

Apr 20, 11 3:00

**perl-bit-boole-op-HOWTO.txt**

Page 1/2

HOWTO zur Perl-Bit-Operatoren und Booleschen Operatoren

(C) 2008 T.Birnthaler/H.Gottschalk &lt;howtos(at)ostc.de&gt;

OSTC GmbH, <http://www.ostc.de>

\$Id: perl-bit-boole-op-HOWTO.txt,v 1.4 2011-04-19 15:03:59 tsbirn Exp \$

Dieses Dokument beschreibt die in Perl verfügbaren Operatoren für Bit-Operationen und Boolesche Operationen.

## Inhaltsverzeichnis

- 1) Bit-Operatoren
- 2) Boolesche Operatoren
- 3) Beispiele für die Bit-Verknüpfungen
- 4) Beispiele für die Booleschen Verknüpfungen

## 1) Bit-Operatoren

Die Bit-Operatoren verarbeiten ihre Operanden bitweise, d.h. jede Bitposition wird für sich verarbeitet (NOT, LEFTSHIFT, RIGHTSHIFT) oder einzeln mit der korrespondierenden Bitposition des anderen Operators verknüpft (AND, OR, XOR).

Op	Name	Bedeutung
A & B	AND	Resultat-Bit WAHR wenn beide Op-Bits WAHR
A   B	OR	Resultat-Bit WAHR wenn eines der Op-Bits WAHR
A ^ B	XOR	Resultat-Bit WAHR wenn ein Op-Bit WAHR + eines FALSCH
~ A	NOT	Resultat-Bit WAHR wenn Op-Bit FALSCH (invertieren)
A << N	LEFTSHIFT	Bits um N Stellen nach links schieben (0 nach)
A >> N	RIGHTSHIFT	Bits um N Stellen nach rechts schieben (0 nach)

## 2) Boolesche Operatoren

Bitte nicht verwechseln mit den ähnlichen logischen (Booleschen) Operatoren (&& and, || or, ! not, xor), die ihre Operanden nicht bitweise, sondern als Ganzes unter dem Aspekt Wert WAHR oder Wert FALSCH (0 0.0 " ' "0" '0' () undef) verarbeiten und als Ergebnis wieder WAHR (meist "1") oder FALSCH (meist "") liefern (genauer: einen der Operanden unter dieser Bedeutung als Ergebnis liefern):

Op	Alternativ	Bedeutung
A && B	A and B	Resultat WAHR wenn beide Operanden WAHR
A    B	A or B	Resultat WAHR wenn einer der Operanden WAHR
! A	not A	Resultat WAHR wenn Operand FALSCH (negieren)
	A xor B	Resultat WAHR wenn ein Operand WAHR und einer FALSCH

Die Operatoren "&&" und "and", "||" und "or" sowie "!" und "not" haben exakt die gleiche Funktion BIS AUF DEN VORRANG. Während "&&", "||" und "!" VOR den Zuweisungs- (= -= += ...) und Listenoperatoren ( , => print sort ...) angesiedelt sind, sind "and", "or" und "not" NACH ALLEN Operatoren angesiedelt.

Dies ist einerseits in gewissen Situationen von Vorteil, kann andererseits aber zu subtilen Fehlern führen. Ein einfacher Austausch von "&&" <-> "and" bzw. "||" <-> "or" bzw. "!" <-> "not" in einem Programm führt normalerweise zu einem anderen Programmverhalten. Durch Klammerung können diese Unterschiede aber behoben werden. Beispiel:

```
&fields("create", $cont1, $cont2 or $cont3, $empty); # PENG!
&fields("create", $cont1, ($cont2 or $cont3), $empty); # OK!
&fields("create", $cont1, $cont2 || $cont3, $empty); # OK!

$readonly = $type ne "copy" and $type ne "rename"; # PENG!
$readonly = ($type ne "copy" and $type ne "rename"); # OK!
$readonly = $type ne "copy" && $type ne "rename"; # OK!
```

## 3) Beispiele für die Bit-Verknüpfungen

\* &amp; (Bitweise AND)

	AND	0	1		
11001100	+	+	+	+	0 & 0 -> 0
10101010		0	0	0	0 & 1 -> 0
-----	+	+	+	+	1 & 0 -> 0
10001000		1	0	1	1 & 1 -> 1

```

+-----+-----+
* | (Bitweise OR)
      | OR | 0 | 1 |
11001100  +-----+-----+   0 | 0 -> 0
10101010  | 0 | 0 | 1 |   0 | 1 -> 1
-----+-----+   1 | 0 -> 1
11101110  | 1 | 1 | 1 |   1 | 1 -> 1
+-----+-----+

```

```

* ^ (Bitweise XOR = entweder-oder, exclusive or)
      | XOR | 0 | 1 |
11001100  +-----+-----+   0 ^ 0 -> 0
10101010  | 0 | 0 | 1 |   0 ^ 1 -> 1
-----+-----+   1 ^ 0 -> 1
01100110  | 1 | 1 | 0 |   1 ^ 1 -> 0
+-----+-----+

```

```

* ~ (Bitweise NOT = invertieren)
      | NOT | 0 | 1 |
11001100  +-----+-----+   ~ 0 -> 1
-----+-----+   ~ 1 -> 0
00110011  | 1 | 0 | 1 |
+-----+-----+

```

```

* << (Bitweise LEFTSHIFT = Multiplikation mit 2)
      11001100 (204)      10101010111 (1367)
-----+-----+
110011000 (408)      101010101110 (2734)

```

```

* >> (Bitweise RIGHTSHIFT = Division durch 2)
      11001100 (204)      10101010111 (1367)
-----+-----+
01100110 (102)      1010101011 (683)

```

#### 4) Beispiele für die Booleschen Verknüpfungen

```

* F = Falsch (5 Werte in Perl: 0 "" "0" () undef)
  W = Wahr (ALLE anderen Werte in Perl)

```

```

* && bzw. and (logisches AND)

```

```

| AND | F | W |
+-----+
| F | F | F |   W and W -> W
+-----+   W and F -> F
| F | F | F |   F and W -> F
+-----+   F and F -> F
| W | F | W |
+-----+

```

```

* || bzw. or (logisches OR)

```

```

| OR | W | F |
+-----+
| W | W | F |   W or W -> W
+-----+   W or F -> W
| W | W | F |   F or W -> W
+-----+   F or F -> F
| F | F | F |
+-----+

```

```

* xor (logisches XOR = entweder-oder, exclusive or)

```

```

| XOR | W | F |
+-----+
| W | W | F |   W xor W -> F
+-----+   W xor F -> W
| W | W | F |   F xor W -> W
+-----+   F xor F -> F
| F | F | W |
+-----+

```

```

* ! oder not (logisches NOT = negieren)

```

```

| NOT | W | F |
+-----+
|   | F | W |   not W -> W
+-----+   not F -> F

```