Die Definition eigener Textausgabeformate ist möglich!

Analog zu C gibt auch den printf()-Befehl.

```
printf Filehandle (Formatstring, Werteliste);
# wie in C
```

Prozesse starten und beenden

eval(string)

```
$String = „print ‚Hallo!'“;  eval($String);
# wandelt $String in eine Perl-Anweisung um und führt diese aus
```

system(@Befehlsliste)

```
@Befehl = („ls“, „-la „, „Verzeichnisname“);
system(@Befehl);    # führt den in @Befehl angegebenen Systembefehl
mit seinen Optionen und Parametern aus. Das laufende Perlprogramm wird
bis zum Ende des aufgerufenen Programms unterbrochen und anschließend
weitergeführt
```

exec(@Befehlsliste)

wie `system(@Befehlsliste)`, jedoch wird das Perlprogramm vor dem Aufruf des anderen Programms beendet, die weiteren Perlbefehle werden nicht mehr ausgeführt.

$ProzessNr = fork(@Befehlsliste)

wie `system(@Befehlsliste)`, jedoch läuft sowohl die Abarbeitung des Perlprogramms als auch die des aufrufenden Programms parallel weiter.

die($Message)

beendet das laufende Perlprogramm mit der angegebenen Meldung.

exit(Exitcode)

beendet das laufende Perlprogramm mit dem angegebenen Exitcode.

kill(Signal, Prozessliste)

sendet das Signal an alle Prozesse der Liste. Zum Beispiel

```
kill(9,12)     # beendet den Prozess mit PID 12
```

Kommandozeilen-Optionen

Mit den vielen Optionen von Perl kann man das Verhalten des Programmablaufs modifizieren. Zum Beispiel:

Zeilenweises Abarbeiten mehrerer Textdateien

Das folgende Programm mit dem Namen „**ausgabe.pl**“ wird für jede Zeile der Textdateien ausgeführt.

Aufruf: **ausgabe.pl Textdatei1 Textdatei2 ..**

```
#!/usr/bin/perl -n
$Zeile = $ ;   # nächste Zeile einlesen
print $Zeile
```

Ausführen eines Perlprogramms mitten aus einer Textdatei

Dies erlaubt das Testen eines Programms aus zum Beispiel einer Hilfedatei.

Aufruf: **perl –x Textdatei**

Inhalt der Textdatei:

```
Hier ist die Textdatei aus deren Mitte ein Perlcode
ausgeführt werden soll. Der Text hier ist reiner Platzfüller.
Beginnen wir mit dem Programmcode ...
#!/usr/bin/perl
  print „Hallo!!\n";   # hier steht der Programmcode
__END__
Viel Spass beim Ausführen!
```

PERL Debugger

Zur Fehlersuche in PERLScripts. Der Aufruf erfolgt mit

perl –d Scriptname

Die einzelnen Befehle werden in der Kommandozeile in Kurzform eingegeben. Jede Eingabe, die nicht als spezielle Debuggeranweisung interpretiert werden kann, wird als PERL-Befehl ausgeführt!

l oder l 10-17	**listet die nächsten Scriptzeilen auf**
L	**listet die nächste auszuführende Scriptzeile auf**
s	**führt die nächste Befehlszeile aus**
n	**wie s, jedoch mit Sprung in eine evtl. Subroutine**
RETURN	**wiederholt die letzte Eingabe**
r	**Beenden des momentanen Subroutine**
x	**listet die aktuellen Variableninhalte aus**
b ZeilenNr	**setzt Breakpoint**
c	**Programmlauf bis zum nächsten Breakpoint oder zum Ende**
d ZeilenNr	**löscht Breakpoint**
t	**trace on und off**
H	**History der Debuggerkommandos**
h	**help**
q	**quit**

Literatur

- Programming PERL, Larry Wall and R.L.Schwartz, O'Reilly & Associates (Die PERL-Bibel)
- Learning PERL, R.L.Schwartz and Tom Christansen, O'Reilly & Associates
- Teach Yourself PERL 5 in 21 Days, David Till, SAMS Publishing
- PERL by Example, Ellie Quigley, Prentice Hall

und viele andere …

Online Kurse

- RRZK Perlkurs, Farid Hajji,) (in Deutsch auch zum Download)
  http://gd.tuwien.ac.at/languages/perl/Hajji-Perlkurs/
- Links zu PERL TUTORIALS,
  http://www.perl.com/reference/query.cgi?tutorials

Tools in PERL

- CPAN, ***http://gd.tuwien.ac.at/languages/perl/CPAN/***
- Alles was man sich wünscht, ***http://www.perl.com/reference/***

Es gibt Leute, die halten einen Unternehmer für einen räudigen Wolf, den man totschlagen müsse. Andere wieder meinen, er wäre eine Kuh, die man ununterbrochen melken könne. Nur wenige sehen in ihm das Pferd, das den Karren zieht.

Sir Winston Churchil