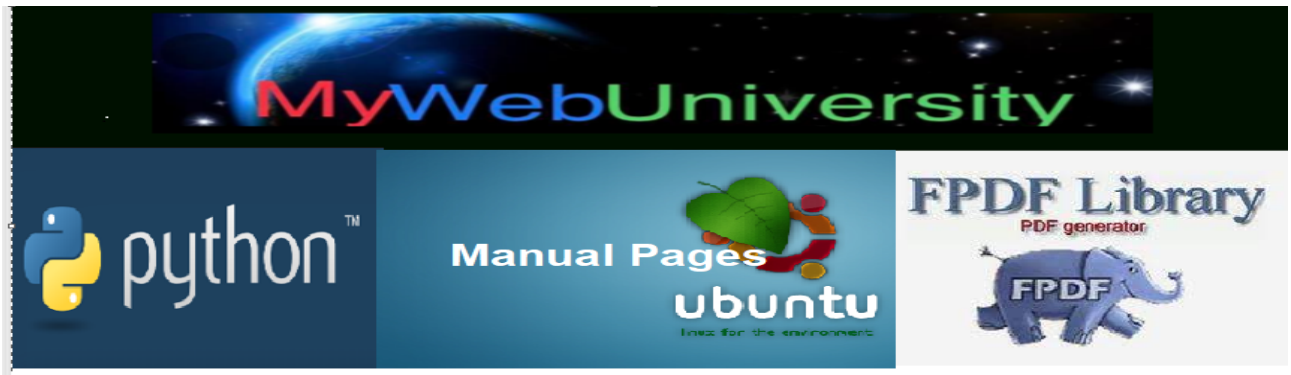

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'preconv.1'

$ man preconv.1

preconv(1) General Commands Manual preconv(1)

Name

preconv - prepare files for typesetting with groff

Synopsis

preconv [-dr] [-D fallback-encoding] [-e encoding] [file ...]

preconv -h

preconv --help

preconv -v

preconv --version

Description

preconv reads each file, converts its encoded characters to a form troff(1) can interpret, and sends the result to the standard output stream. Currently, this means that code points in the range 0?127 (in US-ASCII, ISO 8859, or Unicode) remain as-is and the remainder are converted to the groff special character form ?\[uXXXX]?, where XXXX is a hexadecimal number of four to six digits corresponding to a Unicode code point. By default, preconv also in? serts a roff .lf request at the beginning of each file, identifying it for the benefit of later processing (including diagnostic messages); the -r option suppresses this behavior.

In typical usage scenarios, preconv need not be run directly; instead it should be invoked with the -k or -K options of groff. If no file operands are given on the command line, or if file is ?-?, the standard input stream is read.

preconv tries to find the input encoding with the following algorithm, stopping at the first success.

1. If the input encoding has been explicitly specified with option -e, use it.

2. If the input starts with a Unicode Byte Order Mark, determine the encoding as UTF-8, UTF-16, or UTF-32 accordingly.

3. If the input stream is seekable, check the first and second input lines for a recognized GNU Emacs file-local variable identifying the character encoding, here referred to as the ?coding tag? for brevity. If found, use it.

4. If the input stream is seekable, and if the uchardet library is available on the system, use it to try to infer the encoding of the file.

5. If the -D option specifies an encoding, use it.

6. Use the encoding specified by the current locale (LC_CTYPE), unless the locale is ?C?, ?POSIX?, or empty, in which case assume Latin-1 (ISO 8859-1).

The coding tag and uchardet methods in the above procedure rely upon a seekable input stream; when preconv reads from a pipe, the stream is not seekable, and these detection methods are skipped. If character encoding detection of your input files is unreliable, arrange for one of the other methods to succeed by using preconv's -D or -e options, or by configuring your locale appropriately. groff also supports a GROFF_ENCODING environment variable, which can be overridden by its -K option. Valid values for (or parameters to) all of these are enumerated in the lists of recognized coding tags in the next subsection, and are further influenced by iconv library support.

Coding tags

Text editors that 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 heuristics that are reliable for a preponderance of natural language texts, they are not absolutely reliable. Heuristics can fail on inputs that are too short or don't represent a natural language.

Consequently, preconv supports the coding tag convention used by GNU Emacs (with some re? strictions). This notation appears in specially marked regions of an input file designated for ?file-local variables?.

preconv interprets the following syntax if it occurs in a roff comment in the first or sec? ond line of the input file. Both ?\"? and ?\#? comment forms are recognized, but the con? trol (or no-break control) character must be the default and must begin the line. Simi? larly, the escape character must be the default.

-*- [...;] coding: encoding[; ...] -*-

The only variable preconv interprets is ?coding?, which can take the values listed below.

The following list comprises all MIME ?charset? parameter values recognized, case-insensi? tively, by preconv.

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 list of other coding tags is recognized, each of which is mapped to an appropriate value 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

Trailing ?-dos?, ?-unix?, and ?-mac? suffixes on coding tags (which indicate the end-of-line convention used in the file) are disregarded for the purpose of comparison with the above tags.

iconv support

While preconv recognizes all of the coding tags listed above, it is capable on its own of interpreting only three encodings: Latin-1, code page 1047, and UTF-8. If iconv support is configured at compile time and available at run time, all others are passed to iconv library functions, which may recognize many additional encoding strings. The command ?preconv -v? discloses whether iconv support is configured.

The use of iconv means that characters in the input that encode invalid code points for that encoding may be dropped from the output stream or mapped to the Unicode replacement charac? ter (U+FFFD). Compare the following examples using the input ?caf?? (note the ?e? with an acute accent), which due to its short length challenges inference of the encoding used.

```
printf 'caf\351\n' | LC_ALL=en_US.UTF-8 preconv
printf 'caf\351\n' | preconv -e us-ascii
printf 'caf\351\n' | preconv -e latin-1
```

The fate of the accented ?e? differs in each case. In the first, uchardet fails to detect an encoding (though the library on your system may behave differently) and preconv falls back to the locale settings, where octal 351 starts an incomplete UTF-8 sequence and results in the Unicode replacement character. In the second, it is not a representable character in the declared input encoding of US-ASCII and is discarded by iconv. In the last, it is cor? rectly detected and mapped.

Limitations

preconv cannot perform any transformation on input that it cannot see. Examples include files that are interpolated by preprocessors that run subsequently, including soelim(1); files included by troff itself through ?so? and similar requests; and string definitions passed to troff through its -d command-line option.

preconv assumes that its input uses the default escape character, a backslash \, and writes special character escape sequences accordingly.

Options

-h and --help display a usage message, while -v and --version show version information; all exit afterward.

-d Emit debugging messages to the standard error stream.

-D fallback-encoding

Report fallback-encoding if all detection methods fail.

-e encoding

Skip detection and assume encoding; see groff's -K option.

-r Write files ?raw?; do not add .lf requests.

See also

groff(1), iconv(3), locale(7)