

Perl



*"The three chief virtues of a programmer are:
Laziness, Impatience and
Hubris"*

Larry Wall

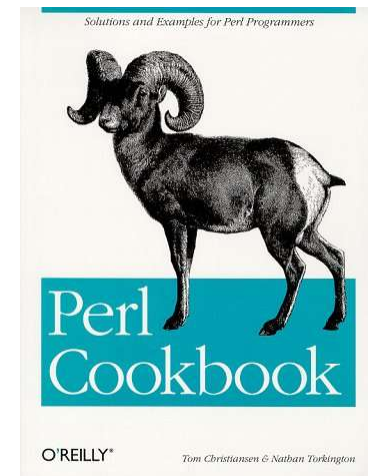
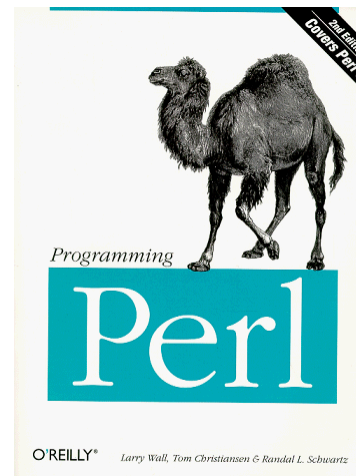
www.wall.org/~larry/

Inhalt

- Allgemeines zu Perl Scripts
- Skalare Variablen
- Rechnen mit Perl
- Vergleiche
- Regular Expressions
- Arrays
- Input
- Output
- Funktionen
- Hash
- Referenzen
- XML
- DBI

Leider nicht dabei 

- CGI
- Perl Module
- OO Perl
- Sockets



Allgemeines zu Perl

`#!/usr/bin/perl`

- DOS/Windows

`C:\>perl hello.pl` oder `C:\>hello.pl`

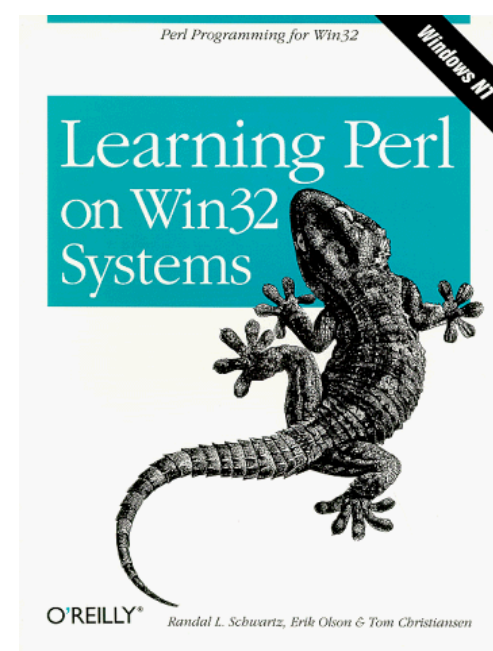
- Unix/Linux

`~$ /usr/bin/perl hello.pl` oder `~$./hello.pl`

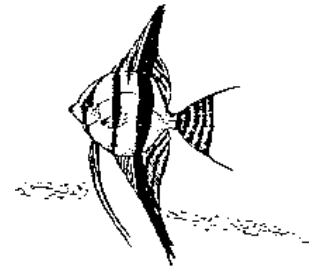
- Textfile

| | | |
|------------|-------------|------------|
| Zeilenende | DOS/Windows | Unix/Linux |
| Zeichen | CR / LF | LF |
| Hex | 0x0D / 0x0A | 0x0A |

File(s) : `hello.pl`



Skalare Variablen



```
use strict;
```

```
$name = "Peter";
```

```
print("Hallo, $name!\n");
```

```
print('Hallo, $name!\n');
```

```
File(s): var01.pl, var02.pl, var03.pl, var04.pl
```

Rechnen in Perl

```
$num1 = "3";  
$num2 = $num1 + 5.2 - "1.5";  
$num3 = "3.14159265";  
$num4 = "$num3" * (2.2 + 1);
```

File(s) : **calc01.pl**

Vergleich (numerisch)

| | | | |
|----|------------------|----|---------------|
| == | equal | != | not equal |
| > | greater than | < | less than |
| >= | greater or equal | <= | less or equal |

```
if ($num1 > 5) {  
    print ("greater\n");  
} else {  
    print ("not\n");  
}
```

File(s): **comp01.pl**

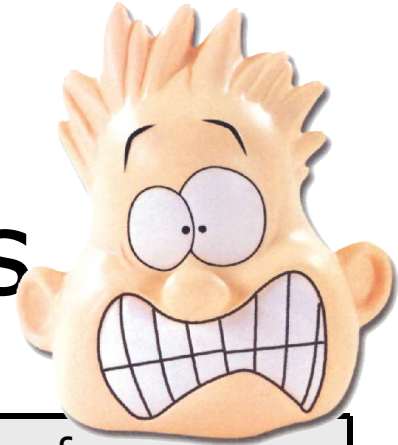
Vergleich (Strings)

| | | | |
|----|------------------|----|---------------|
| eq | equal | ne | not equal |
| gt | greater than | lt | less than |
| ge | greater or equal | le | less or equal |

```
if ($str1 gt "abc") {  
    print ("greater\n");  
} else {  
    print ("not\n");  
}
```

File(s) : comp02.pl

Regular Expressions



Matching Substitution
m/foo/ **s/foo/bar/**

| Pattern | Match |
|-------------------------|----------------------|
| <code>.*</code> | Alles |
| <code>abc*</code> | ab, abc, abcc, abccc |
| <code>(abc)*</code> | "", abc, abcabc, ... |
| <code>^x y</code> | x am Anfang oder y |
| <code>^(x y)</code> | x oder y am Anfang |
| <code>a bc d</code> | a oder bc oder d |
| <code>(a b)(c d)</code> | ac, ad, bc oder bd |

| | | | |
|-----------------|-------------|-----------------|--------------|
| <code>.</code> | ein Zeichen | <code>^</code> | Zeilenanfang |
| <code>()</code> | Gruppe | <code>\$</code> | Zeilenende |
| <code>[]</code> | Klasse | <code>\</code> | Metazeichen |
| <code> </code> | oder | | |

Multiplikatoren

| | | | |
|----------------|---------|--------------------|-----------------|
| <code>*</code> | mind. 0 | <code>{n}</code> | n mal |
| <code>+</code> | mind. 1 | <code>{n,}</code> | mind. n mal |
| <code>?</code> | 0 od. 1 | <code>{n,m}</code> | mind. n, max. m |

Zeichenklassen []

| | | | |
|-----------------|---------------------------|-----------------|-----------------------------|
| <code>\d</code> | <code>[0-9]</code> | <code>\D</code> | <code>[^0-9]</code> |
| <code>\w</code> | <code>[a-zA-Z0-9_]</code> | <code>\W</code> | <code>[^ a-zA-Z0-9_]</code> |
| <code>\s</code> | <code>[\r\t\n\f]</code> | <code>\S</code> | <code>[^ \r\t\n\f]</code> |

File(s) : **ref01.pl, ref02.pl**

Array

```
@name = ("Peter", "Paul", "Marry");  
$name[0] .= " Zwettler";  
print($name[1]);  
$name[5] = "Joe";  
($n1, $n2, $n3) = @name;  
# $_ default variable  
for (@name) { print("$_ "); }  
@name = sort(@name);
```

File(s): **array01.pl**

Input

```
$str = <STDIN>; # standard in
$str = @ARGV[1]; # @ARGV cmd line parameter

for (@ARGV) { print("$_ \n"); }
# input from file
open(INFILE, "$infile");
for ( <INFILE> ) { print("$_"); }
close(INFILE);
```

File(s): **file01.pl**

Output

```
print("$str\n"); # standard out
# output to file
open(OUTFILE, ">$outfile");
print(OUTFILE "foobar\n");
close(OUTFILE);
```

File(s): file02.pl, file03.pl

Funktionen

```
chomp();  
@fields = split(';', $str);
```

```
sub Foo()  
{  
    print("$_[0]");  
}
```

```
Foo("abc");
```

File(s): file04.pl, file05.pl



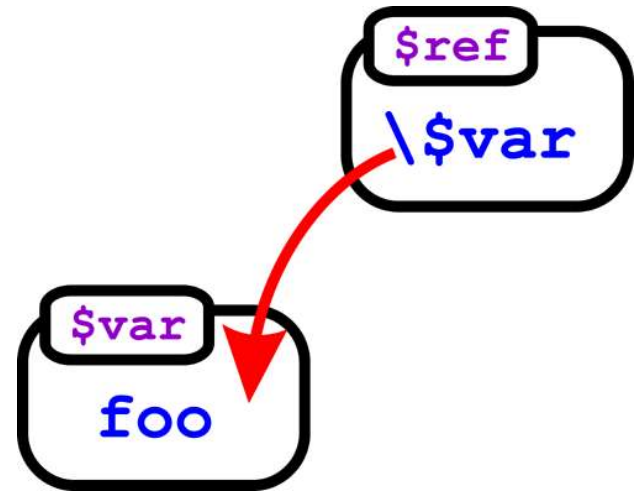
Hash

```
%person{"Name"} = "Peter";  
%person{"PLZ"} = "1030";  
@list = keys(%person);  
for (@list) {  
    print("$_ => %person{$_} \n");  
}
```

File(s): hash01.pl, hash02.pl

Referenzen

```
$var = "foo";  
$ref = \ $var;  
# print "foo"  
print (" $var");  
# print "foo"  
print (" $$ref");
```



File(s): **ref01.pl**

XML

```
XMLout($ref, RootName => "foobar",  
      KeyAttr => [ ]);
```

```
$ref = XMLin("$infile",  
            ForceArray => 1, KeyAttr=>[]);
```

File(s): **xml01.pl, xml02.pl**

DBI



```
$dbh = DBI->connect($connect_str);  
$sth = $dbh->prepare($sql);  
$sth->execute;  
$href = $sth->fetchrow_hashref();  
$dbh->disconnect();
```

DBI module installieren

```
c:\> ppm  
ppm> install DBI  
ppm> install DBD-DB2  
ppm> install DBD-ODBC
```

ODBC DataSource erstellen

File(s): **dbi01.pl, dbi02.pl, dbi03.pl**

Thanks!
There's More Than
One Way To Do It!
Lang Wall

