Weird Programming 2
continous headache

Lecture for the 21th Chaos Communication Congress, 27th-29th 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 lession 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.

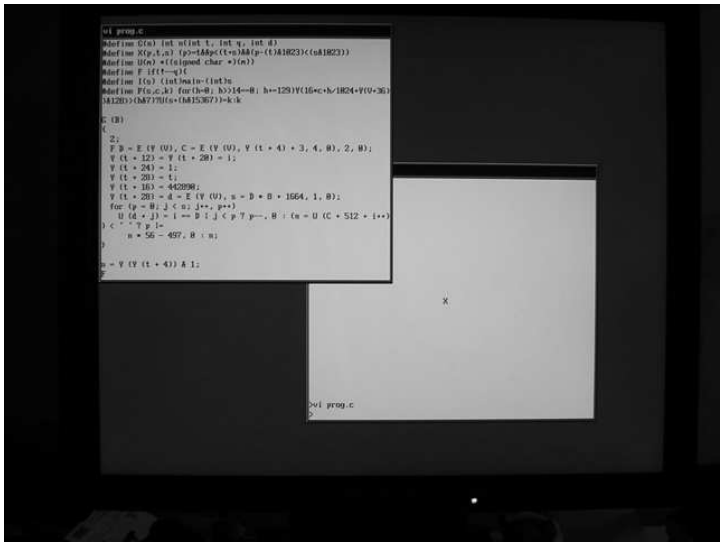
17th IOCCC

The 17th IOCCC (International Obfuscated C Coding Contest) started at 7th Jan 2004. Although the IOCCC is announced as annually, you may have noticed that the 16th 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 17th 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 4th C for chaotic.

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

```
#define G(n) int n(int t, int q, int d)
#define X(p,t,s) (p>=t&&p<(t+s)&&(p-(t)&1023)<(s&1023))
#define U(m) *((signed char *)(m))
#define F if(!--q){
#define I(s) (int)main-(int)s
#define P(s,c,k) for(h=0; h>>14==0; h+=129)Y(16*c+h/1024+Y(V+36))&128>>(h&7)?U(s+(h&15367))=k:k

G (B)
{
  Z;
  F D = E (Y (V), C = E (Y (V), Y (t + 4) + 3, 4, 0), 2, 0);
  Y (t + 12) = Y (t + 20) = i;
  Y (t + 24) = 1;
  Y (t + 28) = t;
  Y (t + 16) = 442890;
  Y (t + 28) = d = E (Y (V), s = D * 8 + 1664, 1, 0);
  for (p = 0; j < s; j++, p++)
    U (d + j) = i == D | j < p ? p--, 0 : (n = U (C + 512 + i++)) < ' ' ? p |= n * 56 - 497, 0 : n;
}

n = Y (Y (t + 4)) & 1;
F
U (Y (t + 28) + 1536) |=
62 & -n;
M
U (d + D) =
X (D, Y (t + 12) + 26628, 412162) ? X (D, Y (t + 12) + 27653,

410112) ? 31 : 0 : U (d + D);
for (; j < 12800; j += 8)
  P (d + 27653 + Y (t + 12) + ' ' * (j & ~511) + j % 512,
      U (Y (t + 28) + j / 8 + 64 * Y (t + 20)), 0);
}

F if (n)
  {
    D = Y (t + 28);
    if (d - 10)
      U (++Y (t + 24) + D + 1535) = d;
    else
      {
      for (i = D; i < D + 1600; i++)
        U (i) = U (i + 64);
      Y (t + 24) = 1;
      E (Y (V), i - 127, 3, 0);
      }
  }
else
  Y (t + 20) += ((d >> 4) ^ (d >> 5)) - 3;
}
}

G (_);
G (o);
G (main)
{
  Z, k = K;
  if (!t)
    {
      Y (V) = V + 208 - (I (_));
      L (209, 223) L (168, 0) L (212, 244) _((int) &s, 3, 0);
      for (; 1;)
      R n = Y (V - 12);
      if (C & ' ')
      {
        k++;
        k %= 3;
        if (k < 2)
          {
            Y (j) -= p;
            Y (j) += p += U (&D) * (1 - k * 1025);
            if (k)
                goto y;
          }
        else
          {
            for (C = V - 20;
                   !i && D & 1 && n
                   && (X (p, Y (n + 12), Y (n + 16)) ? j = n + 12, Y (C + 8) =
                         Y (n + 8), Y (n + 8) = Y (V - 12), Y (V - 12) =
                           n, 0 : n); C = n, n = Y (n + 8));
            i = D & 1;
            j &= -i;
          }
      }
      else if (128 & ~D)
      {
        E (Y (n), n, 3, U (V + D % 64 + 131) ^ 32);
        n = Y (V - 12);
      y:C = 1 << 24;
        M U (C + D) = 125;
        o (n, 0, C);
        P (C + p - 8196, 88, 0);
        M U (Y (0x11028) + D) = U (C + D);
      }
    }
}

for (D = 720; D > -3888; D--)
  putchar (D >
           0 ?
           "  )!\320\234\360\256\370\256\230F           .,mnbvcxz    ;lkjhgfdsa\n][poiuytrewq  =-0987654321   \357\262\337\337 \357\272   \337\337         ( )\"\343\312F\320!/ !\230 26!/\16K>!/\16\332\4\16\251\0160\355&\2271\20\2300\355`x{0\355\347\2560 \237qpa%\231o!\230\337\337\337     ,                )\"K\240   \343\316qrpxzy\0sRDh\16\313\212u\343\314qrzy    !0( "
         [D] ^ 32 : Y (I (D)));
return 0;
}

G (o)
{
  Z;
  if (t)
    {
      C = Y (t + 12);
      j = Y (t + 16);
      o (Y (t + 8), 0, d);
      M U (d + D) =
      X (D, C, j) ? X (D, C + 1025, j - 2050) ? X (D, C + 2050,
                       j - 3075) ? X (D,
C + 2050,
j -
4100) ?
      X (D, C + 4100,
         ((j & 1023) + 18424)) ? 176 : 24 : 20 : 28 : 0 : U (d + D);
      for (n = Y (t + 4); U (i + n); i++)
      P (d + Y (t + 12) + 5126 + i * 8, U (n + i), 31);
      E (Y (t), t, 2, d);
    }
}

G (_)
{
  Z = Y (V + 24);
  F Y (V - 16) += t;
  D = Y (V - 16) - t;
}

F for (i = 124; i < 135; i++)
  D = D << 3 | Y (t + i) & 7;
}

if (q > 0)
  {
    for (; n = U (D + i); i++)
      if (n - U (t + i))
      {
        D += _(D, 2, 0) + 1023 & ~511;
        i = ~0;
      }
    F if (Y (D))
      {
      n = _(164, 1, 0);
      Y (n + 8) = Y (V - 12);
      Y (V - 12) = n;
      Y (n + 4) = i = n + 64;
      for (; j < 96; j++)
        Y (i + j) = Y (t + j);
      i = D + 512;
      j = i + Y (i + 32);
      for (; Y (j + 12) != Y (i + 24); j += 40);
      E (Y (n) = Y (j + 16) + i, n, 1, 0);
      }
  }
}

return D;
}
```

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 avialable 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:

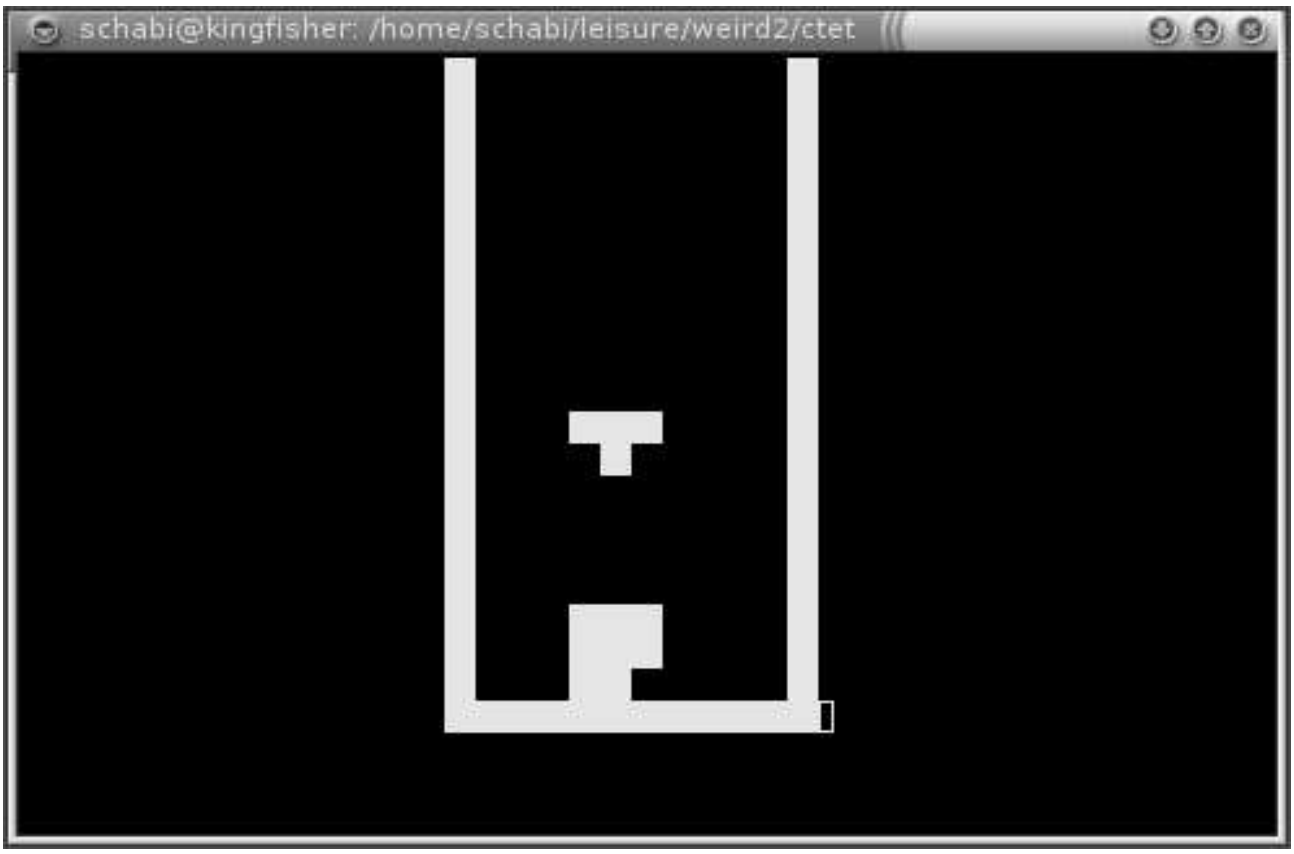
```
#include<stdio.h>
int
f[61],c,i;main(){for(;c=getchar()+1;*f=-38)if(c>33)for(c=c<61?c-
49:8+c&31,i=1;i<60;f[i++]=c%11,c/=11)c+=f[i]*23;else
for(c=0;i>=0;f[i--]=0)if(c|=f[i]|i<2)putchar(f[i]>9?32:48+f[i]);}
```

Compact something

Across some unreproducable 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;srand(getpid());system("stty cbreak -echo stop u");sigvec(14,v,
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){
for(;j%12;q[j--]=0);u();for(;--j;q[j+12]=q[j]);u();}n=f+rand()%7*4;G(x=17)||(c
=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
(++x)||--x;if(c==a[3])for(;G(x+12);++w)x+=12;if(c==a[4]||c==a[5]){s=sigblock(
8192);printf("\033[H\033[J\033[0m%d\n",w);if(c==a[5])break;for(j=264;j--;Q[j]=
0);while(getchar()-a[4]);puts("\033[H\033[J\033[7m");sigsetmask(s);}}d=popen(
"stty -cbreak echo stop \023;sort -mnr -o HI - HI;cat HI","w");fprintf(d,
"%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 that 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

virtual machines. Argh! and Aargh! were both adjusted to fit the special needs of customary unix text mode terminals. Their 2-dimensional micro machine is sized 80 columns. Argh! limits this to 40 lines, while Aargh! does not impose any limit to the number of lines. Aargh! thus allows self-modifying Programs to grow limitless, and is believed to be turing complete. Both provide an unlimited stack.

They are ideal for free software developers, because they are licensed under then GNU GPL, and provide a complete tool chain. There is a C-based interpreter, a perl based Debugger, an Emacs mode including interpreter and debugger, and a VIM editing mode. And it manages to express the frustration of developers, als all error messages begin with “Argh!”.

Hello world, as delivered in the file hello.agh as part of the distribution, reads as follows:

```
1  lpppppppppppsrfj
2   Hello World*  j
3               qPh
4
5
6  hello.agh - the classic killer application
```

Note that line 6 is part of the program. Argh! and Aargh! let you freely use unused places of your source for comments, without any syntactic restrictions.

Of course, those languages are usable for productive uses, too. The following example, which is part of the contrib directory of the Argh! distribution, implements rot13 as a shell filter. It reads from stdin and puts the result to stdout. And it takes advantage of the special # instruction that allows unixoid environments to automatically find the Argh! interpreter via the famous #! shebang trick (which is also used e. G. by shells, python and perl). This allows you to use Argh! programs as shell commands, directly from the commandline.

```
1  #!/usr/bin/env argh
2  llllllj        lllllllj     lllllllj
3  jhhheh         D NA  j      D na  j
4  j    jh    ldrXDARllj  ldrXDARllj
5  jlsdRxXq   D A  lDarj  D a  lDarj
6  jk   llDldrXDdrxk ANldrXDdrxk anlfj
7  jkGh      N   ZlDlllk n   zlDlllk j
8  lllkhhhhhhhhhhhhhhhhhhhhhhhhhhhhPh
9
10 # rot13 written in Argh!
11 # Copyright (C) 2004 by Thomas Arendsen Hein <thomas@intevation.de>
12 # This program is free software under the GNU GPL (>=v2)
```

Of course, the famous 99 Bottles of Beer is also possible in Argh!

```
1  lsdllDsalrxdarxsL       HshDpppppppppppppppppppppppppppppppppppsrddffj
2   *    e124j 21lDsJ       DD No more bottles of beer on the wall.*     j
3  jhhDfdFdRAhDh   q         s      HDh        jfpppppppppppppppppppppppPPh
4  j                    0  lllllllllllJ        j !reeb erom yub ot emiT
5  lsllSsrfsrfllllllsrfSj kphhhhph            lPq
6  j0   KA KA        21  H   0 D  k
7  lDllllRxlAllllfsjllRxlDlFRxklFk
8        k lfsrfj  0jk  lfsrfjlak             The classic
9        k   21 jjShhk    21 j 0              99 bottles of beer
10       khhShhAhllllkhhhShhAh                in Argh!
11                 q h         l   lsK
12                 lDSdlllllflSJh hJ*        wilde@sha-bang.de
13   jfrsppppppppppppppppppppppppppppph
14   j  *,llaw eht no reeb fo selttob
15   lPlllllllllllllllllllDSlllllllssK
16                         JSfdDh  fJo
17         jdrsppppppppppppppppppppppph
18    jfhhhh  *,...reeeeeb fo selttob
```

```
    j Take one down, pass it around,
    lPPPPPPPPPPPPPPPPPPPPPPPPPPPPPfj
                                  s/  j
                            lKhhhhhSSDPh
                            lDfSllllK /
jffdrsppppppppppppppppppppppppppppph
j    *.llaw eht no reeb fo selttob
lPPlDSlllllllllllllllllllllllllssK
                                  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 otherwhise 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 prices 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 7th 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 stuations 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 abbrevation of "**M**ico **i**s **CO**RBA". But this may be a bad excuse, and the real meaning may be rather "**M**ico **i**s **c**uri**o**us – maybe". The reason for this assumption may be the following excerpt of Mico 2.3.11 source, file orb/except.cc, lines 348-360:

```
1       switch (_completed) {
2       case COMPLETED_YES:
3            os << "completed";
4            break;
5       case COMPLETED_NO:
6            os << "not-completed";
7            break;
8       case COMPLETED_MAYBE:
9            os << "maybe-completed";
10           break;
11      default:
12           assert (0);
13      }
```

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:

```
1  private String spaces(
```

```
    int number
) {
      char[] indent = {' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','
',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','
',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','
',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','
',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','
',' ',' '};
      return(new String(indent, 0, number));
}
```

This solution has a fixed limit of indentation width, and exceeding that limit results in an Exception. Thankfully, this is written in Java. Using the same Algorithm in C would have lead to indenting the output with innocent data that follows the string in the compilers memory layout.

But apart from this, this code is rather inefficient. According to the Java semantics, every call to this method creates a new instance of the char array. Then, a new String instance is created that gets the char array and the number of characters as parameter. This makes the string instance allocate its own new char array, and afterwards, it copies the needed number of spaces from the first array.

Too much cookies cause caries

Application servers such as jboss or zope often use cookies for session management. This solution has the lowest impact on HTML generation, as it does not need URL mangling and works easily with static pages. To inform the user about this requirement, most pages display an appropriate error message when cookies are not set. This is a good intention, and far better than silently failing to work for mysterious reasons.

However, sometimes this test can break a fine working application. The aforementioned OpenMDX 1.4.1 has this problem. They don't check whether the session management works, but use their own test cookie which is baken in the file openmdx-inspector.war/Login.jsp:

```
Cookie cookie = new Cookie(cookieName, cookieValue);
cookie.setMaxAge(60);
response.addCookie(cookie);
```

This Cookie is sent with an expirytime of 60 seconds. Now this is not a relative value, but an absolute time. In our case, the clock of our browser test machine was 3 minutes ahead, so the Cookie was always expired.

They also place JavaScript into the test page that checks whether the cookie is there. It turned out that Mozilla based browsers and Opera accept the Cookie nevertheless (at least the JavaScript can read it, I did not check whether it is sent back to the server). However, Internet explorer does not exhibit the expired Cookie to JavaScript. This has cost us some time to debug as we suspected the problem in the security settings of the IE.

Understanding OOP

Most object oriented programming languages are based on classes and inheritance. All data is organized in classes. Classes define common properties. All data of the same kind is the same class. Instances fill those classes with life, an instance is a set of data that represents the state of a concrete object.

Often, one has some classes that share a lot of properties. This is where subclassing and inheritance comes in handy. You define the common properties in the superclass. Then you define the subclasses which inherit all the common properties. The different properties are then changed in the subclasses, this is called overriding.

This modifications are transparent to the superclass, and to the users as long as they use only the superclass' interface. A superclass usually does not know anything about their subclasses, those can be added by third parties.

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.

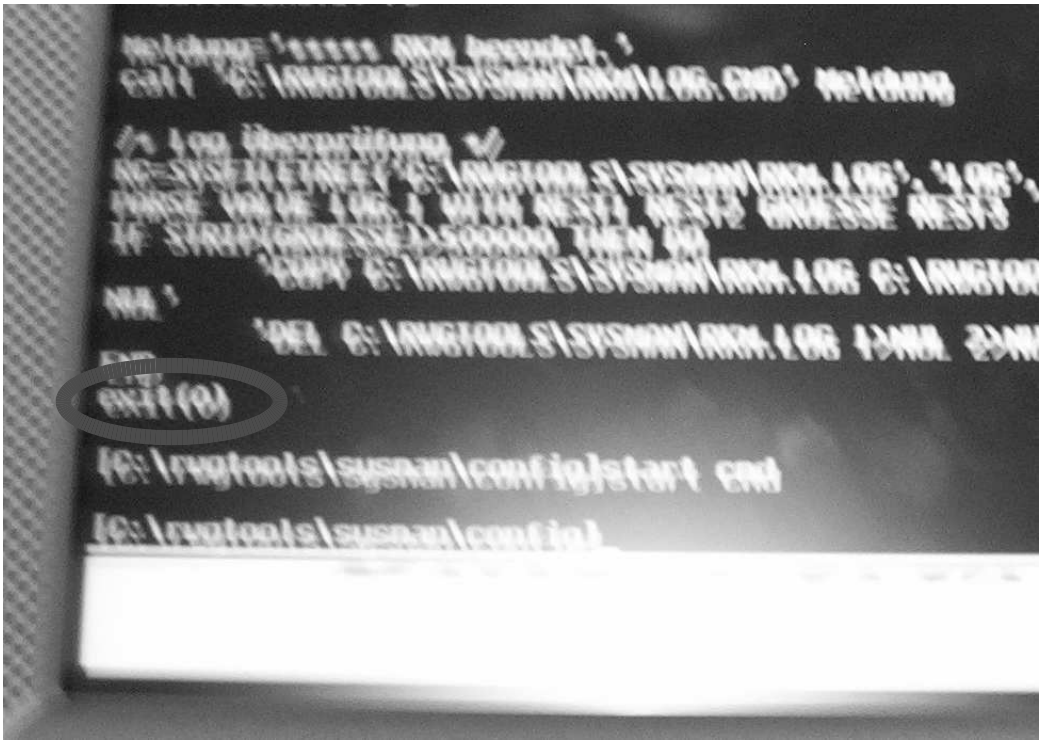
```
if(v == null) {
  s = "";
}
else if(this instanceof ObjectReferenceValue) {
  Action action = ((ObjectReference)v).getSelectObjectAction();
  s = "<a href=\"" + forView.getHRef(action) + "\">" + action.getTitle() +
"</a>";
}
else if(this instanceof BinaryValue) {
  Action action = (Action)v;
  s = "<a href=\"" + forView.getHRef(action) + "\">" + action.getTitle() +
"</a>";
}
else if(this instanceof BooleanValue) {
  s = ((Boolean)v).booleanValue()
    ? application.getTexts().getTrueText()
    : application.getTexts().getFalseText();
}
else {
  s = v.toString().trim();
}
```

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 to 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 window 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.