Character conversion

126476_PixelRule.tiff ¬

When converting in either direction, any character below 127 is left unchanged (with the exception of single quote conversion, and four characters mentioned below).

To determine what conversions would be needed for the upper 128 characters, I used the character encoding found in Mac type families like Times. This is a superset of the character set defined in Inside Mac volume I. For the NeXT, I used the character encoding documented in the NeXTSTEP 2.1 Summaries/06_KeyInfo document. Note that: ZapfDingbats and Symbol turn out to have identical encodings, so are not touched, of course.

When converting from the Macintosh to the NeXT, many of the characters above 127 have straight forward conversions (e.g. Mac bullet to NeXT bullet). Some characters in the Mac encoding, however, don't have an equivalent in the NeXT encoding, but do have an equivalent in Symbol. So, Convert RTF converts these Symbol in the NeXT rtf document (and adds a font table entry for Symbol if needed). For those few remaining characters that can't be converted, I decided to write them out with their PostScript names in brackets. While this has the potential for messing up a document, it does make it pretty clear when it could not be converted properly, and what should have been there instead (I feel that `[apple]' is much clearer than `ð' in my converted document when there was an apple character in the source one). The following list shows the characters that can't be converted. On the left is the character's number in hex, and on the right is the name displayed in the document (an example of 0xD9, in Geneva on the mac, is a picture of a sheep, or in New York it is a picture of a robot):

0x11	commandsymbol

0x12	check
0x13	diamond
0x14	apple
0xD9	Ydieresis (often a picture in `old' fonts)
0xF0	apple

The NeXT to Mac conversion is, in some ways, simpler. All characters in the NeXT encoding either map to a character in the standard Mac encoding, or they don't, and there's no chance for using Symbol instead. The characters that can't be converted are also written as bracketed PostScript names. The following table, with the same format as the one above, lists the characters that can't be converted directly onto the Macintosh:

0x90	Eth
0x9B	Yacute
0x9C	Thorn
0x9E	multiply
0xB5	brokenbar
0xC0	onesuperior
0xC9	twosuperior
0xCC	threesuperior
0xD2	onequarter
0xD3	onehalf
0xD4	threequarters
0xE6	eth
0xE8	Lslash
0xF7	yacute
0xF8	lslash
0xFC	thorn
0xFE	not assigned
0xFF	ascii control char

Another issue relevant to character conversions is how characters are stored in the rtf document. Mac RTF files that I have seen generally use the \'AA notation to encode a character above 127.

NeXT, on the other hand, seems to avoid this and use the literal 8 bit character values instead. Regardless of my opinion on which is a better scheme, I felt only safe maintaining each platform's conventions. Thus, all Mac \' are converted to 8 bit characters on the NeXT, and 8 bit characters on the NeXT are converted to \' for the Mac. I do make the assumption that all Mac applications write out the \' notation. If I am wrong, then some characters will not be converted to the NeXT properly, and you should let me know what application generates such files.