Die Skriptsprache

Ein TeamOS/2-Vortrag
von Wolfram Körner
koerner@informatik.uni-wuerzburg.de
(Würzburg, 3.4.1997)

Einführung

- **perl** = practical extraction and report language
- erfunden 1986 von Larry Wall
- aktuelle Version: ca. 5.0.0.3
- Portierungen: Unixe, OS/2, Win32, DOS-X...
- Typenlose Skriptsprache (String=Float=Integer)
  - keine Compilierung >>> schnellere Entwicklung
  - Quasi-Compile zur Laufzeit >>> Geschwindigkeit
  - Speichermanagement durch perl
  - Nachteil: Langsamer als Voll-Compilierung
- Frei, Kostenlos, Umsonst !

Überblick

- Variablentypen und automatische Konversion
- Programmfluß-Steuerung
- Dateihandling
- Reguläre Ausdrücke

- `while (<>) {print ;} # Kurz !!!`

Syntax-Besonderheiten

- # (Number) Kommentar bis Zeilenende
- ; (Semikolon) nach jeder Anweisung
- {.....} Blöcke von Anweisungen
- if, for, while - Blöcke immer in {....}
- Strings beliebiger Länge (RAM!) erlaubt
- Rekursion beliebiger Tiefe (RAM!) erlaubt
- String-Konkatenation mit Punkt: $x = "A"."B";
- Vergleich String: "eq", Number: "=="

Variablentypen und automatische Konversion

- Variablennamen beachten Groß- u. Kleinschreibung
- $var ein Skalar (string, float, int...)
- @var ein Int-Array ($#var Anzahl Elemente)

$var[0] Zugriff auf 1. Array-Skalar

- %var ein String-Array (assoziatives Array)

$var{"schluessel"} Zugriff auf String-Array

- $camels = "123";

```
print $camels + 1 . " Kamele\n";
# ergibt 124 Kamele
```

Programmfluß

```
if ($zahl > 10)
    {print "groesser 10\n";}
else
    {print "kleiner gleich 10\n";}

while ($os2 > $win95)
    {print "Endlos-Schleife... ";}

for ($i=1; $i<=10; $i++)
    {print "$i mal $i = ", $i*$i, "\n";}
```

Subroutinen

```
print "Ergebnis:  ", &MeinSub(10, 12), "\n";

sub MeinSub
{
    local ($argument1) = $_[0];
    local ($argument2) = $_[1];
    return $argument1 + $argument2;
}
```

Datei-Handling

- Dateihandles z.B. IN oder OUT oder Datei
- Filetests:

```
if (-e "perl.exe") {print "existiert!";}
if (-d "c:\\Desktop") {print "directory");}
if (-T "text.txt") {print "textfile");}
...und viele mehr...
```

-

```
open (inFile, "ALT.TXT");
open (outFile, ">NEU.TXT");
while ($zeile = <inFile>)
    {print outFile $zeile;}   # Kein Komma!
close (inFile);
close (outFile);
```

Reguläre Ausdrücke #1

- SUCHEN /...../

```
if ($zeile =~ /os\/2 [warp]* v.*[34]/i)
{   print "z.B. OS/2 V4\n";
    print "z.B. os/2 warp version 3";
}
```

- ERSETZEN (substitute) s/...../...../

```
$zeile =~ s/win.{0,5}95/OS\/2/ig;
# win95  -> OS/2
# Windows 95 -> OS/2
```

- Ende-Modifikatoren:

i "ignore case"

g "global" (alle suchen oder ersetzen)

- `quotemeta("Ein *illegales* [Pattern]")`

```
---> Ein\ \*illegales\*\ \[Pattern\]
```

Reguläre Ausdrücke #2

- Nicht-Sonderzeichen findet sich selbst (aber: *)
- . steht für ein einzelnes Zeichen (außer Newline)
- + vorhergehendes Zeichen ein- o. mehrmals
- ? vorhergehendes Zeichen null- o. einmal
- * vorhergehendes Zeichen null- o. mehrmals
- [...] Zeichenklasse [xyz] [a-zA-Z0-9]*
- [^...] Nicht aus dieser Zeichenklasse
- (...|...|...) Alternativen
- (....) späterer Klammernzugriff mit $1, $2, $3...

```
if ($a =~ /\t(os\/2.*4)\t) {print "$1\n";}
```

Goodies #1

- @ARGV - Argumente von Kommandozeile

z.B. $ARGV[0]

- @_ - Argumente für Subroutine

z.B. $_[0]

- %ENV - Umgebungsvariablen

z.B. $ENV{"PATH"}

-

Goodies #2

- pack ($Schablone, @Liste) -> $binaererString
  unpack ($Schablone, $binaererString) -> @Liste
- chop ($String) - letztes Zeichen (\n) abschneiden
- eval ($String) - für selbstmodifizierenden Code
- keys (%AssoArray) -> @AllKeys
- push(), pop(), sort(), splice(), shift() : @Array-Fkts.
- @Array = split (/..pattern.../ , $String)

perl V5 - Verbesserungen

- Komplexe, verschachtelte Datenstrukturen
- Pointer-ähnliche Gebilde
- Objektorientierte Programmierung
- *? - wie * aber "nicht gierig"
- Kommentare in regulären Ausdrücken
- "-w" Kommandozeilen-Schalter für Warnungen
- u.v.m.

Ein kleines Beispiel

```
# Enternt HTML-Tags aus Datei
# z.B. <A HREF = "...">XYZ</A>   -->  XYZ

open (ReinDatei, $ARGV[0]) || die "open()\n";
while ($zeile = <ReinDatei>)
{
    chop ($zeile);              # \n raus
    $alleZeilen = $alleZeilen.$zeile."\t";
}
close (ReinDatei);

$alleZeilen =~ s/\<.*?\>//gi; # Nicht-Gierig
$alleZeilen =~ s/\t/\n/g;     # \n wieder rein
print $alleZeilen;
```

Quellenangaben

- Die perl-Hauptseite im WWW
http://www.perl.com/perl/index.html
- Diverse Portierungen (auch: OS/2)
ftp://ftp.leo.org/pub/comp/programming/languages/perl/CPAN/ports/
- Speziell: Win32-Port: http://www.hip.activeware.com/
- "Programming perl" (deutsch oder englisch)
Larry Wall & Randal L. Schwartz
O'Reilly & Associates, Inc. (Nutshell-Handbooks)
ca. 80,-DM