Thomas Fahle

How To? String::Dump
Was genau ist in einem String enthalten?

Manchmal steckt in einem String nicht das drin, was man erwartet, z.B. nicht druckbare oder UTF-Zeichen.

String::Dump - `Dump strings of characters (or bytes) for printing and debugging` von `Nick Patch` vereinfacht das Debuggen solcher Probleme erheblich, indem es jedes Byte und jeden Codepoint eines Strings ausgibt.

Funktionen

String::Dump stellt **sechs** unterschiedliche Ausgabeformate über folgende Funktionen zur Verfügung.

- dump_hex($string): Hexadecimal (base 16)
- dump_dec($string): Decimal (base 10)
- dump_oct($string): Octal (base 8)
- dump_bin($string): Binary (base 2)
- dump_names($string): Unicode character name
- dump_codes($string): Unicode code point

```
#!/usr/bin/perl
use strict;
use warnings;

use utf8;

use String::Dump qw( :all );

my $string = 'Føø Bār';

print dump_hex($string), "\n";

print dump_dec($string), "\n";

print dump_oct($string), "\n";

print dump_bin($string), "\n";

print dump_names($string), "\n";

print dump_codes($string), "\n";
```

Listing 1

Beispiel

Statt vieler Worte ein einfaches Beispiel in Listing 1.

Das Programm erzeugt folgende Ausgabe:

```
66 F8 F8 20 62 101 72

102 248 248 32 98 257 114

146 370 370 40 142 401 162

1100110 11111000 11111000 100000 1100010
100000001 1110010

LATIN CAPITAL LETTER F, LATIN SMALL LETTER
O WITH STROKE, LATIN SMALL LETTER O WITH
STROKE, SPACE, LATIN SMALL LETTER B,
LATIN SMALL LETTER A WITH MACRON, LATIN
SMALL LETTER R

U+0066 U+00F8 U+00F8 U+0020 U+0062
U+0101 U+0072
```

```
#!/usr/bin/perl
use strict;
use warnings;

use String::Dump qw( :all );

use utf8;

my $string = 'Føø Bār';
print dump_hex($string),    "\n";
print dump_names($string), "\n";

{
  # Der gleiche String ohne
  # Pragma utf8
  no utf8;
  my $string = 'Føø Bār';
  print dump_hex($string),    "\n";
  print dump_names($string), "\n";
}
```

Listing 2

Debugging-Tipps

Algemein gilt: Wenn man UTF-8 debuggen möchte, sollten Strings auch UTF8-kodiert sein.

Literale Strings im Quelltext

Bei der Verwendung von Unicode-Strings im Quelltext sollte das Pragma `utf8` eingeschaltet sein (siehe Listing 2).

Das Programm erzeugt folgende Ausgabe:

```
46 F8 F8 20 42 101 72

LATIN CAPITAL LETTER F, LATIN SMALL LETTER
O WITH STROKE, LATIN SMALL LETTER O WITH
STROKE, SPACE, LATIN CAPITAL LETTER B,
LATIN SMALL LETTER A WITH MACRON, LATIN
SMALL LETTER R

46 C3 B8 C3 B8 20 42 C4 81 72

LATIN CAPITAL LETTER F, LATIN CAPITAL
LETTER A WITH TILDE, CEDILLA, LATIN
CAPITAL LETTER A WITH TILDE, CEDILLA,
SPACE, LATIN CAPITAL LETTER B, LATIN CAPITAL
LETTER A WITH DIAERESIS, HIGH OCTET PRESET,
LATIN SMALL LETTER R
```

Kommandozeilenargumente und Dateihandles

Auch hier sollte der Entwickler (m/w) sicherstellen, dass **alle** Eingaben UTF-8-kodiert sind oder eben nicht.

Für UTF-8 bietet sich die Verwendung des CPAN-Moduls `utf8::all` an, das alle Dateihandles und `@ARGV` automatisch UTF-8-kodiert.

Sonstige Eingabequellen

Strings aus Quellen wie Netzwerksockets, Formulardaten usw. sollten ggf. vorher mit `Encode::decode` von UTF8 dekodiert werden.

```
use Encode qw( decode );

print dump_hex(
           decode('UTF-8', $string)
      ) , "\n";
```

Kommandozeilentool dumpstr

Zum Debuggen auf der Kommandozeile eignet sich das mitgelieferte Tool `dumpstr`.

Der Schalter `-m` bzw. `--mode` legt das Ausgabeformat (hex, dec, oct, bin, names, codes) fest.

```
$ dumpstr -m names 'Føø Bār'
```