NAME

enc2xs -- Perl Encode Module Generator

SYNOPSIS

```
enc2xs -[options]
enc2xs -M ModName mapfiles...
enc2xs -C
```

DESCRIPTION

enc2xs builds a Perl extension for use by Encode from either Unicode Character Mapping files (.ucm) or Tcl Encoding Files (.enc). Besides being used internally during the build process of the Encode module, you can use *enc2xs* to add your own encoding to perl. No knowledge of XS is necessary.

Quick Guide

If you want to know as little about Perl as possible but need to add a new encoding, just read this chapter and forget the rest.

0. Have a .ucm file ready. You can get it from somewhere or you can write your own from scratch or you can grab one from the Encode distribution and customize it. For the UCM format, see the next Chapter. In the example below, I'll call my theoretical encoding myascii, defined in *my.ucm*. `$` is a shell prompt.

   ```
   $ ls -F
   my.ucm
   ```

1. Issue a command as follows;

   ```
   $ enc2xs -M My my.ucm
   generating Makefile.PL
   generating My.pm
   generating README
   generating Changes
   ```

   Now take a look at your current directory. It should look like this.

   ```
   $ ls -F
   Makefile.PL   My.pm         my.ucm        t/
   ```

   The following files were created.

   ```
   Makefile.PL - MakeMaker script
   My.pm       - Encode submodule
   t/My.t      - test file
   ```

   1.1. If you want *.ucm installed together with the modules, do as follows;

   ```
   $ mkdir Encode
   $ mv *.ucm Encode
   $ enc2xs -M My Encode/*ucm
   ```

2. Edit the files generated. You don't have to if you have no time AND no intention to give it to someone else. But it is a good idea to edit the pod and to add more tests.

3. Now issue a command all Perl Mongers love:

   ```
   $ perl Makefile.PL
   Writing Makefile for Encode::My
   ```

4. Now all you have to do is make.

```
$ make
cp My.pm blib/lib/Encode/My.pm
/usr/local/bin/perl /usr/local/bin/enc2xs -Q -O \
  -o encode_t.c -f encode_t.fnm
Reading myascii (myascii)
Writing compiled form
128 bytes in string tables
384 bytes (75%) saved spotting duplicates
1 bytes (0.775%) saved using substrings
....
chmod 644 blib/arch/auto/Encode/My/My.bs
$
```

The time it takes varies depending on how fast your machine is and how large your encoding is. Unless you are working on something big like euc-tw, it won't take too long.

5. You can "make install" already but you should test first.

```
$ make test
PERL_DL_NONLAZY=1 /usr/local/bin/perl -Iblib/arch -Iblib/lib \
  -e 'use Test::Harness  qw(&runtests $verbose); \
  $verbose=0; runtests @ARGV;' t/*.t
t/My....ok
All tests successful.
Files=1, Tests=2,  0 wallclock secs
 ( 0.09 cusr + 0.01 csys = 0.09 CPU)
```

6. If you are content with the test result, just "make install"

7. If you want to add your encoding to Encode's demand-loading list (so you don't have to "use Encode::YourEncoding"), run

```
enc2xs -C
```

to update Encode::ConfigLocal, a module that controls local settings. After that, "use Encode;" is enough to load your encodings on demand.

The Unicode Character Map

Encode uses the Unicode Character Map (UCM) format for source character mappings. This format is used by IBM's ICU package and was adopted by Nick Ing-Simmons for use with the Encode module. Since UCM is more flexible than Tcl's Encoding Map and far more user-friendly, this is the recommended format for Encode now.

A UCM file looks like this.

```
#
# Comments
#
<code_set_name> "US-ascii" # Required
<code_set_alias> "ascii"   # Optional
<mb_cur_min> 1             # Required; usually 1
<mb_cur_max> 1             # Max. # of bytes/char
<subchar> \x3F             # Substitution char
#
CHARMAP
<U0000> \x00 |0 # <control>
<U0001> \x01 |0 # <control>
<U0002> \x02 |0 # <control>
....
<U007C> \x7C |0 # VERTICAL LINE
```

```
<U007D> \x7D |0 # RIGHT CURLY BRACKET
<U007E> \x7E |0 # TILDE
<U007F> \x7F |0 # <control>
END CHARMAP
```

- Anything that follows # is treated as a comment.
- The header section continues until a line containing the word CHARMAP. This section has a form of *<keyword> value*, one pair per line. Strings used as values must be quoted. Barewords are treated as numbers. *\xXX* represents a byte.

  Most of the keywords are self-explanatory. *subchar* means substitution character, not subcharacter. When you decode a Unicode sequence to this encoding but no matching character is found, the byte sequence defined here will be used. For most cases, the value here is \x3F; in ASCII, this is a question mark.
- CHARMAP starts the character map section. Each line has a form as follows:

  ```
  <UXXXX> \xXX.. |0 # comment
    ^      ^     ^
    |      |     +- Fallback flag
    |      +------- Encoded byte sequence
    +-------------- Unicode Character ID in hex
  ```

  The format is roughly the same as a header section except for the fallback flag: | followed by 0..3. The meaning of the possible values is as follows:

  |0 Round trip safe. A character decoded to Unicode encodes back to the same byte sequence. Most characters have this flag.

  |1 Fallback for unicode −> encoding. When seen, enc2xs adds this character for the encode map only.

  |2 Skip sub-char mapping should there be no code point.

  |3 Fallback for encoding −> unicode. When seen, enc2xs adds this character for the decode map only.
- And finally, END OF CHARMAP ends the section.

When you are manually creating a UCM file, you should copy ascii.ucm or an existing encoding which is close to yours, rather than write your own from scratch.

When you do so, make sure you leave at least **U0000** to **U0020** as is, unless your environment is EBCDIC.

CAVEAT: not all features in UCM are implemented. For example, icu:state is not used. Because of that, you need to write a perl module if you want to support algorithmical encodings, notably the ISO−2022 series. Such modules include Encode::JP::2022_JP, Encode::KR::2022_KR, and Encode::TW::HZ.

Coping with duplicate mappings

When you create a map, you SHOULD make your mappings round-trip safe. That is, `encode('your-encoding', decode('your-encoding', $data)) eq $data` stands for all characters that are marked as `|0`. Here is how to make sure:

- Sort your map in Unicode order.
- When you have a duplicate entry, mark either one with '|1' or '|3'.
- And make sure the '|1' or '|3' entry FOLLOWS the '|0' entry.

Here is an example from big5−eten.

```
<U2550> \xF9\xF9 |0
<U2550> \xA2\xA4 |3
```

Internally Encoding −> Unicode and Unicode −> Encoding Map looks like this;

```
E to U               U to E
---------------------------------------
\xF9\xF9 => U2550    U2550 => \xF9\xF9
\xA2\xA4 => U2550
```

So it is round-trip safe for \xF9\xF9. But if the line above is upside down, here is what happens.

```
E to U               U to E
---------------------------------------
\xA2\xA4 => U2550    U2550 => \xF9\xF9
(\xF9\xF9 => U2550 is now overwritten!)
```

The Encode package comes with *ucmlint*, a crude but sufficient utility to check the integrity of a UCM file. Check under the Encode/bin directory for this.

When in doubt, you can use *ucmsort*, yet another utility under Encode/bin directory.

Bookmarks

- ICU Home Page <http://www.icu−project.org/>
- ICU Character Mapping Tables <http://site.icu−project.org/charts/charset>
- ICU:Conversion Data <http://www.icu−project.org/userguide/conversion−data.html>

SEE ALSO

Encode, perlmod, perlpod