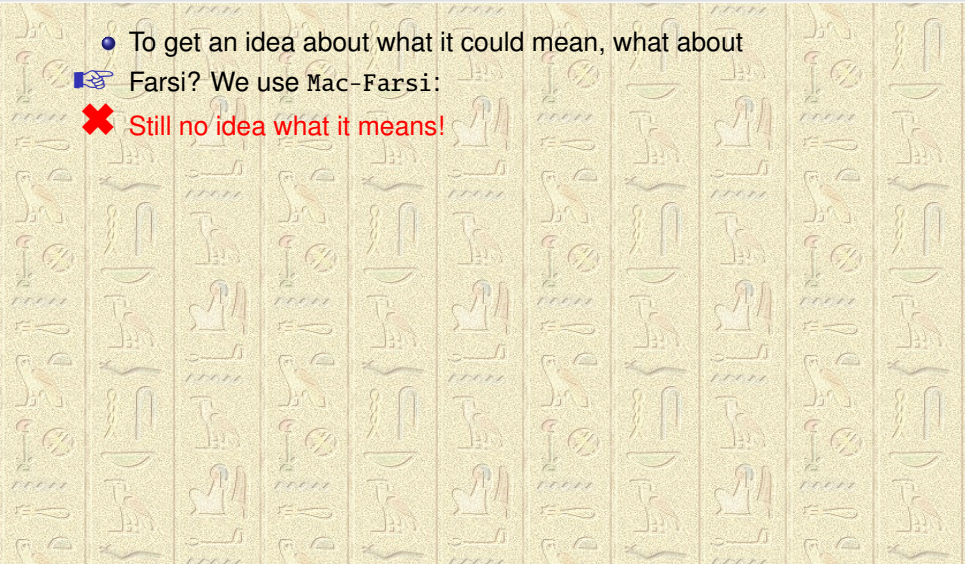


Let's try to uncover the secret

- To get an idea about what it could mean, what about

☞ Farsi? We use Mac-Farsi:

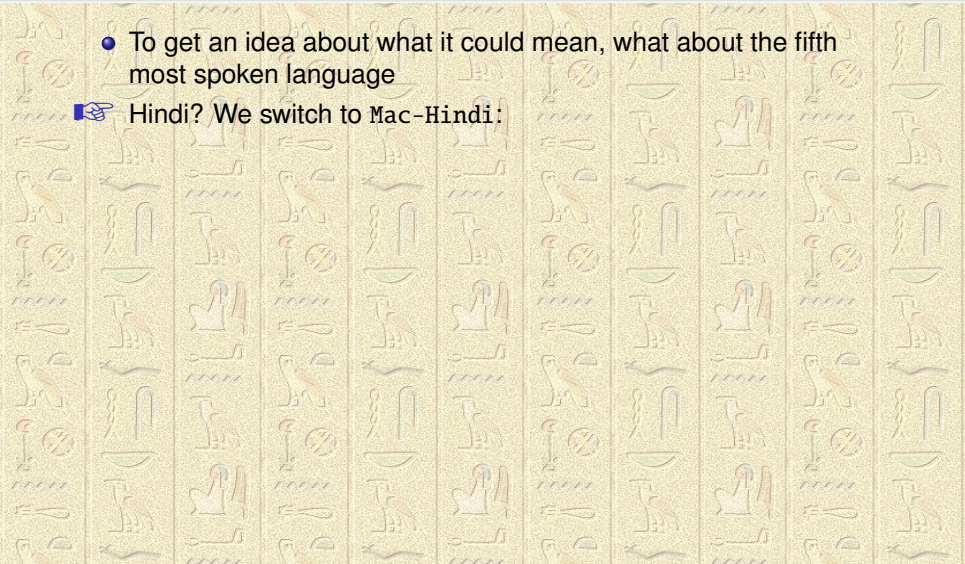
✘ **Still no idea what it means!**



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

❖ करदणधवळल ाधयमल ळधमहपरा यणायणूधयफल
रव ळरतळवसधययफभण २१ ❖❖❖❖

यूणळवभफ: 24 तण, 732 तण, 1290 तण, 1361 तण

धयण: 8000 कशतमधव ण तण (80 कशतमधव/ळरवभण) ३ ❖ १॥

यूणळवभफ सदधमधय, ळसफदधवधमहळवसण ४०१२ ररमशूधय.

१णळररमरषधयफध: ४महलयरसळभणल रतमणळवह, य़ाधयळभफव ळणवरय.

ेरभशयधयव रर णरळरळश.

❖ रयवणभव: +7 (916) 99❖-१6-61, 8052511(a)gmail.com

❖ णभळफय

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:

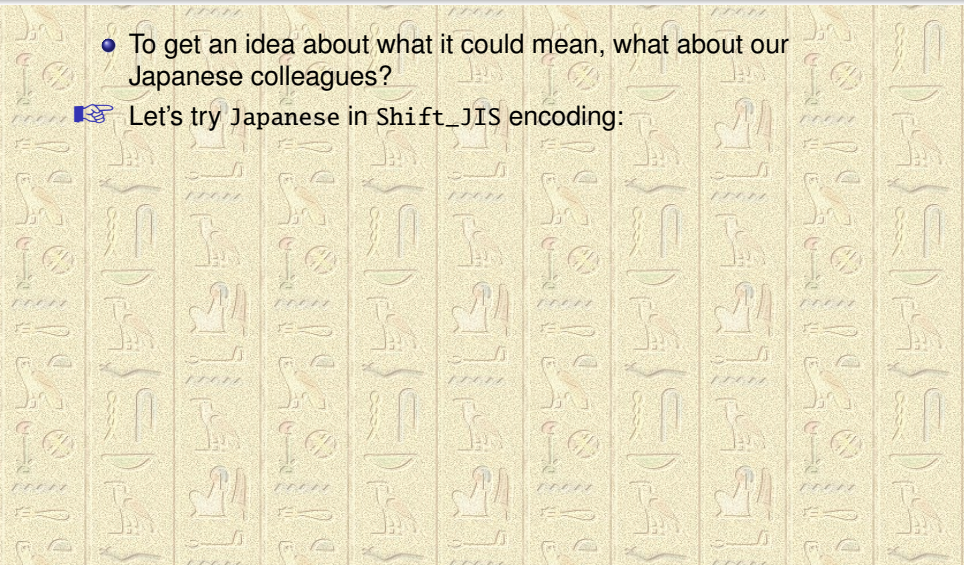
✘ Still black diamonds!

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

?マトチナヤモム レナヘアム モナアリネマレ ホチレホチナホノム
マヤ モマツモヤラナホホノヒチ ? · i

?チモヤヒノ: 24 スチ, 732 スチ, 1290 スチ, 1361 スチ

翡翠ホチ: 8000 メユツアサハ レチ スチ (80 メユツアサハ/モマヤヒチ) ??

?チモヤヒノ ラルトナフナホル, モラノナヤナアリモヤラチ ?? ミマフユナホル.

?モミマフマヨナホノサ: ?リムホマラモヒチム マツフチモヤリ, 魚念レナホモヒノハ メチハマホ.

凄ヒユヘナホヤル ミマ レチミメマモユ.

・ホヤチヒヤル: +7 (916) 99?- · -61, 8052511(a)gmail.com

晴ヒモノハ

Let's try to uncover the secret

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

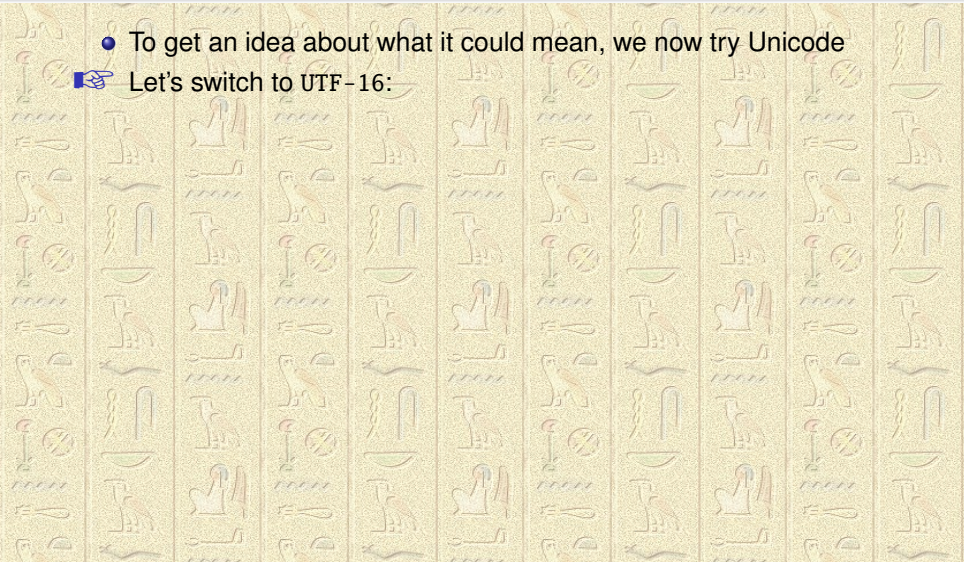
☞ Let's try Japanese in Shift_JIS encoding:

✘ **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:

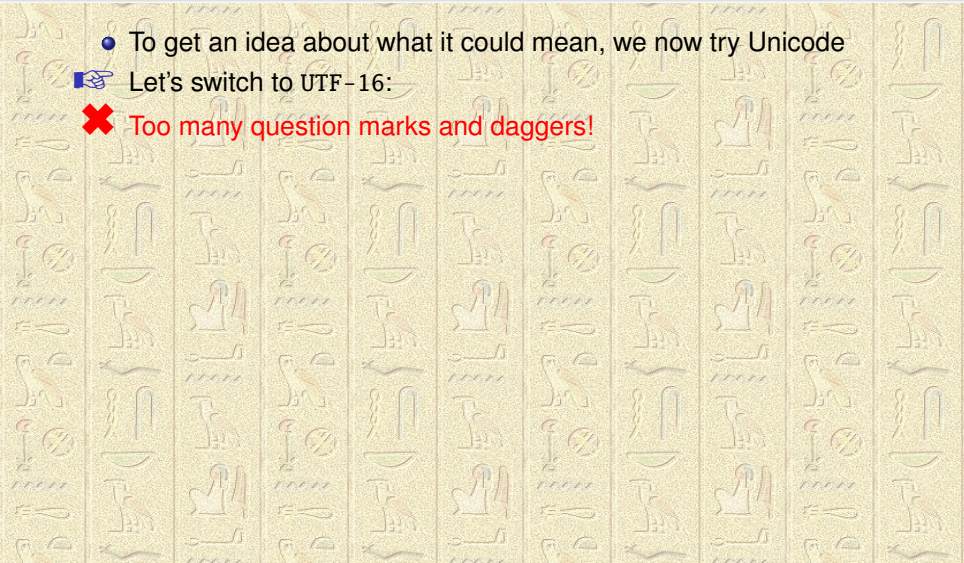
+++?교색핀택◆취택편철쫘◆강◆쌔엮집†하+++++콰।퀵펜?갹집선?@?ㄱ?◆펜
쫘?ㄱ?삼?삼%代ㄱ쟁?ㄱ??선ㄱ?강??~|햇채짜◆?선?ㄱ특싫엇?콰쫘
???ㄱ?◆펜쫘+◆엮엮◆।?쫘핑철펜????탈천◆갹??쌔탕칭촉꺽?첼타긱꺽꺽?싫
싫꺽??◆갹쫘짜쯉쯉쫘ㄱ?긱헛엮꺽-긱◆타꺽꺽꺽?갹쌔꺽?????????广+???X恨
杭慥氮捌洋ㄱ?쫘쫘ㄱ

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:

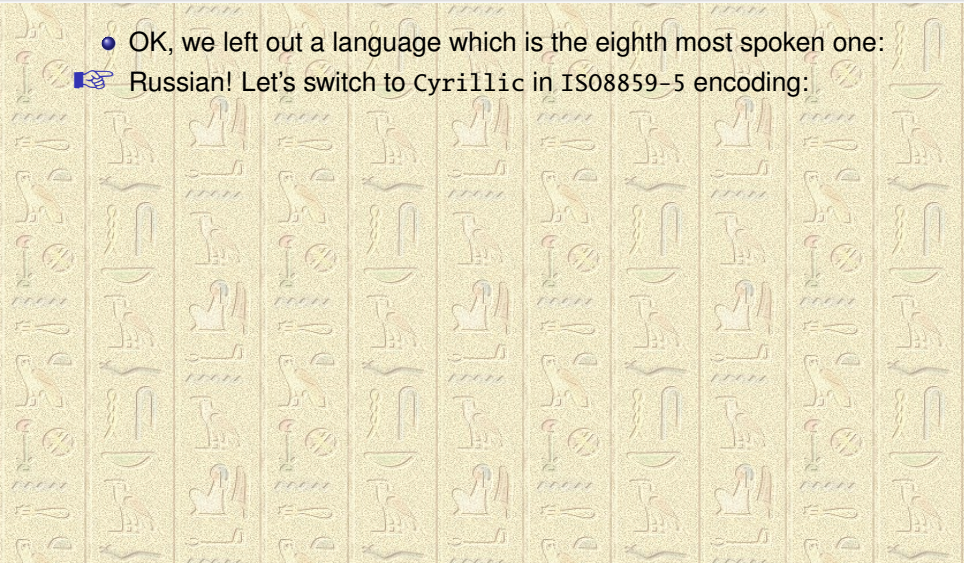
✘ Too many question marks and daggers!



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, Инзенский район.

Документы по запросу.

Контакты: +7 (916) 994-36-61, 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:

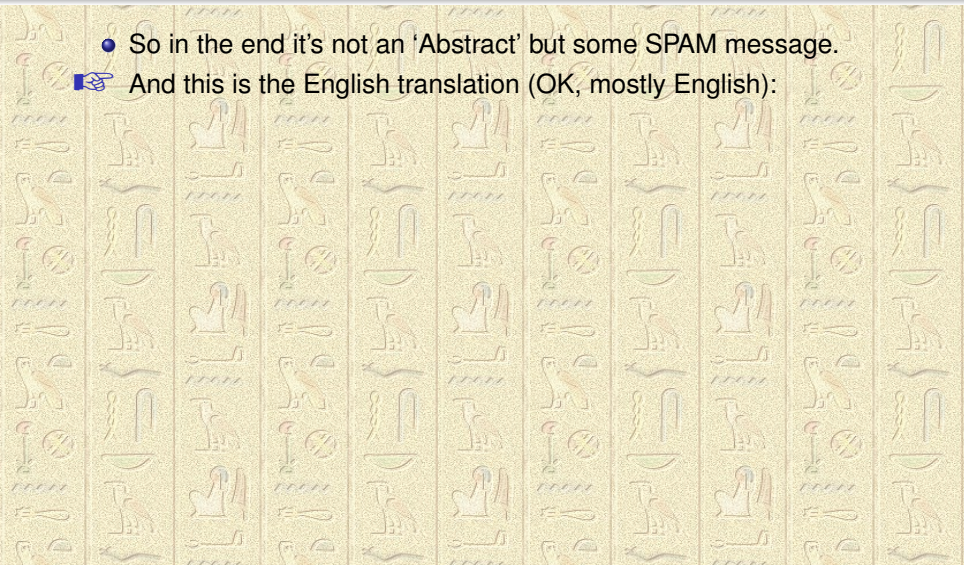
☞ Russian! Let's switch to Cyrillic in ISO8859-5 encoding:

☞ 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):



Let's try to uncover the secret

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 УФРС are received.
Arrangement: the Ulyanovsk area, area Inzensky.

Documents by inquiry.
Contacts: 7 (916) 994-36-61, 8052511 (a) gmail.com
Maxim

Encoding

- We now have understood that **Encoding** is important as it points to the right character: `x'a4'` + IS08859-15 \Rightarrow €


Encoding

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



Encoding

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

Encoding

- We now have understood that **Encoding** is important as it points to the right character: $x \text{ 'a4' } + \text{ISO8859-15} \Rightarrow \text{€}$
 - If we would have used: $x \text{ 'a4' } + \text{ISO8859-14} \Rightarrow \text{Ć}$
 - 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)

Encoding

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

Multiplication and division sign operators

2A2F	×	VECTOR OR CROSS PRODUCT
		\rightarrow 00D7 \times multiplication sign
2A30	×	MULTIPLICATION SIGN WITH DOT ABOVE
2A31	×	MULTIPLICATION SIGN WITH UNDERBAR
2A32	×	SEMIDIRECT PRODUCT WITH BOTTOM CLOSED
2A33	⊗	SMASH PRODUCT
2A34	⊗	MULTIPLICATION SIGN IN LEFT HALF CIRCLE
2A35	⊗	MULTIPLICATION SIGN IN RIGHT HALF CIRCLE
2A36	⊗	CIRCLED MULTIPLICATION SIGN WITH CIRCUMFLEX ACCENT
2A37	⊗	MULTIPLICATION SIGN IN DOUBLE CIRCLE
2A38	⊕	CIRCLED DIVISION SIGN

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 256, 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 encoding

```
/BaseEncoding [
% 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 /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 /braceleft /bar /braceright /asciitilde /.notdef
% 0x80
/Euro /.notdef /quotesinglbase /florin
/quotedblbase /ellipsis /dagger /daggerdbl
/circumflex /perthousand /Scaron /guilsinglleft
/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! )
```

more than the
of the code point

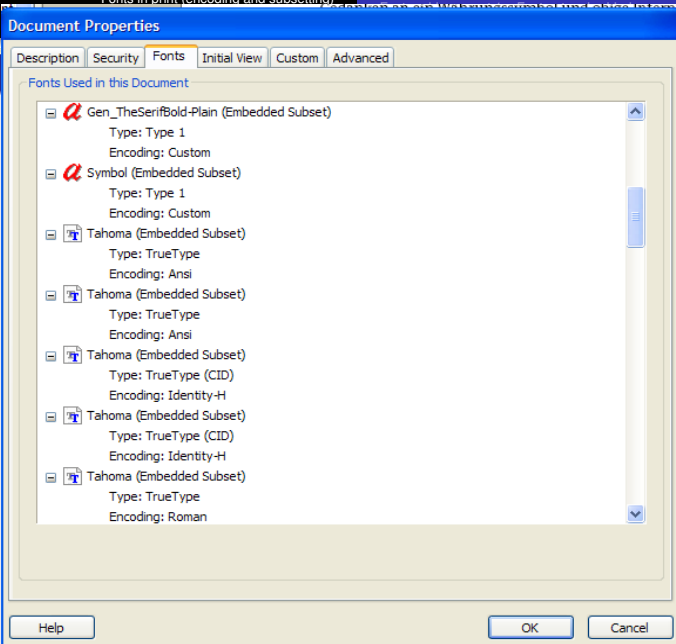
look like this:

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 look 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:



Map

ne
point
:
n?
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)

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 $\approx 35\%$ 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.

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”.

Xerox Phaser 7700DX Advanced Options



Xerox Phaser 7700DX Advanced Document Settings

- Paper/Output
 - Copy Count: 1 Copy
- Graphic
 - Image Color Management
 - ICM Method: ICM Disabled
 - ICM Intent: Pictures
 - Scaling: 100 %
 - TrueType Font: Substitute with Device Font
- Document Options
 - PostScript Options
 - Download as Softfont
 - Printer Features
 - Image Smoothing: Off
 - Offset Collated Sets: On

OK

Cancel

Why
Just
confe



ers missing

d the
ript

was

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.

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 $\neq 0$ (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). [Like during PAC'09 on Mac computers]

- *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.

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?

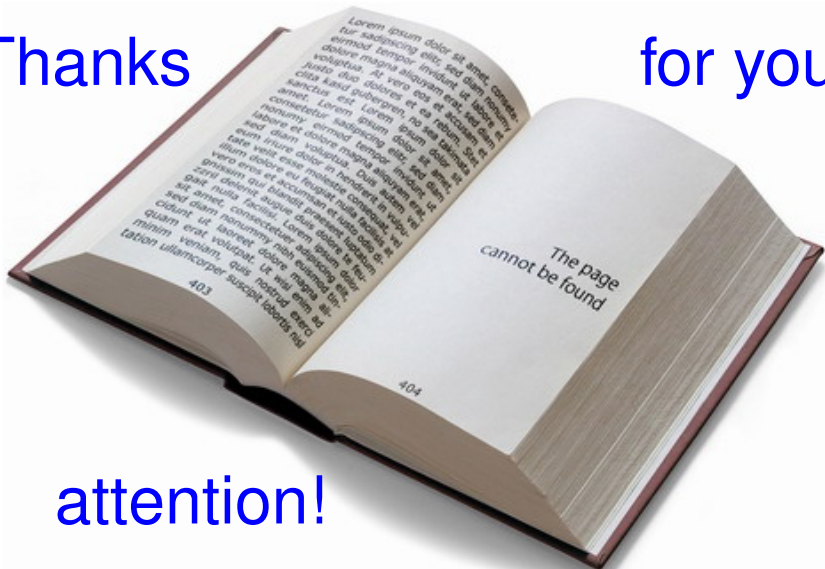
- Make sure when installing Word on PC or Mac that the newest fonts are installed and that there is only one version on the computer.
- Make sure that the newest firmware is loaded on printers used in the editorial office.
- The editors should check twice that fonts are embedded.
- We should provide 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!