Herbert Breunung

Sereal - sichert alle Daten - schnell und kompakt

Wenn der Benutzer das Licht ausknipst, fallen alle Siliziumchips in den tiefen Schlaf des Vergessens. Damit die ganze Rechnerei aber nicht vergebens war, speichern Programme ihre Ergebnisse auf Festplatte oder Flash-Speicher. Dafür gibt es eine sehr große Auswahl an Formaten. Voriges Jahr fügten die Entwickler der Hotelreservierungsplattform *Booking.com* sogar noch eines dazu. Denn das Programm einfach nur am nächsten Tag (oder auf einem anderen Rechner) fortzufahren, war bisher (zumindest im Land der Dromedare) nicht so einfach.

Nichts gegen YAML, JSON, Storable oder Data::Dumper - aber was diese Module interessiert, sind lediglich Daten im Sinne von Zahlen, Text, Arrays und Hashes. YAML kann etwas mehr, da der bekannte Perl-Haudegen Ingy daran beteiligt war. Es kann auch Perl-Objekte, *typeglob* und Referenzen serialisieren (in eine Form bringen, die man in einem Schwung in eine Datei legen kann und aus der sich später alles wieder herstellen lässt). Doch wer Perl näher kennt, weiß, dass es zwischen Himmel und Hölle noch weit mehr Dinge gibt, wie etwa schwache Referenzen. Eines der Ziele von Sereal war es, wirklich alles realitätsgetreu speichern und wiedergeben zu können, selbst bereits kompilierte reguläre Ausdrücke.

```
use v5.14;
use Sereal qw/encode_sereal decode_sereal/;

my $rxout = encode_sereal(qr/d(.+)r/);
say decode_sereal($rxout);
say ref decode_sereal($rxout);
```

ergibt:

```
(?^:d(.+)r)
Regexp
```

Sereals grundsätzliche API ist denkbar einfach und funktioniert nach dem bekannten freeze/thaw - Prinzip: je eine Funktion pro Richtung der Umwandlung. Die zweite Ausgabe zeigt: es ist eine Regex, welche im beim print/say der ersten Ausgabe ihren Inhalt präsentiert. Jede Datenstruktur, die Perl im Speicher halten kann, lässt sich nun festhalten - keine Bedenken mehr, ob ein Attribut etwas Problematisches enthielt. Für so eine Arbeit muss man Ahnung von Perl's internen Speicherstrukturen haben. Die ist zweifellos bei Hauptautor Steffen Müller (tsee) zu finden, der bereits seit Jahren sehr aktiv in der p5p ist. Auch Helfer wie Yves Orton (demerhq), der den Großteil des 5.10er Regex-Umbaus gestemmt hat, oder der ehemalige Pumpking Rafaël Garcia-Suarez besitzen Kaliber, was am klaren und gut strukturierten Quellcode deutlich sichtbar ist.

Nicht um die Begrenzungen des Serialisierers herumprogrammieren zu müssen war Ziel von Sereal. Ein anders war Geschwindigkeit und eine kompakte Ausgabe. Und obwohl ersteres Priorität hatte, scheint es in beiden Disziplinen sämtliche andere CPAN-Module zu überflügeln, auch den bisherigen Rekordhalter `Data::MessagePack`. Dies war natürlich nur mit einem durchdachten, binären Format möglich, welches mit C-Funktionen erzeugt und gelesen wird. Das Perlmodul enthält nur die XS-Schnittstelle und Dokumentation. Wer mehr Kompression zu Lasten der Rechenzeit bevorzugt, kann Google's Snappy-Algorithmus aktivieren.

```
encode_sereal($data, {snappy => 1});
```

Eine Reihe weiterer Optionen zeigen, dass Sereal ein robustes Werkzeug für schwere Einsätze sein möchte.

Version 2 - weiter gehts!

Diesen Herbst, noch vor seinem ersten Geburtstag, erfuhr das Format eine kleinere Neugestaltung - mit voller Vorwärts- und Rückwärtskompatibilität. Es ging hauptsächlich darum, zwischen die Datenblöcke Metainformationen

abzulegen. Dadurch kann man auch schnell auf wesentliche Informationen zugreifen, ohne den gesamten Datensatz entpacken zu müssen.

Die Praxistauglichkeit scheint sich herumzusprechen, denn im Github-Repository *http://github.com/Sereal/Sereal* finden sich bereits Implementationen für sieben weitere Sprachen. Dies gibt Anlass zur Hoffnung, dass Sereal breit akzeptierter Standard wird, wie bereits YAML oder TAP.

Links

- http://blog.booking.com/sereal-a-binary-data-serialization-format.html
- http://blog.booking.com/the-next-sereal-is-coming.html