NAME

Encode::Guess -- Guesses encoding from data

SYNOPSIS

```
  # if you are sure $data won't contain anything bogus

  use Encode;
  use Encode::Guess qw/euc-jp shiftjis 7bit-jis/;
  my $utf8 = decode("Guess", $data);
  my $data = encode("Guess", $utf8);   # this doesn't work!

  # more elaborate way
  use Encode::Guess;
  my $enc = guess_encoding($data, qw/euc-jp shiftjis 7bit-jis/);
  ref($enc) or die "Can't guess: $enc"; # trap error this way
  $utf8 = $enc->decode($data);
  # or
  $utf8 = decode($enc->name, $data)
```

ABSTRACT

Encode::Guess enables you to guess in what encoding a given data is encoded, or at least tries to.

DESCRIPTION

By default, it checks only ascii, utf8 and UTF-16/32 with BOM.

```
  use Encode::Guess; # ascii/utf8/BOMed UTF
```

To use it more practically, you have to give the names of encodings to check (*suspects* as follows). The name of suspects can either be canonical names or aliases.

CAVEAT: Unlike UTF-(16|32), BOM in utf8 is NOT AUTOMATICALLY STRIPPED.

```
 # tries all major Japanese Encodings as well
  use Encode::Guess qw/euc-jp shiftjis 7bit-jis/;
```

If the `$Encode::Guess::NoUTFAutoGuess` variable is set to a true value, no heuristics will be applied to UTF8/16/32, and the result will be limited to the suspects and `ascii`.

Encode::Guess->set_suspects

You can also change the internal suspects list via `set_suspects` method.

```
  use Encode::Guess;
  Encode::Guess->set_suspects(qw/euc-jp shiftjis 7bit-jis/);
```

Encode::Guess->add_suspects

Or you can use `add_suspects` method. The difference is that `set_suspects` flushes the current suspects list while `add_suspects` adds.

```
  use Encode::Guess;
  Encode::Guess->add_suspects(qw/euc-jp shiftjis 7bit-jis/);
  # now the suspects are euc-jp,shiftjis,7bit-jis, AND
  # euc-kr,euc-cn, and big5-eten
  Encode::Guess->add_suspects(qw/euc-kr euc-cn big5-eten/);
```

Encode::decode("Guess" ...)

When you are content with suspects list, you can now

```
my $utf8 = Encode::decode("Guess", $data);
```

Encode::Guess->guess($data)

But it will croak if:

- Two or more suspects remain
- No suspects left

So you should instead try this;

```
my $decoder = Encode::Guess->guess($data);
```

On success, $decoder is an object that is documented in *Encode::Encoding*. So you can now do this;

```
my $utf8 = $decoder->decode($data);
```

On failure, $decoder now contains an error message so the whole thing would be as follows;

```
my $decoder = Encode::Guess->guess($data);
die $decoder unless ref($decoder);
my $utf8 = $decoder->decode($data);
```

guess_encoding($data, [, *list of suspects*])

You can also try `guess_encoding` function which is exported by default. It takes $data to check and it also takes the list of suspects by option. The optional suspect list is *not reflected* to the internal suspects list.

```
my $decoder = guess_encoding($data, qw/euc-jp euc-kr euc-cn/);
die $decoder unless ref($decoder);
my $utf8 = $decoder->decode($data);
# check only ascii, utf8 and UTF-(16|32) with BOM
my $decoder = guess_encoding($data);
```

CAVEATS

- Because of the algorithm used, ISO-8859 series and other single-byte encodings do not work well unless either one of ISO-8859 is the only one suspect (besides ascii and utf8).

  ```
  use Encode::Guess;
  # perhaps ok
  my $decoder = guess_encoding($data, 'latin1');
  # definitely NOT ok
  my $decoder = guess_encoding($data, qw/latin1 greek/);
  ```

  The reason is that Encode::Guess guesses encoding by trial and error. It first splits $data into lines and tries to decode the line for each suspect. It keeps it going until all but one encoding is eliminated out of suspects list. ISO-8859 series is just too successful for most cases (because it fills almost all code points in \x00-\xff).

- Do not mix national standard encodings and the corresponding vendor encodings.

  ```
  # a very bad idea
  my $decoder
     = guess_encoding($data, qw/shiftjis MacJapanese cp932/);
  ```

  The reason is that vendor encoding is usually a superset of national standard so it becomes too ambiguous for most cases.

- On the other hand, mixing various national standard encodings automagically works unless $data is too short to allow for guessing.

```
# This is ok if $data is long enough
my $decoder =
 guess_encoding($data, qw/euc-cn
                          euc-jp shiftjis 7bit-jis
                          euc-kr
                          big5-eten/);
```

- DO NOT PUT TOO MANY SUSPECTS! Don't you try something like this!

```
my $decoder = guess_encoding($data,
                             Encode->encodings(":all"));
```

It is, after all, just a guess. You should alway be explicit when it comes to encodings. But there are some, especially Japanese, environment that guess-coding is a must. Use this module with care.

TO DO

Encode::Guess does not work on EBCDIC platforms.

SEE ALSO

Encode, *Encode::Encoding*