Crashkurs Perl

Vortrag: *Jens Ohlig* <jo@devcon.net>

Bericht: *Jens Ohlig* <jo@devcon.net>

Knapp eine Stunde Perl -- nicht gerade viel Zeit, um ein Lieblingsspielzeug der Hackergemeinde vorzustellen. Die Programmiersprache Perl (Practical Extraction and Report Language) ist das ideale Werkzeug zur Verwurstung von Texten und durch die enge Anbindung an das Betriebsystem bietet sie ein ganz neues, entspanntes Gefühl der Unix-Programmierung.

Perl zeichnet sich zunächst durch interessante Datentypen aus. Die einfachen (skalaren) Datentypen sind durch ein vorgestelltes $-Zeichen markiert. Die Variable $var kann Zahlen enthalten, Zeichen oder auch Strings (Zeichenketten). Kein unappetitliches Hantieren mit Pointern, keine Deklaration der Variablen, bevor man sich an die eigentliche Lösung des Problems macht; Perl kümmert sich einfach darum, daß diese skalare Variablen funktionieren.

Wie es sich für eine moderne Programmiersprache gehört, gibt es neben einfachen Datentypen aber noch Listen. In Perl gibt es sie in zwei Geschmacksrichtungen: Zunächst die einfache Liste, gekennzeichnet durch ein vorgestelltes @, so daß eine Liste z.B. @var heissen könnte. Mit diesen Listen kann man die üblichen Operationen vornehmen: einzelne Elemente herausziehen, die ganze Liste permutieren, sortieren und vieles mehr.

Interessant ist hier, daß Perl quasi mitdenkt und Variablenzuweisungen abhängig vom Kontext vornehmen kann:

```
$var = ;
```

liest eine Zeile von der Standardeingabe ein und weist sie der Variablen $var zu.

```
@var = ;
```

liest gleich die ganze Datei ein und legt die einzelnen Zeilen als Elemente in der Liste ab. Ein Teil von Perl, der bei praktisch jeder Problemstellung ins Spiel kommt, sind die sogenannten regulären Ausdrücke. Mit dieser Minisprache innerhalb von Perl kann man Regeln für Zeichenmuster beschreiben.

```
/Hallo, Welt/
```

trifft auf alle Strings zu, in denen die Folge "Hallo, Welt" enthalten ist. Wenn man sich nicht so sicher ist, wie der String aussieht, kann man die Regel auch unschärfer formulieren:

```
/Hallo.*/
```

trifft auf einen String zu, in dem auf "Hallo" beliebig viele Zeichen folgen, möglicherweise auch gar keine. Hier steht der Punkt für ein beliebiges Zeichen und der Stern für beliebig viele (auch gar keine) Folgezeichen. Analog steht

```
/.+/
```

für ein Zeichen, beliebig oft, aber mindestens einmal wiederholt. Oft will man aber mit Teilen der Zeichenmuster weiterarbeiten.

Mit Klammern innerhalb des regulären Ausdrucks kann man hier Subausdrücke einfangen.

```
/(Hallo) (Welt)/
```

schnappt sich die Teile "Hallo" und "Welt" und weist sie stillschweigend den Perl-internen Skalar-Variablen $1 und $2 zu, die man dann beliebig weiterbearbeiten kann. Mit der Version 5 von Perl gab es eine Neuerung, die das Leben noch angenehmer macht: Module. Mit dem Modul:

```
Net::Telnet
```

etwa wird es kinderleicht, Clients im Internet zu schreiben, während beim klassischen Modell der Sockets-Programmierung (etwa unter C) reichlich Gefrickel angesagt ist. Ein anderes Beispiel wäre das Modul "Shell". Einfach das Modul einbinden und schon hat Perl reichlich neue Funktionen. Durch sogenanntes Method-overloading wird jede Funktion, die Perl nicht kennt, einfach an die Shell weitergeleitet.

Kurz: Perl bietet eine Menge Möglichkeiten für den Hacker, ohne daß man sich mit Standärgernissen wie Variablen deklarieren, Speicher allozieren und freigeben oder ähnlichen Widrigkeiten abgeben muß. Oder, um es in den Worten von Perl-Entwickler Larry Wall zu sagen: "Perl makes easy things easy and hard things possible".