

Weird programming

Vom Nutzen unnützer Programmierung



- Ziel: Ein unterhaltsamer Blick über den ernsten Tellerrand
- Gehalten auf dem 20C3
- Auf vielfachen Wunsch heute wiederaufgelegt
- Referent: Markus Schaber
 - <http://www.schabi.de/>
 - [REDACTED]
- Vortrag ist im HTML-Format online
 - <http://ulm.ccc.de/~schabi/weirdprog20c3/>
- Fortsetzung "Weird Programming 2" morgen, 11:00, Saal 1 (in Englisch)

Inhaltsübersicht

- Motivation
- Disziplinen
 - Hello World
 - 99 Bottles of Beer
 - Quines
 - Spaßige Programmiersprachen
 - Kürzestmögliches Programm
 - Obfuscated Programming
 - Polyglot
 - Kombidisziplinen
 - Weiteres...



Aspirin bereithalten!

Motivation



- Übliche Ziele bei der Programmierung:
 - Ein Problem gelöst bekommen
 - Geld verdienen
 - Gut wartbaren Code erzeugen
 - Möglichst wenig Kosten
 - Möglichst wenig Zeit
 - Übungsaufgabe lösen
 - Arbeit erleichtern
- Übliche Ziele im Sprachdesign / Compilerbau:
 - Einfache Compiler ermöglichen
 - "Universalsprache" erfinden
 - Bestimmte Aufgaben effizient lösbar machen
 - Ein Beispiel für ein Lehrbuch / eine Vorlesung schaffen

Motivation



- Ziele heute Abend aber:
 - Einfach mal was anders machen
 - Programmieren als Wettkampf
 - Zeitvertreib
 - Möglichst knifflige Rätsel schaffen
 - Erstaunen / Anerkennung beim Publikum hervorrufen
 - Satire / Veralberung althergebrachter Grundsätze
 - Etwas "unmögliches" erreichen
 - Einfaches kompliziert machen
 - Spass haben
 - vielleicht auch: Alkoholrausch / Drogentrip ausleben?

Hello World



- Beliebte Aufgabe im Bereich "mein erstes Programm"
- Wohl am häufigsten gelöste Programmieraufgabe
- Sammlung des ACM
 - <http://www2.latech.edu/~acm/HelloWorld.shtml>
 - Über 200 Programmiersprachen
- Eigentlicher Nutzen eher sehr gering
- Hier ein paar heute eher "exotische" Programmiersprachen

Hello World - Exoten:



- Caml

```
let printHelloWorld() =  
    print_string "Hello World\n";;
```

- Haskell

```
main = print("Hello World")
```

- HyperTalk

```
on OpenStack  
    show message box  
    put "Hello World!" into message box  
end OpenStack
```



99 Bottles of Beer

- Fortgeschrittenenaufgabe in Programmierkursen
- Dient zur Einführung von Schleifen
- Erfunden im Usenet als Reaktion auf Troll
- <http://99-bottles-of-beer.ls-la.net/> - derzeit über 620 Beispiele
- Das historische erste Auftreten:

```
10 REM Basic version of 99 bottles of beer
20 FOR X=100 TO 1 STEP -1
30 PRINT X;"Bottle(s) of beer on the wall,";X;"bottle(s) of beer"
40 PRINT "Take one down and pass it around,"
50 PRINT X-1;"bottle(s) of beer on the wall"
60 NEXT
```

99 Bottles of Beer



- APL2-Version:

```

BEER;⍵IO;⍵PW
⍝APL2 version (no loops) of "99 bottles of beer" by Chuck Kennedy
IO←0          ⍝Start counting from 0
PW←150        ⍝Set printing width to 150 chars
a←,[10]εε1100  ⍝Integers from 99 thru 0 as a column matrix
b←c' bottles of beer on the wall,'
c←c' bottles of beer,'
d←c' Take one down and pass it around.'
e←((a,b,a,c),d),(1εa),b  ⍝Build the song, next two lines fix the English
e[98;1 3 4 5]←(c(≡b)~'s')(c(≡c)~'s')(c'Take it down and pass it around.')(c'No more')
e[99;]←'No more' b 'No more' c 'Go to the store and buy some more.' 99 b
e  ⍝Display the song

```


99 Bottles of Beer



- Makefile

```
# quick effort at 99 bottles program using gnu make
#
# the file must be called makefile.bottles
#
# Author: Andrew Dunstan (andrew.dunstan@its.maynick.com.au)
#

default:
    $(MAKE) -f makefile.bottles BOTTLES=99 bottles

.SILENT:

bottles:
    echo $(BOTTLES) bottles of beer on the wall
    echo $(BOTTLES) of beer
    echo Take one down and pass it around
ifeq ($(BOTTLES),0)
    echo No bottles of beer on the wall
else
    echo `expr $(BOTTLES) - 1` bottles of beer on the wall
    echo
    $(MAKE) -f makefile.bottles BOTTLES=`expr $(BOTTLES) - 1` bottles
endif
```



Quines

- Ein Programm, das den eigenen Quelltext ausgibt
- Benannt nach Willard van Orman Quine, via Douglas Hofstadter
- Aufgabe ist eigentlich nicht so einfach:

```
print "hallo"
```

ZU

```
print "print \"hallo\""
```

tut nicht!

- In BASIC trivial, da Zugriff auf den Quelltext

```
10 LIST
```

- Einfachstes Quine: Leeres Programm
 - zulässig z. B. in Bash, Python, Perl, HQ9+, K&R-C...
- Für ein "echtes" Quine ist aber etwas Trickserei notwendig



Kurze Quines

- Lisp, Scheme:

```
((lambda (x)
  (list x (list (quote quote) x)))
 (quote
  (lambda (x)
    (list x (list (quote quote) x)))))
```

- C

```
char*f="char*f=%c%s%c;main(){printf(f,34,f,34,10);}%c";main(){printf(f,34,f,34,10);}
```

- Perl

```
$_=q{$_=q{Q};s/Q/$_/;print};s/Q/$_/;print
```

- Python

```
_='_=%s;print _%%'\_'\';print _%\_\'
```

Spassige Programmiersprachen



- Anscheinend nach dem Genuss von 99 Bottles of Beer entstanden
- Bereiten manchmal Kopfschmerzen
- Einige davon haben Obfuscation (siehe später) als Hauptziel

HQ9+



- Optimiert auf die wichtigsten Problemstellungen des Programmieranfängers:
 - H - gibt "Hello World" aus
 - Q - gibt den eigenen Quelltext aus
 - 9 - gibt 99 Bottles of Beer aus
 - + - erhöht das Arbeitsregister um 1
- Einige Beispiele:
 - Hello World
`H`
 - Quine 4-fach
`QQQQ`

Brainfuck



- Erste Implementierung von Urban Müller für den Amiga
- Brainfuck-Compiler für AmigaOS 2.0: nur 240 Bytes (!)
- Compiler für i386/Linux/ELF: 172 bytes
- Quine benötigt etwa 3 Kilobyte Größe!
- Primitive "virtuelle Maschine" mit 1 Adressregister
- 30000 Bytes Datenspeicher (am Anfang 0)
- Programmspeicher "unsichtbar"
- Varianten: (Compressed BF, Preprozessor, I/O extensions, self-modifying code, Multi-Threading mit Concurrent Brainfuck...)



Brainfuck - Opcodes

- 8 Befehle:

Opcode	Bedeutung	C-Code
>	Adressregister erhöhen	++p;
<	Adressregister erniedrigen	--p;
+	Speicherzelle an aktueller Adresse erhöhen	++*p;
-	Speicherzelle an aktueller Adresse erniedrigen	--*p;
.	Aktuelle Speicherzelle ausgeben	putchar(*p);
,	Einlesen und speichern in aktueller Speicherzelle	*p=getchar()
[Springe hinter die zugehörige], falls p=0	while (*p) {
]	Springe zur zugehörigen [}

Brainfuck



- Kürzester BF-Interpreter in C:

```
char m[9999],*n[99],*r=m,*p=m+5000,**s=n,d,c;main(){for(read(0,r,4000);c=*r;
r++)c-' ]' || (d>1 || (r=*p?*s:(--s,r)),!d || d--),c-' [' || d++ || (*++s=r),d || (*p+=c==
'+',*p-=c=='-',p+=c=='>',p-=c=='<',c-'.' || write(2,p,1),c-', ' || read(2,p,1));}
```

- Hello World in Brainfuck:

```
>+++++++ [ <+++++++>- ] < . >+++++++ [ <++++>- ] < + . ++++++ . . + + + . [
- ] >+++++++ [ <++++>- ] < . >+++++++ [ <+++++++>- ] < - . - - - - - - .
+ + + . - - - - - . - - - - - . [ - ] >+++++++ [ <++++>- ] < + . [ - ] + + + + + + + + .
```


Intercal



- Entstanden am 26. Mai 1972
- Erste Implementierung auf Atari, leider verschollen
- Unix-basierter Compiler von Eric S. Raymond 18 Jahre später
- Der Compilername steht für "Compiler Language With No Pronounceable Acronym"
- Enthält Statements wie
 - DON'T GIVE UP (effektiv ein NOP)
 - COME FROM (motiviert durch ACM April-Scherz: R. L. Clark, "A linguistic contribution to GOTO-less programming," *Commun. ACM* 27 (1984), pp. 349-350))
- Ebenfalls diverse Erweiterungen verfügbar



Intercal - Beispiel

- Berechnet den Absolutwert der eingegebenen Zahlen:

```

DO (5) NEXT
(5) DO FORGET #1
PLEASE WRITE IN :1
DO .1 <- '?' :1~'#32768$#0' "$#1'~#3
DO (1) NEXT
DO :1 <- "'?' :1~'#65535$#0' "$#65535'
~'#0$#65535' "$''?' :1~'#0$#65535' "
$#65535'~'#0$#65535' "
DO :2 <- #1
PLEASE DO (4) NEXT
(4) DO FORGET #1
DO .1 <- "'?' :1~:2'$#1"~#3
DO :1 <- "'?' :1~'#65535$#0' "$":2~'#65535
$#0' "'~'#0$#65535' "$''?' :1~'#0
$#65535' "$":2~'#0$#65535' "'~'#0$#65535' "
DO (1) NEXT
DO :2 <- ":2~'#0$#65535' "
$"'":2~'#65535$#0' "$#0'~'#32767$#1' "
DO (4) NEXT
(2) DO RESUME .1
(1) PLEASE DO (2) NEXT
PLEASE FORGET #1
DO READ OUT :1
PLEASE DO .1 <- "'?' :1~:1'~#1"$#1'~#3
DO (3) NEXT
PLEASE DO (5) NEXT
(3) DO (2) NEXT
PLEASE GIVE UP

```

- Geht auch kürzer, z. B. in APL: $[1] \Rightarrow \square \neq \square \leftarrow | \square$

BeFunge & Co.



- Herkömmliche Quelltexte sind eindimensional
- "Logische Erweiterung" bzw. "nächsthöhere Ebene": Programme und Daten in mehrdimensionalen Array
- Instruction Pointer wird mit Sprungvektor bearbeitet
- Arbeitet mit einem Stack of Stacks
- Aktuelles Standard-Dokument: Funge98
- Auch UneFunge und TriFunge sind spezifiziert
- Angedachte Erweiterungen: Sechseckige Felder, Klein-Bottles etc. :-)



BeFunge - Beispiele

- Hello World:

```
>v"Hello world!"0<
^:
^_25*,@
```

- 99 Bottles of Beer:

```
92+9*          :. v <
>v"bottles of beer on the wall"+910<
^:
^_ $
          :.v
          >v"bottles of beer"+910<
          ^:
          ^_ $
>v"Take one down, pass it around"+910<
^:
^_ $          1-v
          :
          >v"bottles of beer"+910.:_ v
          ^:
          ^_ $          ^
          >v" no more beer..." +910<
          ^:
          ^_ $$ @
```

- Quine mit 14 Bytes:

```
:0g,:93+`#@_1+
```

Java2k



- *"Propabilistische Programmiersprache für Physiker"*
- IDE namens DIE verfügbar für Win32, Linux und Amiga
- Bezeichner: Vielfache von 7 zur Basis 11 (Ziffern: "0123456789 ")
- Alle Funktionen haben genau 2 Argumente
 - Ausnahme: If-Goto-Else-Statement
- Variablen nur über indirekte Arrays
- Statements haben genau 90% "Erfolgswahrscheinlichkeit"
 - Ausnahme: "119 " - Wahrscheinlichkeit erhöhen

Java2k



- Selbstdefinierte Funktionen nur indirekt aufrufbar:
 - EVALUATE-FUNCTION-BRANCH-EQUAL-CALL-OTHERWISE function

```
A<=>~(B*C);
```

- Berechne selbstdefinierte Funktion A
 - Wenn A 0 zurückgibt, rufe Funktion C auf
 - Andernfalls, rufe Instruktionssequenz B auf
- Zahlen nicht als Literale möglich, nur durch Errechnung

- Hilfe: * für Zufallszahl, _ wiederholt letztes Argument
- Erzeugen von 1 mit 90% Wahrscheinlichkeit:

```
11 6/*/_\
```

- Erzeugen von 2 mit 81% Wahrscheinlichkeit:

```
125 /11 6/*/_\/_\
```

Java2k



- Einziger logischer Operator: "13 2", implementiert NAND
 - A OR B schreibbar als:
`13 2/13 2/A/B\13 2/A/B\`
- Whitespaces: -, A-D, F-R, T-Z
 - Können überall (auch innerhalb von Bezeichnern) sein
 - Kommentare gibt es nicht mehr ("use whitespaces instead")
 - "selten benutzte" E und S jedoch reserviert zur Thread-ID-Adressierung
 - Threads wurden im selben Release abgeschafft.
- Die Doku an sich ist genauso schlimm, wie die Sprache

Chef



- Programme sehen aus wie Kochrezepte
- Zutaten sind Daten (Variablen).
 - Flüssige Zutaten beinhalten Unicode-Zeichen
 - alle sonstigen Zutaten beinhalten Zahlen
 - Liquify etc. zum umwandeln
- Stacks zum Speichern: "Rührschüsseln und Backformen"
- Optionale Statements für Kochzeit und Ofen-Temperatur
- Unterprogramme:
 - hinten angehängte Zutaten-Rezepte (z. B. Soßen)
 - Haben eigene Rührschüsseln
 - Aber: auch Zugriff auf Kopien der Haupt-Behälter
 - Liefern exakt eine Rührschüssel zurück.

Chef



- Hello-World-Soufflet:

Hello World Souffle.

This recipe prints the immortal words "Hello world!", in a basically brute force way. It also makes a lot of food for one person.

Ingredients.

72 g haricot beans
101 eggs
108 g lard
111 cups oil
32 zucchinis
119 ml water
114 g red salmon
100 g dijon mustard
33 potatoes

Method.

Put potatoes into the mixing bowl. Put dijon mustard into the mixing bowl. Put lard into the mixing bowl. Put red salmon into the mixing bowl. Put oil into the mixing bowl. Put water into the mixing bowl. Put zucchinis into the mixing bowl. Put oil into the mixing bowl. Put lard into the mixing bowl. Put lard into the mixing bowl. Put eggs into the mixing bowl. Put haricot beans into the mixing bowl. Liquify contents of the mixing bowl. Pour contents of the mixing bowl into the baking dish.

Serves 1.

Beatnick



- Einfach zu lernen:
 - Sehr wenige Kommandos
 - Sehr freie Syntax
 - Vokabel-Referenz in jedem Spielzeuggeschäft erhältlich
- Bedeutung der Wörter errechnet sich aus Scrabble-Punktzahl
- Stack-basierte Maschine
- Auszug aus der Befehlstabelle:

Score	Function
<5	Does nothing. The Beatnik Interpreter may mock you for your poor scoring, at its discretion. Low scoring words such as "I" or "of" are probably not good words to program with immediately after stealing all of the interpreter's cigarettes and stomping on its beret.
18-23	Does nothing. However, the score is high enough that the Beatnik Interpreter will not mock you, unless it's had a really bad day.
>23	Garners "Beatnik applause" for the programmer. This generally consists of reserved finger-snapping.

Whitespace

- Befehle bestehen aus Folgen von Blank, Tab und Newline
- Vorteil: massenhafte Ausdrücke von Source-Listings sparen Tinte
- Beispiel Quelltext:



Kürzestmögliches Programm



- Gesucht: die kürzestmögliche Lösung einer Aufgabe
- Beispiel: Kürzestes Programm, das die Standardeingabe als Binärausgabe umwandelt

```
main(c){while(~(c=c<4?getchar()|256:c/2))putchar(48|c%2);}
```

(58 Zeichen, Gewinner-Beitrag im lokalen CCC-Contest diesen Sommer)

- Beispiel-Eingabe:

```
abc
```

- Beispiel-Ausgabe:

```
100001100100011011000110
```

- Aktuell: Congress-Wettbewerb, siehe <http://ulm.ccc.de/shortest/>

Obfuscated Programming



- Die Kunst, Programme so zu schreiben, daß kein Mensch den Quelltext wieder verstehen kann.
- Wird ebenfalls von Perl-Programmierern immer wieder gerne gepflegt :-)

Perl-Camel



- Ausgabe des Programms:

```

      mJXXLm.          .mJXXLm.          .mJXXLm.          .mJXXLm.
      JXXXXXXXXXL.    JXXLm.            .mJXXL            .JXXXXXXXXXL
      {XXXXXXXXXXXXX.  JXXXXmXXXXXm mXXXXmXXXXL .XXXXXXXXXXXXX}
      .XXXXXXXXXXXXXL. {XXXXXXXXXXXXF 7XXXXXXXXXXXX} .JXXXXXXXXXXXXX.
      JXXXXXXXXXXXXXXXXXXXXXXXXL. \XXXXXX. .XXXXXXXX'.JXXXXXXXXXXXXXXXXXXXXL
      JXXXXXXXXXXXXXXXXXXXXXXXXmXXXXXXXX. .XXXXXXXXmXXXXXXXXXXXXXXXXXXXXL
      .XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX. {XXXXXXXXXXXXXXXXXXXXXXXXXXXXX.
      .XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX. XXXXXXXXXXXXXXXXXXXXXXXXXXXXX.
      JXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 7XXXXXXXXXXXXXXXXXXXXXXXXXXXXL
      XX^7XXXXXXXXXXXXXXXXXXXXXXXXX 7XXXXXXXXXXXXXXXXXXXXXXXXX^XX
      XX {XXXFXXXXXXXXXXXXXXXXXXXXXX' 'XXX' 'XXXXXXXXXXXXXXXXXXXX7XXX' XX
      'X' {XXX'7XXXFXXXXXX^XXXX ' ' ' XXXXX^XXXXX7XXXFX'XXX' {X'
      'XXX' {XXX'XXXXX 7XXXFX 7XXXFX XXXXX'XXX' 'XXX' '
      .XX} {XXF {XXXX} 'XXX' {XXX' {XXXX} 7XX} {XX.
      {XX 'XXL '7XX} 7XX} {XXF {XXF' JXX' XX}
      'XX 'XXL mXXF {XX XX' 7XXm JXX' XX'
      XX 7XXXFX 'XX XX' 7XXXFX XX
      XX. JXXXX. 7X. .XF .XXXXL .XX
      {XXL 7XF7XXX. {XX XX} .XXXFXFX JXX}
      'XXX' 'XXXm mXXX' 'XXX'
      mmm.m.mmm..mm7XXXX.mm.mm. .mm.mm.XXXXFmm..mmm.m.mmm
      '7LFJXXJX\FLXXL7FLXJF\X{ }X\7LXJ7FJXXJ7\XLXXL7JF'
      ^^^^^^^^^^^^ ^^^^^^^^^^^^
      .mJXXLm.          mJXXLm.          mJXXLm.          mJXXLm.
      JXXXXXXXXXL.    JXXXXXXXXXL.    JXXLm.          JXXLm.
      {XXXXXXXXXXXXX.  .XXXXXXXXXXXXX.  {XXXXXXXXXXXXX.  JXXXXmXXXXXm
      7XXXXXXXXXXXX.  .JXXXXXXXXXXXXX.  .XXXXXXXXXXXXXL.  {XXXXXXXXXXXXX
      .XXXXXXXX'.JXXXXXXXXXXXXXXXXXXXXL. JXXXXXXXXXXXXXXXXXXXXL. \XXXXXX.
      .XXXXXXXXmXXXXXXXXXXXXXXXXXXXXL. JXXXXXXXXXXXXXXXXXXXXXXXXmXXXXXXXX.
      {XXXXXXXXXXXXXXXXXXXXXXXXXXXXX. .XXXXXXXXXXXXXXXXXXXXXXXXXXXXX.
      .XXXXXXXXXXXXXXXXXXXXXXXXXXXXX. XXXXXXXXXXXXXXXXXXXXXXXXXXXXX.
      7XXXXXXXXXXXXXXXXXXXXXXXXXXXXL JXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      7XXXXXXXXXXXXXXXXXXXXXXXXXXXXX^XX XX^7XXXXXXXXXXXXXXXXXXXXX
      '7XXXXXXXXXXXXXXXXXXXX7XXX' XX XX {XXXFXXXXXXXXXXXXXXXXXXXXXX'
      ' ' XXXXX^XXXXX7XXXFX'XXX' {X' 'X' {XXX'7XXXFXXXXXX^XXXX ' '
      7XXXFX XXXXX'XXX' 'XXX' ' 'XXX' {XXX'XXXXX 7XXXFX
      {XXX' {XXXX} 7XX} {XX. .XX} {XXF {XXXX} 'XXX'
      {XXF {XXF' JXX' XX} {XX 'XXL '7XX} 7XX}
      XX' 7XXm JXX' XX' XX 'XXL mXXF {XX
      XX' 7XXXFX XX XX 7XXXFX 'XX
      .XF .XXXXL .XX XX. JXXXX. 7X.
      XX} .XXXFXFX JXX} {XXL 7XF7XXX. {XX
      mXXX' 'XXX' 'XXXm 'XXX' 'XXX'
      .mm.mm.XXXXFmm..mmm.m.mmm. mmm.m.mmm..mm7XXXX.mm.mm.
      }X\7LXJ7FJXXJ7\XLXXL7JF' '7LFJXXJX\FLXXL7FLXJF\X{
      ^^^^^^^^^^^^ ^^^^^^^^^^^^
  
```

The use of a camel image in association with Perl is a trademark of O'Reilly & Associates, Inc. Used with permission.#camelcode,copywrite2000byStephenB.Jenkins

IOCCC

- The International Obfuscated C Code Contest
- <http://www.ioccc.org/>
- (fast) jährlicher Wettbewerb





IOCCC - Appetithäppchen 2001

hello_2001.c

```
#include <stdio.h>
#define S(s)char x[]=#s;s
#define Q(x)x
#define A(x,y)y##x
#define B(x,y)A(y,x)
#define C(x,y)B(y,x)
#define Z(s,t,u)case s:if(*p!=32){t;}else{u;}break;
S(B( A( a ,m ),A(n ,i))() {B (A(h,c ),A(r ,a ))*p=x ;B(A( n,
=0;B(A(n , i),t)s =0;B( f ,A(r, o )) (;*p;Q( p)++){C( B( A(c,
w, s),i))( s){ Z( 0,t+=8 *8-00 ,s ++)Z( 1,t+= 8 ;,s++ )Z
( 2, t++ ,putchar(t-73);t=s=0)}}})
```

Ausgabe:

Hello, world!

IOCCC - anderson.c



- Contest 2000 - anderson.c
- Konvertiert stdin in ASCII-Grafik Flaggencode
- Beispiel: Hello World als Eingabe erzeugt die Ausgabe

```

    _()
   / ^
  <>[

    ()/
   |^^
  [][

    ()/
   /^^
  <>][

    ()/
   /^^
  <>][

    _\ )
   ^ ^
  [ ] ]

    ( )
   |^^|
  [ ] ]

    (/ _
   ^^ [ ]

    _\ )
   ^^ ]

    _()
   ^^ [ ]

    ()/
   /^^
  <>][

    [ ]
   | ( )
   ^^ |
  ] [ ]

    ( )
   |^^|
  [ ] [ ]
    
```

IOCCC - anderson.c



```

/* anderson.c */
#include <stdio.h>

char
*T="IeJkLMaYQCE]jbZrskc[slDU^V\X\\|/_<[<:90!\\"$434-./2>]s",
K[3][1000],*F,x,A,*M[2],*J,r[4],*g,N,Y,*Q,W,*k,q,D;X(){r [r
[r[3]=M[1-(x&1)][*r=W,1],2]=*Q+2,1]=x+1+Y,*g++=((x&
-1)>>1-1)?*r:r[x>>3],(++x<*r)&&X();}E(){A|X(x=0,g
),x=7&(T>A*3),J[(x[F]-W-x)^A*7]=Q[x&3]^A*(M[2
x&1]),g=J+(x[k]-W)^A*7)-A,g[1]=(M)[*g=M[T+=A
][x&1],x&1],(A^=1)&&E(),J+=W);}l(){E(--q&&l
);}B(){*J&&B((D=*J,Q[2]<D&&D<k[1])&&(*g++=1
!(D-W&&D-9&&D-10&&D-13)&&(!*r&&(*g++=0)
r=1)||64<D&&D<91&&(*r=0,*g++=D-63)||D
97&&D<123&&(*r=0,*g++=D-95)||!(D-k[
]&&(*r=0,*g++=12)||D>k[3]&&D<=k[
-1&&(*r=0,*g++=D-47),J++)};j(
putchar(A);}b(){(j(A=(*K)[D*
r[2]*Y+x),++x<Y)&&b());}t
{(j((b(D=q[g],x=0),A=W)
++q<(*r+1)<Y?*r+1):
)&&t();}R(){(A=(t(
0),'n'),j(),++r
]<N)&&R();}O(
j((r[2]=0,R(
),r[1]-=q)
O(g=-=q)
C(){(
gets
[1])&&C((B(g=K[2]),*r=!(*r&&(*g++=0)),(*r)[r]=g-K[2],g=K[2
],r[
1]&&
O())
);};
main
(){C
((l(
J=(
A=0)
[K],
A[M]
=(F=
(k=(
M[A]
]=Q)
=T+(
q=(Y
=(W=
32)-
(N=4
)))
+N)+
2)+7
)+7)
),Y=
N<<(
*r=!
-A)
);};

```



```
MAiN      MAiN      maIn
mAln     main      MAIn
Ma1N     MAiN     main
mAln     main     ma1N  MA1n
mAln     MA1n
mAln     mAln
Ma1N     maIn
mAln     mAln
MAiN     main
ma1N     MAiN
Main     MA1n
```



"Job-Erhalt" durch unwartbaren Code



- Sonderform der Obfuscation
- "*Never ascribe to malice, that which can be explained by incompetence.*"
- Howto auf <http://mindprod.com/unmain.html>
- Code soll nicht unwartbar aussehen, sondern es nur sein.

"Job-Erhalt" durch unwartbaren Code



- Beispiel-Tipps:
 - Exakt das Offensichtliche kommentieren
 - Zeichen-Ähnlichkeiten ausnutzen
 - Gleiche Bezeichner in verschiedenen Scopes
 - Präprozessor intensiv missbrauchen
 - "Toten" Code einbauen
 - Variablen aus einem Baby-Namensbuch benennen
 - Einfach und schnell zu tippende Namen verwenden
 - Esperanto, Klingonisch und Hobbitesse für Bezeichner und Kommentare

Mehrsprachigkeit (Polyglot)



- Portierbarkeit bedeutet normalerweise:
 - Verschiedene Implementierungen derselben Sprache
 - Unterschiedliche Plattformen und Bibliotheken
- Polyglot-Programme erweitern das Konzept:
 - in mehreren Sprachen gleichzeitig gültiger Quelltext
 - Dazu noch sematisch äquivalent

Quine in TurboPascal und ANSI-C



```
(*a);main(){char i,*s[]={"%c%c%c%c",
"(*a);main(){char i,*s[]={",
"%c%c%c%c%c%c%c%c%c%c/%*c%c",
";printf(s[1]);for(i=0;i<=12;i++)printf(s[0],34,s[i],34,44,10);",
"printf(s[2],34,34,125,s[3],10,s[4],10,s[5],10,s[6],10,41,s[7]);",
"for(i=0;i<=12;i++)printf(s[0],39,s[i],39,44,10);",
"printf(s[12],39,39,41,s[8],10,s[9],10,s[10],10,s[11],10,47,125);",
"const q=#34;w=#39;n=#13#10;s=array[0..13] of string=",
";var i:integer;begin write(s[1]);for i:=0to 12do write(q,s[i],q,#44,n);",
"write(#34#34#125,s[3],n,s[4],n,s[5],n,s[6],n,#47#42#41,s[7]);",
"for i:=0to 12do write(w,s[i],w,#44,n); ",
"write(#39#39#41,s[8],n,s[9],n,s[10],n,s[11],n,#123#42#47#125);end.",
"%c%c%c%c%c%c%c%c%c%c/%*c%c",
""};printf(s[1]);for(i=0;i<=12;i++)printf(s[0],34,s[i],34,44,10);
printf(s[2],34,34,125,s[3],10,s[4],10,s[5],10,s[6],10,41,s[7]);
for(i=0;i<=12;i++)printf(s[0],39,s[i],39,44,10);
printf(s[12],39,39,41,s[8],10,s[9],10,s[10],10,s[11],10,47,125);
/*)const q=#34;w=#39;n=#13#10;s=array[0..13] of string=('%c%c%c%c',
'(*a);main(){char i,*s[]={',
'%c%c%c%c%c%c%c%c%c%c/%*c%c',
';printf(s[1]);for(i=0;i<=12;i++)printf(s[0],34,s[i],34,44,10);',
'printf(s[2],34,34,125,s[3],10,s[4],10,s[5],10,s[6],10,41,s[7]);',
'for(i=0;i<=12;i++)printf(s[0],39,s[i],39,44,10);',
'printf(s[12],39,39,41,s[8],10,s[9],10,s[10],10,s[11],10,47,125);',
'const q=#34;w=#39;n=#13#10;s=array[0..13] of string=',
';var i:integer;begin write(s[1]);for i:=0to 12do write(q,s[i],q,#44,n);',
'write(#34#34#125,s[3],n,s[4],n,s[5],n,s[6],n,#47#42#41,s[7]);',
'for i:=0to 12do write(w,s[i],w,#44,n); ',
'write(#39#39#41,s[8],n,s[9],n,s[10],n,s[11],n,#123#42#47#125);end.',
'%c%c%c%c%c%c%c%c%c%c/%*c%c',
'');var i:integer;begin write(s[1]);for i:=0to 12do write(q,s[i],q,#44,n);
write(#34#34#125,s[3],n,s[4],n,s[5],n,s[6],n,#47#42#41,s[7]);
for i:=0to 12do write(w,s[i],w,#44,n);
write(#39#39#41,s[8],n,s[9],n,s[10],n,s[11],n,#123#42#47#125);end.
{*/}
```

"hello polyglot" in 7 Sprachen



---cut here--(two blank lines are MANDATORY)---

```

CuG  #** )pop mark/CuG 4 def/# 2 def%%%%@P[TX---P\P_SXPY!Ex(mx2ex("SX!Ex4P)Ex=
CuG  #**                                     *+Ex=
CuG  #**-----*+Ex=
CuG  #** POLYGLLOT - a program in seven languages      15 February 1991 *+Ex=
CuG  #**                                     *+Ex=
CuG  #** Written by Kevin Bungard, Peter Lisle, and Chris Tham *+Ex=
CuG  #**                                     *+Ex=
CuG  #** We have successfully run this program using the following: *+Ex=
CuG  #** ANSI COBOL:      MicroFocus COBOL85 (not COBOL74) *+Ex=
CuG  #** ISO Pascal:      Turbo Pascal (DOS & Mac), Unix PC, *+Ex=
CuG  #**                                     AIX VS Pascal *+Ex=
CuG  #** ANSI Fortran:    Unix f77, AIX VS Fortran *+Ex=
CuG  #** ANSI C (lint free): Microsoft C, Unix CC, GCC, Turbo C++, *+Ex=
CuG  #**                                     Think C (Mac) *+Ex=
CuG  #** PostScript:      GoScript, HP/Adobe cartridge, *+Ex=
CuG  #**                                     Apple LaserWriter *+Ex=
CuG  #** Shell script:    gnu bash, sh (SysV, BSD, MKS), ksh *+Ex=
CuG  #** 8086 machine language: MS-DOS 2.00, 3.03, 4.01, 5.00 beta *+Ex=
CuG  #**                                     VPix & DOS Merge (under unix) *+Ex=
CuG  #**                                     SoftPC (on a Mac), MKS shell *+Ex=
CuG  #**                                     *+Ex=
CuG  #** Usage: *+Ex=
CuG  #** 1. Rename this file to polyglot.[cob|pas|f77|c|ps|sh|com] *+Ex=
CuG  #** 2. Compile and/or run with appropriate compiler and *+Ex=
CuG  #** operating system *+Ex=
CuG  #**                                     *+Ex=
CuG  #** Notes: *+Ex=
CuG  #** 1. We have attempted to use only standard language features. *+Ex=
CuG  #** Without the -traditional flag gcc will issue a warning. *+Ex=
CuG  #**                                     *+Ex=
CuG  #** 2. This text is a comment block in all seven languages. *+Ex=
CuG  #**                                     *+Ex=
CuG  #** 3. When run as a .COM file with MS-DOS it makes certain *+Ex=
CuG  #** (not unreasonable) assumptions about the contents of *+Ex=
CuG  #** the registers. *+Ex=
CuG  #**                                     *+Ex=
CuG  #** 4. When transferring from Unix to DOS make sure that a LF *+Ex=
CuG  #** is correctly translated into a CR/LF. *+Ex=
CuG  #**                                     *+Ex=
CuG  #** Please mail any comments, corrections or additions to *+Ex=
CuG  #** peril@extro.ucc.su.oz.au *+Ex=
CuG  #**                                     *+Ex=
CuG  #**-----*QuZ=
CuG  #**                                     *+Ex=
CuG  #**!Mx)ExQX4ZPZ4SP5n#5X!)Ex+ExPQXH,B+ExP[-9Z-9Z)GA(W@'UTTER_XYZZY'CPK**
CuG  #**(*) * (
C  # */); /*(
C  # *) program polyglot (output); (*+
C  # identification division.
C  # program-id. polyglot.
C  #
C  # data division.
C  # procedure division.
C  #
C  # *)cleartomark /Bookman-Demi findfont 36 scalefont setfont (
C  # *
C  #
C  # * hello polyglots$
C  #
C  # main.
C  # perform
C  # * ) 2>_$$$; echo "hello polyglots"; rm _$$$; exit
C  # print
C  # stop run.
C  # *, 'hello polyglots'

```

```
C      print.
C      display  "hello polyglots".
C      /* int i;
C      /* main () {
C      /*      i=printf ("hello polyglots\n"); 0= &i; return *0;
C      *)
C      *) begin
C      *)      writeln ('hello polyglots');
C      *)
C      *) pop 60 360
C      *) pop moveto (hello polyglots) show
C      *) pop showpage
C      *)
C      end
C)pop%      program      polyglot.
```



Kombi-Disziplin: Obfuscation und Dreier-Quine



- Ebenfalls Eingereicht zum IOCCC
- Autor: Don Yang
- `saitou.c` ist ein Bild von Saitou Hajime
- Es generiert einen Quine-Dreierzyklus, die Bilder Saitous Motto symbolisieren:
 - `aku` (sin - Sünde)
 - `soku` (swift - schnell, geschickt)
 - `zan` (slay - töten)

saitu.c



```

#define/**/X
char*d="X0!4cM,! "
"4cK!*!4cJc(!4cHg&!4c$ j"
"8f'!&-!9e)! '|:d+)rAc-! *m"
":d/!4c(b4e0!1r2e2!/t0e4!-y-c6!"
"+|,c6!)f$b(h*c6!(d'b(i)d5!(b*a'&c"
")c5! 'b+'&b'c)c4!&b-_$c'd*c3!&a.h'd+"
"d1!%a/g'e+e0!%b-g(d.d/!&c'h'dld-!(d%g)"
"d4d+!*1,d7d)! ,h-d;c'!.b0c>d%!A`Dc$![7]35E"
"! '1cA,,!2kE`*!-s@d(! (k(f//g&! )f.e5'f(1+a)"
"f%2g*! ?f5f,! =f-*e/! <d6e1!9e0'f3!6f)-g5!4d*b"
"+e6!0f%k)d7!+~^'c7!)z/d-+!'n%a0(d5!%c1a+/d4"
"!2)c9e2!9b;e!18b>e/! 7cAd-!5fAe+!7fBe(!"
"8hBd&! :iAd$![7S,Q0!11 bF 7!1b?'_6!1c,8b4"
"!2b*a,*d3!2n4f2!$ {4 f. '!%y4e5!&f%"
"d-^-d7!4c+b)d9!4c-a 'd :!/i('`&d"
";!+! 'a+d<! )1*b(d=! ' m- a &d>!&d'"
" '_0_&c?!$dAc@!$cBc@!$ b < ^&d$`"
":!$d9_&l+^+&$!%f3a' nl - $ !&"
"%f/c(o/_%!(f+c)q*c %! * f &d+"
"%f$&!-n,d)n(10i- c- k) ! 3d"
"/b0h*!H'7a,! [7* i] 5 4 71"
" [=chr&o*t*q`*`d *v *r ; 02"
"7*~h./)tcrsth &t : r 9b"
"] ,.b-725-.t--// #r [ < t8-"
"752793? <.-;b ] .t--+r / # 53"
"7-r[/9-X .v90 <6/<.v;-52/={ k goh"
"./)q; u vto hr `i*$engt$ $ ,b"
";$/ =t ;v; 6 =`it.`;7= ` : ,b-"
"725 = / o`. .d ;b]`--[/+ 55/ }o"
"`.d : - ?5 / }o`. ' v/i]q - "
"-[; 5 2 =` it . o;53- . "
"v96 <7 / =o : d =o"
"--/i ]q-- [; h. / = "
"i]q-[ ;v 9h ./ < - "
"52={cj u c&` i t . o ; "
"?4=o:d= o-- / i ]q - "
"-[;54={ cj uc& i]q - -"
" [;76=i]q[;6 =vsr u.i / ={"
"=),BihY_gha ,)\0 " , o [
3217];int i, r,w,f , , ,x ,
p;n(){return r <X X X X
768?d[X(143+ X r++ + *d ) %
768]:r>2659 ? 59: ( x ) = d
[(r++-768)% X 947 + 768] ) ?
x^(p?6:0):(p = 34 X X X)
}s(){for(x= n (); ( x^
?6:0))=32;x= n (); ;return x ;}
void/**/main X () { r = p
=0;w=sprintf (X X X X X X
,"char*d="); for ( f=1;f <
+143;)if(33-( b=d [ f++ X
){if(b<93){if X(! p )
[w++] =34;for X(i = 35
(p?0:1);i<b; i++ ) o
[w++] =s();o[ w++ ]
=p?s():34;} else X
{for(i=92; i<b; i
++)o[w++] = 32;}
else o [w++]
=10;o [
w]=0 ;
puts(o);}

```

aku.c



```

char*d=")35E!1cA,,!"2kE`*!-s@d(!k(f//g&!).e5'f(!+a+)f%2g*!??f5f,!=-*e/!<d"
"6e1!9e0'f3!6f)-g"      "5!4d*b+e6!0f%k)d7!+-^'c7!)z/d-+!'n%a0(d5!""%cla+/d4"
"12)c9e2!9b;e1!8b"      ">e/!7cAd-!5fAe+!7fEb(!8hBd&!::iAd$![7s,Q"      "0!1bF7"
"!1b?'_6!1c,8b4!2"      "b*a,*d3!2n4f2!$[4f.'!%y4e5!&f&d-^_d7!4"      "c+b)"
"d9!4c-a'd:!/i(')"      "&d;+!1'a+d<! )l*b(d='m-a&d>!&d'!0_&c"      "?!"
"$dAc@!$cBc@!$b<^"      ""      "&d$':!$d9_&l+++!$!%f3"      "a'n"
"1"      ""      "$!&f/c(o/!(f+c)q*c%"      "!"f&d"
"+f$"      "s&!-n,d)n(!0i-c-k)!3d/"      "b0h*!H"
"7a,!["      "7X0[!4cM,!4cK`*!4cJc(!4cHg&!4"      "c$]8f'!&-"
"19e)!*"      "|:d+!)"rAc-!*"m:d!4c(b4e0!1r"      "2e2!/t0e4!-"
"y-c6!+|,c6!)f$b("      "h*c6"      "!(d'b(i)d5!(b*a'"      "!"&c)c5!'b+'&"
"b'c)c4!&b_&c"      "d*c3!&a.h'd+d"      "!"%a/g'e+e0!%b"
"-g(d.d/!&c*"      "h'did-!(d%g)"      "d4d+!*1,d7d)!,-"
"d;c'!.b0c"      "d%!A'Dc$"      "![7*i]5471[=ohr&o*"
"t*q*!d"      ""      ""      ""      "v*r;027"      "=-h./)tcrsth&t:r9"
"b].,b"      ""      "-725"      ""      "-t--/"      "/#r[<t8-752793<.-"
";b)."      "t--"      "+z/#"      "537-r"      "[/9-X.v90<6/<.v;-"
"52/= "      "{kgoh."      "}/q"      "vtchr"      ".i*$engt$$,b;$/= "
"t;v"      ";6='it."      "};"      "7="      "};,b-7"      "25=/o`.d;b]'--["
"/+"      "55/}o`.d:"      ""      "-?5"      "}/o`.r"      "v/i]q--[;52='it"
".o"      ";53-.v96<7"      ""      "/=o"      "d=0-/"      "i]q--[;h./=i]"
"q"      "--[;v9h./<- "      "52="      "{cjuc&"      "it.o;?4=o:d;"
"o"      "--/i]q-[-"      "54="      "{cjuc&i]q-"      "-[;76=i]q[;"
"6="      "vsru.i"      "}/={ "      "}=,BihY_gha,)"      ""      "o[3217];"
int i,      r,w      ,f,b,p,      t=64l,x;n(){return      r<t?d[(*
d+143+(r      ++))%t]:r>      +1341?59:(x=d[(x++t)      %351+t]
)?x^(p?6:0      ): (p+=34);}      main(){w=sprintf(o,"char"      "%d="
);r=p=0;for(      f=1;f<d+143;)      if((b=d[f++])-33){if(b<+93){      if(
!p)o[w++]=34;for(i=35+(p?0:1);i      <b;i++)o[w++]=n();o[w++]=p;n():+34      ;
}
else for(i=92;i<b;i++)o[w++]=32;}else o[w++]=10;o[w]=0;puts(o);};/*Don_Yang*/;

```


soku.c



```

char*d="s,Q0!1bF7!1b?'_"6!1c,8b4!2b*a,*d3""!2n4f2!${4f.'!%y4e5!&f%d-^-d7!4c+"
"b)d""9!4c-a'd:!/i("      "/&d;!+!l'a+d<!)"1*b(d=!m-a&d>""!&d'0_&""c?!$dAc@"
"!$cBc@!$b<^&d$"      "":!$d9_&l+^$!%f3a'n!_!&f/c(o/_%"      "!(f+c)"
"q*c%!*f&d"      "+f$$s&!-n,d)n(!0i-c-k)!3d/b0h"      "**!H\"
"7a,!\"      "[7X0"      "[!4cM,!4cK\"""!4cJc(!4cH"      "g&"
"!4c$j"      "8f'!&~!9e)"      "!'|:d+!)rAc-!m*:"d/!"      "4c(b)"
"4e0!1r2"      "e2!/t0e""41-y-"      "c""6!+|,c6!f$b(h"      "*c6!(d)"
"b(i)d5!(b*a'\&c)c5!'b+\&b'"      "c)c4!&b-_c'd*c3!"      "&a.h'd+d"
"!%a/g'e+e0!%b-g(d.d/!&c*"      "h'd!d-!(d""%g)d4d"      "+!*!1,d7d)!",
"h-d;c'.!b0c>d%!A'Dc$!{7)"      "35E!'lCA,!2kE'!*!-"      "s@d(!k(f//g&"
"!f.e5'f(!+a+)f%2g*!?"      "f5f,!f-*e/!""<d6"      "e!19e0'f3!6f)-g"
"5!4d*b+e6!0f%k)d7!"      "+^^'c""7!)z/d-+!"      "'n%a0(d5!%c!a+/d4"
"!2)c9e2!9b;e!18b"      ">e/!7c"      "Ad-!5fA"      "e+!7fBe(!8hBd&!:iA"
"d$!{7*i]5471"      "["      "=ohr&"      "o*t*q''*d*v*x;027*~"
"=h./}tc"      "xst"      "h&t:r9b].,b-725-.t-<"
"-//#r"      "[<t8-752793"      "?<.;b].t""--r/#5"
"37-"      "[/9~X.v90<6/"<.v;"      "-52/={kgoh./}q;uv"
"t"      "chr`.i*$engt$"      "$,b;$/= ""t;v;6='it.\""      ";7=':,b-725=/o\"
".d;b)\--[/+55/"o`.d"      ":-?5/)o`.v/i]q--[;52"      "='it.o;53-.v96"
"<7/=:o:d=o--/i]q--[;h."      "/=i]q--[;v9h./<-52={cju"      "c&'it.o;?4=o:"
"d=o--/i]q--[;54={cju"      "c&i]q--[;76=i]q[;6=vsru.i/"      "={=),BihY_g"
"ha,)",o[3217];int i,      r,w,f,b,p,t=641,x;n(){return r<      t?d[(*d+143
+(r++)%t];r>+1341?      59:(x=d[(r++-t)%351+t])?x^(p?6:      0):(p=+34
);}main(){w=sprintf(o      ,"char"*d=");r=p=0;for(f=1;f<+d      +143;)
if((b=d[f++]-33){if(b      <+93){if(!p)o[w++]=34;for(i=35+(      p?0:
1);i<b;i++)o[w++]=n();o[      w++]p?n(:+34;}else for(i=92;i      <b
;i++)o[w++]=32;}else o[w++]=10;o[w]=0;puts(o);}/*Don_Yang*/;;;;;;;;;;;;;

```

zan.c



```

char*d="X0[!4cM,""14cK*!4cJc(14cHg&14c$J8f'!&~]9e)!'|:d+!)rAc-"!*"m*:d/!4c(b"
"4e0!1r2e2!/'t0"      "e4!-y-c6!+|,c6!)f$b(h*c6!(d'b(i)d5"!!(b*a'\&c)c5!'b+'&b"
"/c)c4!&b-_$c"      "d*c3!&a.h'd+d1!%a/g'e+e0!%b"-g("      "d.d/!&c*h'dld-!(d%"
"gd4d+!*1,d7d"      ")!h-d;c""!l.b0c>d%!A'Dc$![7]3"      "5E!'1cA,,12kE'!*1"
"-s@d!(k(f//g&"      "!)f.e5"      "'f(!+a)"f%2g*!"      "?f5f,!=-*e/!<d"
"6e119e0'f3!6f)"      "-g5!4d*b+e6!0f%k"      ")d7!+~^'c7!)z/"
""      "d-+!'n%a0(d5!c1"      "a+/d4!2)c9""e2!"
"9"      "b;e!18b>e/!7cAd-"      "!"5fAe+!7fBe(!8hBd"
"&!"      ":"      "iAd$![7S,"      "Q0!lbF7!1"      "b?'_6!1c,8b4!2b*a,*"
"d3!2n4f2!$ {4f.'!"      "%y4e5!&"      "f%d-^"      "-d7!4c+b)d9!4c-a'd:!/!"
"i('\&d;!+l'a+d<!"      ")l*b(d=!' "      "m-a"      "&d>!&d'\0_&c?!$dAc@!$c"
"Bc@!$b<&d$"      "':!$"d9"      "&l"      "++^$!%f3a'n1_$!&f/c(o/_"
"%!(f+c)"      "q*c"      "%!*f&d+"      "f$&!--n,d)n(10i-c-k)!3d/"
"b0h*!"      "H'7a,!"      "[7*i"      "]5471[=chr&o*t*q'*d*v*r;"
"027"      "*~h./tc"      "rs"      "th&t:r9b].,b-725-.t--//#r["
"<t"      "8-7"      "52793?<.-;b]"      ".t"      "--+r/#537-r[/9-X.v90<6/<.v;"
""      "-52/={kgoh./}q;uvtohr'.i*$eng"      "t$$,b;$/=t;v;6='it.';7=':;b-"
""      "725=/o\..d;b]'--[/+55/]o\..d:-?"      "5/)o\..v/i]q--[;52='it.o;53-"
""      ".v96<7/=o:d=o--/i]q--[;h"      "./"      ""      "]=i]q--[;v9h./<-52={cju"
""      "c&\it.o;?4=o:d=o--/i]"      "q-"      "-[;54={""cjuc&i]"      ""
"q"      "-[;76=i]q[;6=v"      "sru"      ".i/={,}BihY_"      ""
"gh"      "a,)",o[3217]      ;int i      ,r,w,f,b,p,t= 641
,x;n()      {return r      <t?d[(*      d+143+(r      ++
)%t]:r>+ 1341      ?59:(x=d[      (r      ++t
)%351+t]}?x      ^{p?6:0):(      p=+34;      }main(
){w=sprintf(o,      "char"*d="      );r=p=0;for      (f=1;f<
*d+143;){if((b=d[f      ++])-33){if(b      <+93){if(!p)o[      w++]=34;
for(i=35+(p?0:1);i<b;i++)o[w++]=n();o[      w++]=p?n():+34;}else      for(i=92;i
<b;i++)o[w++]=32;}else o[w++]=10;o[w]=0;puts(o);};/*Don_Yang*/;;;;;;;;;;;;;

```



Weiteres z. B.:

- Selbstmodifizierender Code
 - Camian:
"The only thing better than self-modifying code is code that modifies itself before it does so."
- Variablen-Superposition
 - Perl CPAN Modul `Quantum::Superpositions`
 - ```
if (a==23 && a==42) { blubb(); }
```
- Seltsame Umgebungen
  - Alles programmieren, was annähernd Befehle ausführen kann
  - z. B. Tetris in Openfirmware
  - [Blinkenlights](#): Häuserfront als Bildschirm



## Literatur (hoffentlich vollständig)

- <http://www.google.com/> - die Suchmaschine meiner Wahl

- <http://www.blinkenlights.de/> - Häuserfront als Computerdisplay
- <http://www.ioccc.org/> - International Obfuscated C Coding Contest
- <http://www.mines.edu/students/b/bolmstea/quentes/> - the Quines List
- <http://www2.latech.edu/~acm/HelloWorld.shtml> - the ACM Hello World page
- <http://pauillac.inria.fr/caml/FAQ/> - the Caml FAQ
- <http://www.cliff.biffle.org/esoterica/beatnik.html> - The Beatnik Language
- <http://www.cliff.biffle.org/esoterica/hq9plus.html> - The HQ9+ programming language
- <http://99-bottles-of-beer.ls-la.net/> - über 570 mal 99 Bierflaschen
- <http://www.selectorweb.com/perl.html> - selectorweb Perl seite
- <http://www.stud.uni-hannover.de/~freise/ascii/pqr/perl.txt> - U. a. das Perl-Camel
- <http://www.catseye.mb.ca/esoteric/bf/> - "die" Brainfuck Homepage
- <http://www.muppetlabs.com/~breadbox/bf/> - eine weitere Seite zu Brainfuck
- <http://koeln.ccc.de/projekte/brainfuck/index-e.html> - Brainfuck Projekt des CCC Köln
- <http://cydathria.com/bf/brainfuck.html> - Programming in Brainfuck - Introduction
- <http://www.muppetlabs.com/~breadbox/intercal-man/> - the InterCal Programming Manual
- <http://www.catseye.mb.ca/esoteric/befunge/> - Befunge Homepage
- [http://www.p-nand-q.com/humor/programming\\_languages/java2k.html](http://www.p-nand-q.com/humor/programming_languages/java2k.html) - Java2k Homepage



- <http://www.kraml.at/stupid/> - Esoteric Language Database
- <http://www.dangermouse.net/esoteric/chef.html> - The Chef Programming Language
- <http://mindprod.com/unmain.html> - Unmaintainable Code Howto
- [compsoc.dur.ac.uk/whitespace/](http://compsoc.dur.ac.uk/whitespace/) - the WhiteSpace Homepage
- Linux Magazin, Ausgabe 12/2003, Artikel "Getarnte Quellen", Seite 95ff



# Ende

- Vortrag ist im HTML-Format online
  - <http://ulm.ccc.de/~schabi/weirdprog20c3/>
  - Kontakt: XXXXXXXXXX
- Verwendete Software:
  - Opera
  - nedit
  - Gimp
  - Debian Sarge Linux

