

# Weird Programming 2

## continous headache

Lecture for the 21<sup>th</sup> Chaos Communication Congress, 27<sup>th</sup>-29<sup>th</sup> Dec 2004, Berlin, Germany

Like in the last years first part of this lecture, examples of strange programming (art)work will be shown. In addition to the funny and sportive disciplines known from last year, some examples of painful production code will be presented. Online materials including a german version of the slides for this speech will be made available at <http://www.ulm.ccc.de/~schabi/weirdprog2/>.

The CCC ErfA Group Ulm is planning to hold a shortest C coding contest on this Congress. We learned our lesson from the last years contest, so the rules will be much simplified.

### **Abstract:**

The first part of the presentation will - similar to last years presentation - shed some light on the funny and sportive disciplines of the art of programming. Besides new examples in disciplines that were presented last year, like obfuscated programming and shortest code, core wars and demo coding are new in the agenda.

In the second part of the lessons, some creatively designed programming languages will be introduced. Especially, the two projects "Argh!" and "repsub" will be presented. Both of them evolved in the orbit of the CCC.

Finally, the third part will introduce some extra painful examples of production code. A fertile source for those are some commercially developed projects that were open-sourced afterwards. From time to time, those create the impression that the developers lost control of their own code. They now hope the community will help them to find the way out of their maintainance nightmare.

### **17<sup>th</sup> IOCCC**

The 17<sup>th</sup> IOCCC (International Obfuscated C Coding Contest) started at 7<sup>th</sup> Jan 2004. Although the IOCCC is announced as annually, you may have noticed that the 16<sup>th</sup> IOCCC took place in 2001. It had some delays so it was finished in 2002, and the jury seemed to be temporarily short on ressources so the 17<sup>th</sup> IOCCC which was planned to be hold in 2003 really started in 2004. This mkes one wonder whether the CCC part in IOCCC has something in common with the CCC, or at least that they could add a 4<sup>th</sup> C for chaotic.

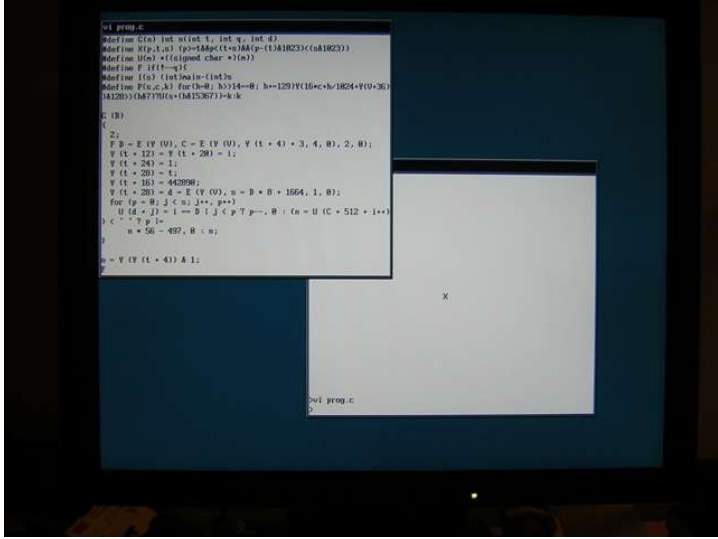
This is the source of one of the winners, the "Best of Show" entry:

```

1 #define G(n) int n(int t, int q, int d)
2 #define X(p,t,s) (p>=t&&p<(t+s)&&(p-(t)
3 #define U(m) *((signed char *) (m))
4 #define F if(!--q){
5 #define I(s) (int)main-(int)s
6 #define P(s,c,k) for(h=0; h>>14==0;
7 h+=129)Y(16*c+h/1024+Y(V+36))&128>>
8 (h&7)?U(s+(h&15367)):k:k
9
10 G (B)
11 {
12     Z;
13     F D = E (Y (V), C = E (Y (V), Y (t +
14 4) + 3, 4, 0), 2, 0);
15     Y (t + 12) = Y (t + 20) = i;
16     Y (t + 24) = 1;
17     Y (t + 28) = t;
18     Y (t + 16) = 442890;
19     Y (t + 28) = d = E (Y (V), s = D * 8 +
20 1664, 1, 0);
21     for (p = 0; j < s; j++, p++)
22         U (d + j) = i == D | j < p ? p--,
23         0 : (n = U (C + 512 + i++) < ' ' ? p |
24         n * 56 - 497, 0 : n;
25 }
26
27 n = Y (Y (t + 4)) & 1;
28 F
29 U (Y (t + 28) + 1536) |=
30 62 & -n;
31 U (d + D) =
32 X (D, Y (t + 12) + 26628, 412162) ? X
33 (D, Y (t + 12) + 27653,
34 410112) ? 31 : 0 : U (d + D);
35 for (; j < 12800; j += 8)
36     P (d + 27653 + Y (t + 12) + ' ' * (j &
37 ~511) + j % 512,
38     U (Y (t + 28) + j / 8 + 64 * Y (t +
39 20)), 0);
40
41 F if (n)
42 {
43     D = Y (t + 28);
44     if (d - 10)
45         U (++Y (t + 24) + D + 1535) = d;
46     else
47     {
48         for (i = D; i < D + 1600; i++)
49             U (i) = U (i + 64);
50         Y (t + 24) = 1;
51         E (Y (V), i - 127, 3, 0);
52     }
53     else
54         Y (t + 20) += ((d >> 4) ^ (d >> 5)) -
55 3;
56 }
57
58 G (_);
59 G (o);
60 G (main)
61 {
62     Z, k = K;
63     if (!t)
64     {
65         Y (V) = V + 208 - (I (_));
66         L (209, 223) L (168, 0) L (212,
67 244) _ (int) &s, 3, 0);
68         for (; l;
69             R n = Y (V - 12);
70             if (C & ' ')
71             {
72                 k++;
73                 k %= 3;
74                 if (k < 2)
75                 {
76                     Y (j) -= p;
77                     Y (j) += p += U (&D) * (1 -
78 k * 1025);
79                     if (k)
80                         goto y;
81                     else
82                     {
83                         for (C = V - 20;
84                             !i && D & 1 && n
85                             && (X (p, Y (n + 12),
86 Y (n + 16)) ? j = n + 12, Y (C + 8) =
87 Y (n + 8), Y (V - 12), Y (V - 12) =
88 n, 0 : n); C = n,
89 n = Y (n + 8));
90                     i = D & 1;
91                     j &= -i;
92                 }
93             else if (128 & ~D)
94             {
95                 E (Y (n), n, 3, U (V + D % 64 +
96 131) ^ 32);
97                 n = Y (V - 12);
98                 y: C = 1 << 24;
99                 M U (C + D) = 125;
100                 o (n, 0, C);
101                 P (C + p - 8196, 88, 0);
102                 M U (Y (0x11028) + D) = U (C +
103 D);
104             }
105         }
106     }
107     for (D = 720; D > -3888; D--)
108         putchar (D >
109             ? "
110             " )! \320\234\360\256\370\256
111             \0\230F .,mbvxcz ;lkjhgfdsa
112             \n[poiuytrewq =-0987654321 \357\262
113             \337\337 \357\272 \337\337 ( )
114             \\"343\312F\320!\230 26!\16
115             K>!\16\332
116             \4\16\251\0160\355&\271\20\2300\355`x
117             {0\355\347\2560 \237qpa%\231o!\230
118             \337\337\337 ,
119             \\"K\240 \343\316qrpzxy\0
120             sRDh\16\313\212u\343\314grzy !0( "
121             [D] ^ 32 : Y (I (D)));
122     return 0;
123 }
124
125 G (o)
126 {
127     Z;
128     if (!t)
129     {
130         C = Y (t + 12);
131         j = Y (t + 16);
132         o (Y (t + 8), 0, d);
133         M U (d + D) =
134         X (D, C, j) ? X (D, C + 1025, j -
135 2050) ? X (D, C + 2050,
136 j - 3075) ? X (D,
137 C + 2050,
138 j -
139 4100) ?
140 X (D, C + 4100,
141 ((j & 1023) + 18424) ? 176 :
142 24 : 20 : 0 : U (d + D);
143         for (n = Y (t + 4); U (i + n);
144             i++)
145             P (d + Y (t + 12) + 5126 + i * 8,
146 U (n + i), 31);
147             E (Y (t), t, 2, d);
148         }
149     }
150     G (_);
151     {
152         Z = Y (V + 24);
153         F Y (V - 16) += t;
154         D = Y (V - 16) - t;
155         F for (i = 124; i < 135; i++)
156             D = D << 3 | Y (t + i) & 7;
157         if (q > 0)
158         {
159             for (; n = U (D + i); i++)
160                 if (n - U (t + i))
161                 {
162                     D += _ (D, 2, 0) + 1023 & ~511;
163                     i = ~0;
164                 }
165             F if (Y (D))
166             {
167                 n = _ (164, 1, 0);
168                 Y (n + 8) = Y (V - 12);
169                 Y (V - 12) = n;
170                 Y (n + 4) = i = n + 64;
171                 for (; j < 96; j++)
172                     Y (i + j) = Y (t + j);
173                 i = D + 512;
174                 j = i + Y (i + 32);
175                 for (; Y (j + 12) != Y (i + 24); j
176                     += 40);
177                 E (Y (n) = Y (j + 16) + i, n, 1,
178 0);
179             }
180         }
181     }
182     return D;
183 }

```

Can you guess what it is? When started, it produces a boot image and the root file system for a complete multitasking operating system, including a window system, a shell and a text viewer. Here's a screenshot:



## Compact C Coding Contest

As last year, the CCC Ulm holds a compact C coding contest this year. The rules and the problem description is available at <http://ulm.ccc.de/shortest> or in the congress wiki in the project pages.

The last years contest involved converting numbers from base 23 to base 11, in the Java2k format. The main problem was that the input numbers were allowed to have 42 digits which was way too much for 32-bit integer arithmetics. The winning Entry was submitted by Holger Kuhn who solved the problem in 401 chars of source:

```
#include <stdio.h>
#define y for(c=0,i=54;i>=0;c=(s[i]+=a[i]+c)/11,s[i]=s[i]%11,i--);
#define z for(i=0
main(){char s[55],a[55];int c,i,d;for(;;){z;i<55;s[i++]=0;}for(;(d=getchar
())!='\n';){d=d>58?(d>77?d-87:d-55):d-48;z;i<55;a[i]=s[i],i++;y z;i<54;s[i]
=s[i+1],i++);s[54]=0;y z;i<53;a[i++]=0);a[53]=d/11;a[54]=d%11;y}z;i<54&&s[i]
==0;i++);for(;i<55;i++)printf("%c",s[i]<10?s[i]+48:32);printf("\n");}}
```

After the contest, some participants and Jury members continued to improve their solutions, and the shortest solution so far produced is 204 chars short:

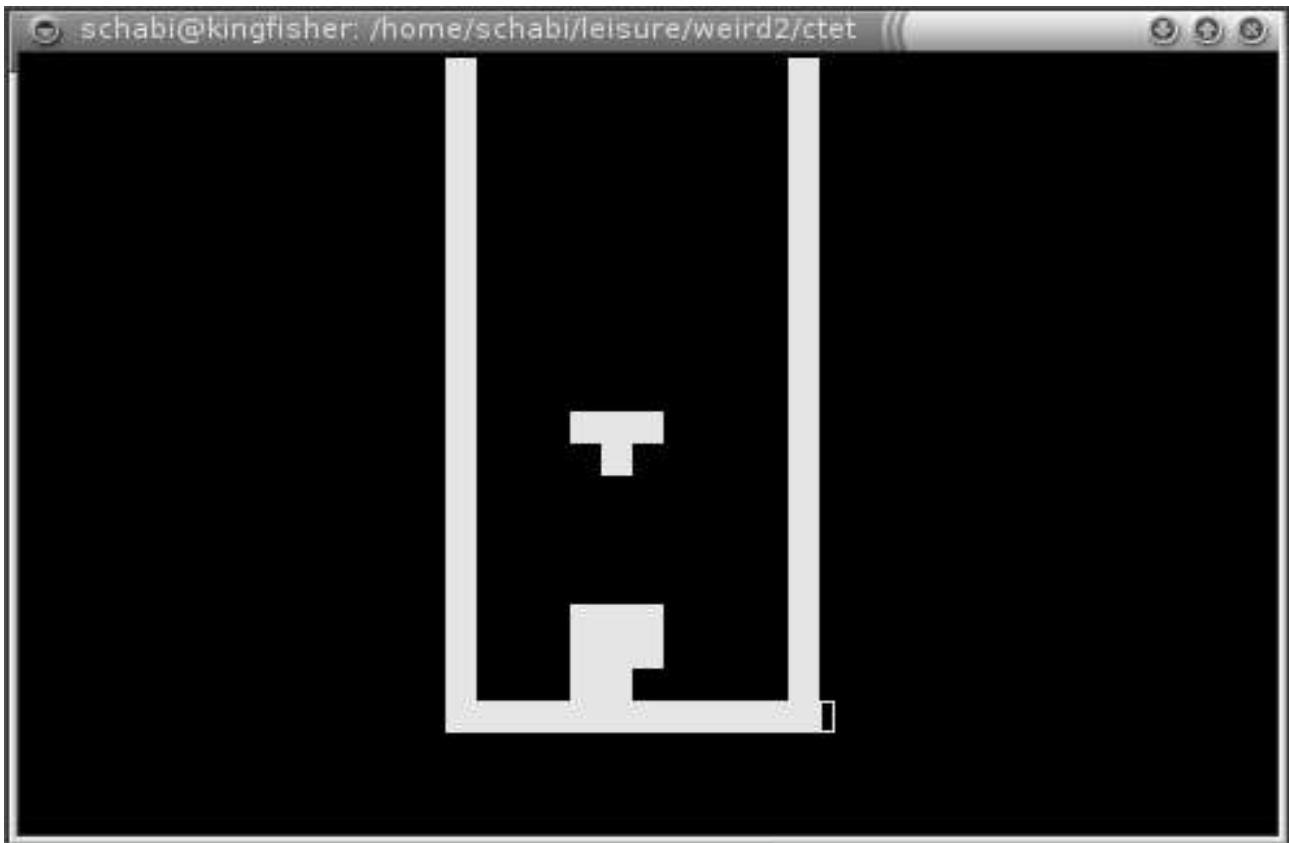
```
#include<stdio.h>
int
f[61],c,i;main(){for(;c=getchar()+1;*f--38)if(c>33)for(c=c<61?c-
49:8+c&31,i=1;i<60;f[i++]=c%11,c/=11)c+=f[i]*23;else
for(c=0;i>=0;f[i--]=0)if(c|=f[i]|i<2)putchar(f[i]>9?32:48+f[i]);}
```

## Compact something

Across some unreproducible ways, I got hands on the URL <http://wartha.informatik.uni-rostock.de/c7/compact.c> which points to the following C program:

```
long h[4];t(){h[3]-=h[3]/3000;setitimer(0,h,0);}c,d,l,v[]={(int)t,0,2},w,s,I,K
=0,i=276,j,k,q[276],Q[276],*n=q,*m,x=17,f[]={7,-13,-12,1,8,-11,-12,-1,9,-1,1,
12,3,-13,-12,-1,12,-1,11,1,15,-1,13,1,18,-1,1,2,0,-12,-1,11,1,-12,1,13,10,-12,
1,12,11,-12,-1,1,2,-12,-1,12,13,-12,12,13,14,-11,-1,1,4,-13,-12,12,16,-11,-12,
12,17,-13,1,-1,5,-12,12,11,6,-12,12,24};u(){for(i=11;++i<264;)if((k=q[i])-Q[i]
){Q[i]=k;if(i-++I||i%12<1)printf("\033[%d;%dH",(I=i)/12,i%12*2+28);printf(
"\033[%dm  "+(K-k?0:5),k);K=k;}Q[263]=c=getchar();}G(b){for(i=4;i--;)if(q[i?b+
n[i]:b])return 0;return 1;}g(b){for(i=4;i--;q[i?x+n[i]:x]=b);}main(C,V,a)char*
*V,*a;{h[3]=1000000/(l=C>1?atoi(V[1]):2);for(a=C>2?V[2]:"jkl pq";i;i--)*n++=i<
25||i%12<2?7:0;srnd(getpid());system("stty cbreak -echo stop u");sigvec(14,v,
11 0);t();puts("\033[H\033[J");for(n=f+rand()%7*4;g(7),u(),g(0)){if(c<0){if(G(x+
12))x+=12;else(g(7);++w;for(j=0;j<252;j=12*(j/12+1))for(;q[++j]);}if(j%12==10){
13 for(;j%12;q[j--]=0);u();for(--j;q[j+12]=q[j]);u();}n=f+rand()%7*4;G(x=17)||c
14 =a[5]);}if(c==a)G(--x)||++x;if(c==a[1])n=f+4** (m=n),G(x)|| (n=m);if(c==a[2])G
15 (++x)||--x;if(c==a[3])for(;G(x+12);++w)x+=12;if(c==a[4]||c==a[5]){s=sigblock(
16 8192);printf("\033[H\033[J\033[0m%d\n",w);if(c==a[5])break;for(j=264;j--;Q[j]=
17 0);while(getchar()-a[4]);puts("\033[H\033[J\033[7m");sigsetmask(s);}d=popen(
18 "stty -cbreak echo stop \023;sort -mnr -o HI - HI;cat HI","w");fprintf(d,
19 "%4d from level %1d by %s\n",w,l,getlogin());pclose(d);}
```

As the following Screenshot demonstrates, it is a fully working, playable tetris, it even includes a high score table.



## ***RepSub – Repeated Substitutions***

RepSub is based on pattern matching and text substitution. Repsub has the high ideal to be a democratic programming environment. There are no hierarchies, all memory cells enjoy equal rights, and can be processed highly parallel. There is no official home page yet, but the Inventor Marco Haschka gave me the permission to put the specs and reference implementation online, so you will find this material on this lectures information page.

RepSub has several advantages over other languages. It contains even less symbols than brainfuck, and is simpler than Forth. This makes it easier to grasp than most other languages. It is also considered to be more elegant compared to Intercal, and to be potentially more cryptic than C. This combination is a powerful tool in the hands of a skilled programmer. Here's a small example, the usual Hello World:

```
s hello_-World!
```

The pattern language is not as expressive as extended regular expressions, but it is sufficient. It is proved that it is sufficient because RepSub is turing complete. The proof is straight forward – a standard turing machine can very easily be emulated. This is a small three-rule binary turing machine:

```
Input: 1011010111000001010start10001001000000010101
Program: start z001
        *?z001*? *0?61*11z002:z003*10;
        *?z002*? *0?61*11z003:z003*10;
        *?z003*? *0?61*11z003:z001*10;
```

The current implementation, which is based on matlab, is in productive use. A big German manufacturer of light bulbs that, surprisingly, does prefer not to be named, uses it for not further specified pattern matching based text processing.

## ***Argh! and Aargh!***

Argh! and its derivative Aargh! are somehow similar to BeFunge in that they are two dimensional



```
19 j Take one down, pass it around,  
20 lPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPfj  
21 s/ j  
22 lKhhhhhSSDPH  
23 lDfS1111K /  
24 jffdrspppppppppppppppppppppppppppppppppppppppph  
25 j *.llaw eht no reeb fo selttob  
26 lPP1DS1111111111111111111111111111111111111111ssK  
27 fJ
```

## Corewars

In core wars, we have a bunch of programmes running in parallel in the same memory. (This is a typical Von-Neumann machine with multitasking, but without memory protection.) The goal is to create a program that survives as long as possible, but at the same time quickly erases the other programs from memory. Another objective may be to own as much memory as possible for as long as possible.

The Corewar programmer is typically constrained by a simple, non-mighty assembly language, a source file limit and the fact that he has only a few CPU cycles before he's overridden by his enemies.

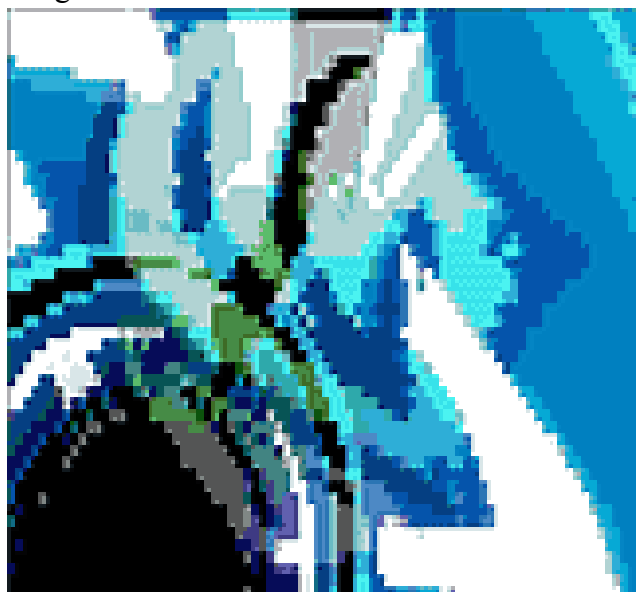
(The congress speech will contain a life demonstration.)

## Democoding

Demo coders try to exploit a given, limited (and often legacy) hardware through the use of crafty software, and thus create unexpected effects and surprising results. They try and often manage to achieve amazing sound effects and mind-blowing graphics in an extremely small executable size. Often, those demos run on ancient or otherwise limited hardware, and other restrictions may be imposed by the contest holders.

On so-called demo partys, those programs are presented, and sometimes even some high valued prizes are put up. The winners are e. G. 3D first person shooters in 64k and video clips with sound in 4k.

A rather interesting restriction is placed by the Text Mode Demo Contest which was just ran in its 7<sup>th</sup> incarnation. They limit the demo programmers to the VGA text mode. The surprising results are shown in the screenshots below. As DosBox 0.63 is one of the supported platforms, the speaker intends to give a life demo of the winning entries during the lesson.



## Strange Production Code

Most of the readers will remember situations where they stumbled over weird code written by others. Currently, FeFe plans to hold a round table following the Idea of the german TV show “Literarisches Quartett” that discusses some of those examples. While they focus on “bad” code, whatever that means, this speech will try to focus on “weird” code, whatever that means.

### *Mico is curious – maybe?*

Officially, it is told, Mico may be the abbreviation of “Mico is CORBA”. But this may be a bad excuse, and the real meaning may be rather “Mico is curious – maybe”. The reason for this assumption may be the following excerpt of Mico 2.3.11 source, file orb/except.cc, lines 348-360:

```
14  switch (_completed) {
15  case COMPLETED_YES:
16      os << "completed";
17      break;
18  case COMPLETED_NO:
19      os << "not-completed";
20      break;
21  case COMPLETED_MAYBE:
22      os << "maybe-completed";
23      break;
24  default:
25      assert (0);
26  }
```

### *To clean or not to clean*

But Mico is not the only software that seems to be unsure about its status. When we look at the GCC Makefile.in, we see some rather fuzzy targets, too:

- do-mostlyclean
- do-clean
- clean-gcc
- distclean-gcc
- maintainer-clean-gcc
- mostlyclean-gcc
- maybe-clean-gcc
- maybe-distclean-gcc
- maybe-maintainerclean-gcc
- maybe-mostlyclean-gcc

But there's no “still-somehow-dirty-gcc” target.

### *How to produce white space*

As most developers know, XML is extremely cool. And developers want to be cool. Or they have to due to marketing reasons. So every new project has to take part in the buzzword bingo and process some XML.

Furthermore, most of the developers distrust existing solutions and re-invent the wheel again and again. So they write their own XML output method. Of course, XML is a good candidate for pretty printed output, so our poor developers need some white space producing method that we can utilize to indent the lines of our output.

The developers of OpenMDX 1.4, which is a somehow usable open source MDA platform, must have followed the thoughts above, and produced the following method in lines 478 to 483 in their class org/openmdx/model1/layer/application/XMISchemaWriter.java:

```
478 private String spaces(
```





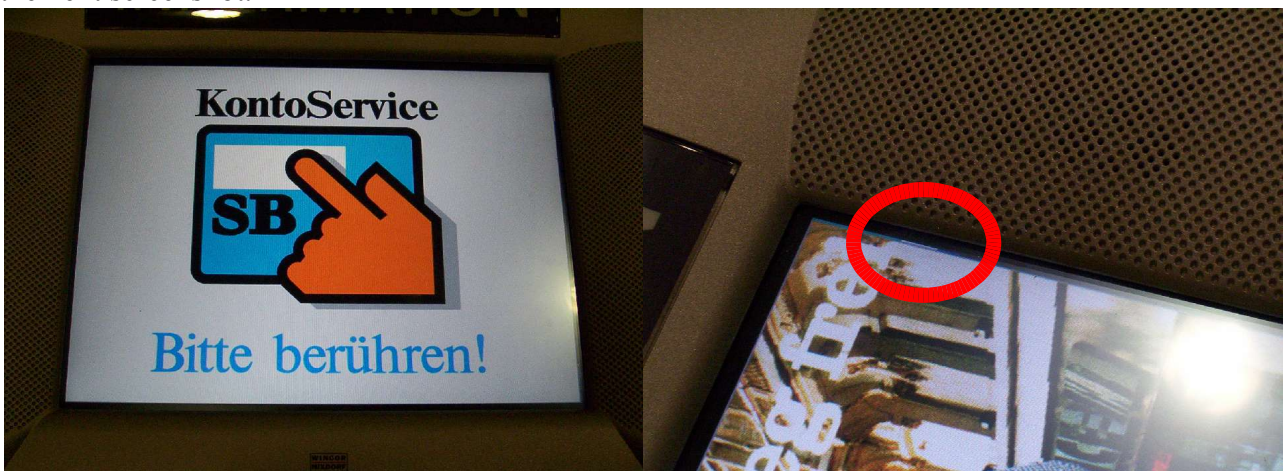
However, our friends from OpenMDX 1.4 managed to get it wrong. In class `org.openmdx.application.gui.generic.servlet.attribute.AttributeValue`, they make the superclass code behave differently depending on which class the concrete instance actually has. The correct way would be to encapsulate the code in a method, and then override this in the appropriate subclasses.

```
11: if(v == null) {
12:     s = "";
13: }
14: else if(this instanceof ObjectReferenceValue) {
15:     Action action = ((ObjectReference)v).getSelectObjectAction();
16:     s = "<a href=\"" + forView.getHRef(action) + "\">" + action.getTitle() +
17: "</a>";
18: }
19: else if(this instanceof BinaryValue) {
20:     Action action = (Action)v;
21:     s = "<a href=\"" + forView.getHRef(action) + "\">" + action.getTitle() +
22: "</a>";
23: }
24: else if(this instanceof BooleanValue) {
25:     s = ((Boolean)v).booleanValue()
26:     ? application.getTexts().getTrueText()
27:     : application.getTexts().getFalseText();
28: }
29: else {
30:     s = v.toString().trim();
31: }
```

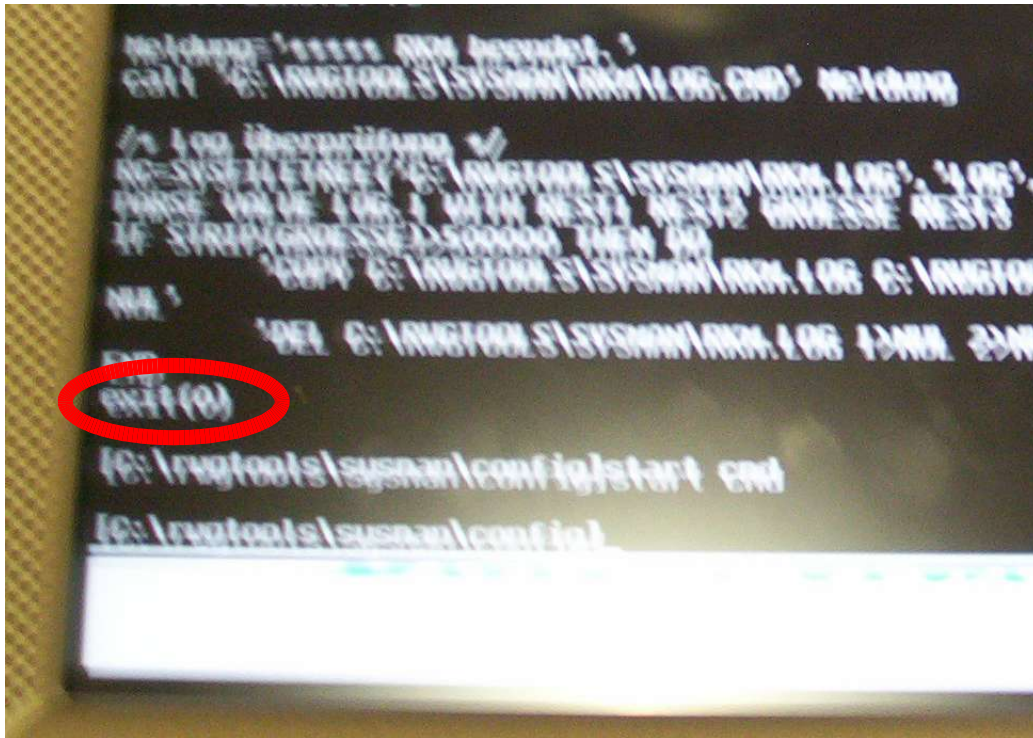
### ***The bank always wins?***

The Chaostreff Bad Waldsee, that belongs to the CCC Ulm, had some funny experiences with an info terminal at a bank. The photo series they took can be seen at <http://ulm.ccc.de/projekte/bankomat>, so I'll shortcut the story to the weird programming part.

The Screen says "Please touch" in big blue letters. This clearly is an invitation. The question is whether this invitation is also valid for those strange pixels at the right border which can be seen in the next screenshot.



Guess what – the menu window seems to be a few pixels too small to cover the whole desktop. And if you touch the artefacts at the border, an OS/2 command shell pops up. You know, OS/2 does not have any real security concept, and the info terminals have a keyboard to allow bank order forms to be filled in. So, you basically touch the screen and are root. The info terminal clearly is very informative this way. It even allows one to find out why the command shell is left open. It seems that there's an startup script that has a weird typing mistake. (Sorry for the bad picture.)



### ***Making a build system***

For evaluation purposes, a friend of mine tried to port the freshly open sourced SapDB aka MaxDB to LinuxPPC. In order to do this, he had to port the MaxDB build system first. As he described me this thing as rather weird, I wanted to show it in this presentation. But I failed to build it. `./configure` bails out with a shell syntax error in the first non-comment line.

But I can give you some enlightening insight nevertheless:

```
schabi@kingfisher:~/leisure/weird2/maxdb$ ls -o -h maxdb*
-rw-r--r--  1 schabi  26M 2004-11-30 22:54 maxdb-buildtools-source-518823.tgz
-rw-r--r--  1 schabi  24M 2004-11-30 23:22 maxdb-source-7_5_00_19.tgz
```

As you see, the build system (basically a make) is bigger than the software itself.

***Thanks for your patience.***