NAME

preconv − convert encoding of input files to something GNU troff understands

SYNOPSIS

preconv [**−dr**] [**−D** *default_encoding*] [**−e** *encoding*] [*file* . . .]
preconv −h
preconv −−help

preconv −v
preconv −−version

DESCRIPTION

preconv reads *files* and converts its encoding(s) to a form GNU **troff**(1) can process, sending the data to standard output. Currently, this means ASCII characters and '\[uXXXX]' entities, where 'XXXX' is a hexadecimal number with four to six digits, representing a Unicode input code. Normally, **preconv** should be invoked with the **−k** and **−K** options of **groff**.

OPTIONS

Whitespace is permitted between a command-line option and its argument.

−d Emit debugging messages to standard error (mainly the used encoding).

−D*encoding*
Specify default encoding if everything fails (see below).

−e*encoding*
Specify input encoding explicitly, overriding all other methods. This corresponds to **groff**'s **−K***encoding* option. Without this switch, **preconv** uses the algorithm described below to select the input encoding.

−−help
−h Print a help message and exit.

−r Do not add .lf requests.

−−version
−v Print the version number and exit.

USAGE

preconv tries to find the input encoding with the following algorithm.

1. If the input encoding has been explicitly specified with option **−e**, use it.
2. Otherwise, check whether the input starts with a *Byte Order Mark* (BOM, see below). If found, use it.
3. Otherwise, check whether there is a known *coding tag* (see below) in either the first or second input line. If found, use it.
4 Finally, if the **uchardet** library (an encoding detector library available on most major distributions) is available on the system, use it to try to detect the encoding of the file.
5. If everything fails, use a default encoding as given with option **−D**, by the current locale, or 'latin1' if the locale is set to 'C', 'POSIX', or empty (in that order).

Note that the **groff** program supports a *GROFF_ENCODING* environment variable which is eventually expanded to option **−k**.

Byte Order Mark

The Unicode Standard defines character U+FEFF as the Byte Order Mark (BOM). On the other hand, value U+FFFE is guaranteed not be a Unicode character at all. This allows detection of the byte order within the data stream (either big-endian or little-endian), and the MIME encodings 'UTF-16' and 'UTF-32' mandate that the data stream starts with U+FEFF. Similarly, the data stream encoded as 'UTF-8' might start with a BOM (to ease the conversion from and to UTF-16 and UTF-32). In all cases, the byte order mark is *not* part of the data but part of the encoding protocol; in other words, **preconv**'s output doesn't

contain it.

Note that U+FEFF not at the start of the input data actually is emitted; it has then the meaning of a 'zero width no-break space' character – something not needed normally in **groff**.

Coding Tags

Editors which support more than a single character encoding need tags within the input files to mark the file's encoding. While it is possible to guess the right input encoding with the help of heuristic algorithms for data which represents a greater amount of a natural language, it is still just a guess. Additionally, all algorithms fail easily for input which is either too short or doesn't represent a natural language.

For these reasons, **preconv** supports the coding tag convention (with some restrictions) as used by **GNU Emacs** and **XEmacs** (and probably other programs too).

Coding tags in **GNU Emacs** and **XEmacs** are stored in so-called *File Variables*. **preconv** recognizes the following syntax form which must be put into a troff comment in the first or second line.

−*− *tag1*: *value1*; *tag2*: *value2*; . . . −*−

The only relevant tag for **preconv** is 'coding' which can take the values listed below. Here an example line which tells **Emacs** to edit a file in troff mode, and to use latin2 as its encoding.

```
.\" -*- mode: troff; coding: latin-2 -*-
```

The following list gives all MIME coding tags (either lowercase or uppercase) supported by **preconv**; this list is hard-coded in the source.

big5, cp1047, euc-jp, euc-kr, gb2312, iso-8859-1, iso-8859-2, iso-8859-5, iso-8859-7, iso-8859-9, iso-8859-13, iso-8859-15, koi8-r, us-ascii, utf-8, utf-16, utf-16be, utf-16le

In addition, the following hard-coded list of other tags is recognized which eventually map to values from the list above.

ascii, chinese-big5, chinese-euc, chinese-iso-8bit, cn-big5, cn-gb, cn-gb-2312, cp878, csascii, csisolatin1, cyrillic-iso-8bit, cyrillic-koi8, euc-china, euc-cn, euc-japan, euc-japan-1990, euc-korea, greek-iso-8bit, iso-10646/utf8, iso-10646/utf-8, iso-latin-1, iso-latin-2, iso-latin-5, iso-latin-7, iso-latin-9, japanese-euc, japanese-iso-8bit, jis8, koi8, korean-euc, korean-iso-8bit, latin-0, latin1, latin-1, latin-2, latin-5, latin-7, latin-9, mule-utf-8, mule-utf-16, mule-utf-16be, mule-utf-16-be, mule-utf-16be-with-signature, mule-utf-16le, mule-utf-16-le, mule-utf-16le-with-signature, utf8, utf-16-be, utf-16-be-with-signature, utf-16be-with-signature, utf-16-le, utf-16-le-with-signature, utf-16le-with-signature

Those tags are taken from **GNU Emacs** and **XEmacs**, together with some aliases. Trailing '-dos', '-unix', and '-mac' suffixes of coding tags (which give the end-of-line convention used in the file) are stripped off before the comparison with the above tags happens.

Iconv Issues

preconv by itself only supports three encodings: latin-1, cp1047, and UTF-8; all other encodings are passed to the **iconv** library functions. At compile time it is searched and checked for a valid **iconv** implementation; a call to 'preconv −−version' shows whether **iconv** is used.

BUGS

preconv doesn't support *local variable lists* yet. This is a different syntax form to specify local variables at the end of a file.

SEE ALSO

groff(1)
the **GNU Emacs** and **XEmacs** info pages