NAME

MIME::Base64 - Encoding and decoding of base64 strings

SYNOPSIS

```
use MIME::Base64;

$encoded = encode_base64('Aladdin:open sesame');
$decoded = decode_base64($encoded);
```

DESCRIPTION

This module provides functions to encode and decode strings into and from the base64 encoding specified in RFC 2045 - *MIME (Multipurpose Internet Mail Extensions)*. The base64 encoding is designed to represent arbitrary sequences of octets in a form that need not be humanly readable. A 65-character subset ([A-Za-z0-9+/=]) of US-ASCII is used, enabling 6 bits to be represented per printable character.

The following functions are provided:

encode_base64($str)

encode_base64($str, $eol);

> Encode data by calling the encode_base64() function. The first argument is the string to encode. The second argument is the line-ending sequence to use. It is optional and defaults to "\n". The returned encoded string is broken into lines of no more than 76 characters each and it will end with $eol unless it is empty. Pass an empty string as second argument if you do not want the encoded string to be broken into lines.

decode_base64($str)

> Decode a base64 string by calling the decode_base64() function. This function takes a single argument which is the string to decode and returns the decoded data.
>
> Any character not part of the 65-character base64 subset is silently ignored. Characters occurring after a '=' padding character are never decoded.
>
> If the length of the string to decode, after ignoring non-base64 chars, is not a multiple of 4 or if padding occurs too early, then a warning is generated if perl is running under `-w`.

If you prefer not to import these routines into your namespace, you can call them as:

```
use MIME::Base64 ();
$encoded = MIME::Base64::encode($decoded);
$decoded = MIME::Base64::decode($encoded);
```

DIAGNOSTICS

The following warnings can be generated if perl is invoked with the `-w` switch:

Premature end of base64 data

> The number of characters to decode is not a multiple of 4. Legal base64 data should be padded with one or two "=" characters to make its length a multiple of 4. The decoded result will be the same whether the padding is present or not.

Premature padding of base64 data

> The '=' padding character occurs as the first or second character in a base64 quartet.

The following exception can be raised:

Wide character in subroutine entry

The string passed to encode_base64() contains characters with code above 255. The base64 encoding is only defined for single-byte characters. Use the Encode module to select the byte encoding you want.

EXAMPLES

If you want to encode a large file, you should encode it in chunks that are a multiple of 57 bytes. This ensures that the base64 lines line up and that you do not end up with padding in the middle. 57 bytes of data fills one complete base64 line (76 == 57*4/3):

```
use MIME::Base64 qw(encode_base64);

open(FILE, "/var/log/wtmp") or die "$!";
while (read(FILE, $buf, 60*57)) {
    print encode_base64($buf);
}
```

or if you know you have enough memory

```
use MIME::Base64 qw(encode_base64);
local($/) = undef;  # slurp
print encode_base64(<STDIN>);
```

The same approach as a command line:

```
perl -MMIME::Base64 -0777 -ne 'print encode_base64($_)' <file
```

Decoding does not need slurp mode if every line contains a multiple of four base64 chars:

```
perl -MMIME::Base64 -ne 'print decode_base64($_)' <file
```

Perl v5.8 and better allow extended Unicode characters in strings. Such strings cannot be encoded directly, as the base64 encoding is only defined for single-byte characters. The solution is to use the Encode module to select the byte encoding you want. For example:

```
use MIME::Base64 qw(encode_base64);
use Encode qw(encode);

$encoded = encode_base64(encode("UTF-8", "\x{FFFF}\n"));
print $encoded;
```

COPYRIGHT

Distantly based on LWP::Base64 written by Martijn Koster <m.koster@nexor.co.uk> and Joerg Reichelt <j.reichelt@nexor.co.uk> and code posted to comp.lang.perl <3pd2lp$6gf@wsinti07.win.tue.nl> by Hans Mulder <hansm@wsinti07.win.tue.nl>

The XS implementation uses code from metamail. Copyright 1991 Bell Communications Research, Inc. (Bellcore)

SEE ALSO

MIME::QuotedPrint