NAME

uconv – convert data from one encoding to another

SYNOPSIS

uconv [**–h**, **–?**, **––help**] [**–V**, **––version**] [**–s**, **––silent**] [**–v**, **––verbose**] [**–l**, **––list** | **–l**, **––list–code** *code* | **––default-code** | **–L**, **––list–transliterators**] [**––canon**] [**–x** *transliteration*] [**––to–callback** *callback* | **–c**] [**––from–callback** *callback* | **–i**] [**––callback** *callback*] [**––fallback** | **––no–fallback**] [**–b**, **––block–size** *size*] [**–f**, **––from–code** *encoding*] [**–t**, **––to–code** *encoding*] [**––add–signature**] [**––remove–signature**] [**–o**, **––output** *file*] [*file*...]

DESCRIPTION

uconv converts, or transcodes, each given *file* (or its standard input if no *file* is specified) from one *encoding* to another. The transcoding is done using Unicode as a pivot encoding (i.e. the data are first transcoded from their original encoding to Unicode, and then from Unicode to the destination encoding).

If an *encoding* is not specified or is **-**, the default encoding is used. Thus, calling **uconv** with no *encoding* provides an easy way to validate and sanitize data files for further consumption by tools requiring data in the default encoding.

When calling **uconv**, it is possible to specify callbacks that are used to handle invalid characters in the input, or characters that cannot be transcoded to the destination encoding. Some encodings, for example, offer a default substitution character that can be used to represent the occurrence of such characters in the input. Other callbacks offer a useful visual representation of the invalid data.

uconv can also run the specified *transliteration* on the transcoded data, in which case transliteration will happen as an intermediate step, after the data have been transcoded to Unicode. The *transliteration* can be either a list of semicolon-separated transliterator names, or an arbitrarily complex set of rules in the ICU transliteration rules format.

For transcoding purposes, **uconv** options are compatible with those of **iconv**(1), making it easy to replace it in scripts. It is not necessarily the case, however, that the encoding names used by **uconv** and ICU are the same as the ones used by **iconv**(1). Also, options that provide informational data, such as the **–l**, **––list** one offered by some **iconv**(1) variants such as GNU's, produce data in a slightly different and easier to parse format.

OPTIONS

–h, **–?**, **––help**
: Print help about usage and exit.

–V, **––version**
: Print the version of **uconv** and exit.

–s, **––silent**
: Suppress messages during execution.

–v, **––verbose**
: Display extra informative messages during execution.

–l, **––list**
: List all the available encodings and exit.

–l, **––list–code** *code*
: List only the *code* encoding and exit. If *code* is not a proper encoding, exit with an error.

––default-code
: List only the name of the default encoding and exit.

–L, **––list–transliterators**
: List all the available transliterators and exit.

–-canon
: If used with **–l**, **––list** or **––default-code**, the list of encodings is produced in a format compatible with **convrtrs.txt**(5). If used with **–L**, **––list–transliterators**, print only one transliterator name

per line.

−x *transliteration*
Run the given *transliteration* on the transcoded Unicode data, and use the transliterated data as input for the transcoding to the destination encoding.

−−to−callback *callback*
Use *callback* to handle characters that cannot be transcoded to the destination encoding. See section **CALLBACKS** for details on valid callbacks.

−c Omit invalid characters from the output. Same as **−−to−callback skip**.

−−from−callback *callback*
Use *callback* to handle characters that cannot be transcoded from the original encoding. See section **CALLBACKS** for details on valid callbacks.

−i Ignore invalid sequences in the input. Same as **−−from−callback skip**.

−−callback *callback*
Use *callback* to handle both characters that cannot be transcoded from the original encoding and characters that cannot be transcoded to the destination encoding. See section **CALLBACKS** for details on valid callbacks.

−−fallback
Use the fallback mapping when transcoding from Unicode to the destination encoding.

−−no−fallback
Do not use the fallback mapping when transcoding from Unicode to the destination encoding. This is the default.

−b, **−−block−size** *size*
Read input in blocks of *size* bytes at a time. The default block size is 4096.

−f, **−−from−code** *encoding*
Set the original encoding of the data to *encoding*.

−t, **−−to−code** *encoding*
Transcode the data to *encoding*.

−−add−signature
Add a U+FEFF Unicode signature character (BOM) if the output charset supports it and does not add one anyway.

−−remove−signature
Remove a U+FEFF Unicode signature character (BOM).

−o, **−−output** *file*
Write the transcoded data to *file*.

CALLBACKS

uconv supports specifying callbacks to handle invalid data. Callbacks can be set for both directions of transcoding: from the original encoding to Unicode, with the **−−from−callback** option, and from Unicode to the destination encoding, with the **−−to−callback** option.

The following is a list of valid *callback* names, along with a description of their behavior. The list of callbacks actually supported by **uconv** is displayed when it is called with **−h**, **−−help**.

substitute Write the encoding's substitute sequence, or the Unicode replacement character **U+FFFD** when transcoding to Unicode.

skip Ignore the invalid data.

stop Stop with an error when encountering invalid data. This is the default callback.

escape Same as **escape-icu**.

escape-icu Replace the missing characters with a string of the format **%U***hhhh* for plane 0 characters, and **%U***hhhh***%U***hhhh* for planes 1 and above characters, where *hhhh* is the hexadecimal value of one of the UTF-16 code units representing the character. Characters from planes 1 and above are written as a pair of UTF-16 surrogate code units.

escape-java Replace the missing characters with a string of the format **\u***hhhh* for plane 0 characters, and **\u***hhhh***\u***hhhh* for planes 1 and above characters, where *hhhh* is the hexadecimal value of one of the UTF-16 code units representing the character. Characters from planes 1 and above are written as a pair of UTF-16 surrogate code units.

escape-c Replace the missing characters with a string of the format **\u***hhhh* for plane 0 characters, and **\U***hhhhhhhh* for planes 1 and above characters, where *hhhh* and *hhhhhhhh* are the hexadecimal values of the Unicode codepoint.

escape-xml Same as **escape-xml-hex**.

escape-xml-hex Replace the missing characters with a string of the format **&#x***hhhh***;**, where *hhhh* is the hexadecimal value of the Unicode codepoint.

escape-xml-dec Replace the missing characters with a string of the format **&#***nnnn***;**, where *nnnn* is the decimal value of the Unicode codepoint.

escape-unicode Replace the missing characters with a string of the format **{U+***hhhh***}**, where *hhhh* is the hexadecimal value of the Unicode codepoint. That hexadecimal string is of variable length and can use from 4 to 6 digits. This is the format universally used to denote a Unicode codepoint in the literature, delimited by curly braces for easy recognition of those substitutions in the output.

EXAMPLES

Convert data from a given *encoding* to the platform encoding:

$ **uconv –f** *encoding*

Check if a *file* contains valid data for a given *encoding*:

$ **uconv –f** *encoding* **–c** *file* **>/dev/null**

Convert a UTF-8 *file* to a given *encoding* and ensure that the resulting text is good for any version of HTML:

$ **uconv –f utf-8 –t** *encoding* ****
––callback escape-xml-dec *file*

Display the names of the Unicode code points in a UTF-file:

$ **uconv –f utf-8 –x any-name** *file*

Print the name of a Unicode code point whose value is known (**U+30AB** in this example):

$ **echo '\u30ab' | uconv –x 'hex-any; any-name'; echo**
{KATAKANA LETTER KA}{LINE FEED}
$

(The names are delimited by curly braces. Also, the name of the line terminator is also displayed.)

Normalize UTF-8 data using Unicode NFKC, remove all control characters, and map Katakana to Hiragana:

$ **uconv –f utf-8 –t utf-8 **
–x '::nfkc; [:Cc:] >; ::katakana-hiragana;'

CAVEATS AND BUGS

uconv does report errors as occurring at the first invalid byte encountered. This may be confusing to users of GNU **iconv**(1), which reports errors as occurring at the first byte of an invalid sequence. For multi-byte character sets or encodings, this means that **uconv** error positions may be at a later offset in the input stream than would be the case with GNU **iconv**(1).

The reporting of error positions when a transliterator is used may be inaccurate or unavailable, in which case **uconv** will report the offset in the output stream at which the error occurred.

AUTHORS

Jonas Utterstroem
Yves Arrouye

VERSION

63.2

COPYRIGHT

SEE ALSO

iconv(1)