Software Tools

Mission Accomplished

or

Mission Failure ?

Poul-Henning Kamp

phk@FreeBSD.org

Software Tools

1970 – 2010 -> approx 40 years old

Central paradigm in UNIX philosophy

But is it still relevant ?

”Midlife crisis” or *”Life begins at 40”* ?

The Gospel according to Salus:

AT&T leaves ”MULTICS” project.

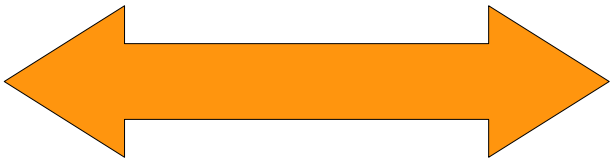
Dennis, Ken & Brian go cold turkey on PDP/8

Saves the world with UNIX, C & AWK

Awarded presidental medal

Everybody Lives
Happy Ever After

MULTICS	UNIX
Big	Compact
Complex	Simple
Unwieldy	Elegant
Delayed	Available
Red Tape	”Call Ken”
BAD OS	GOOD OS



Software Tools

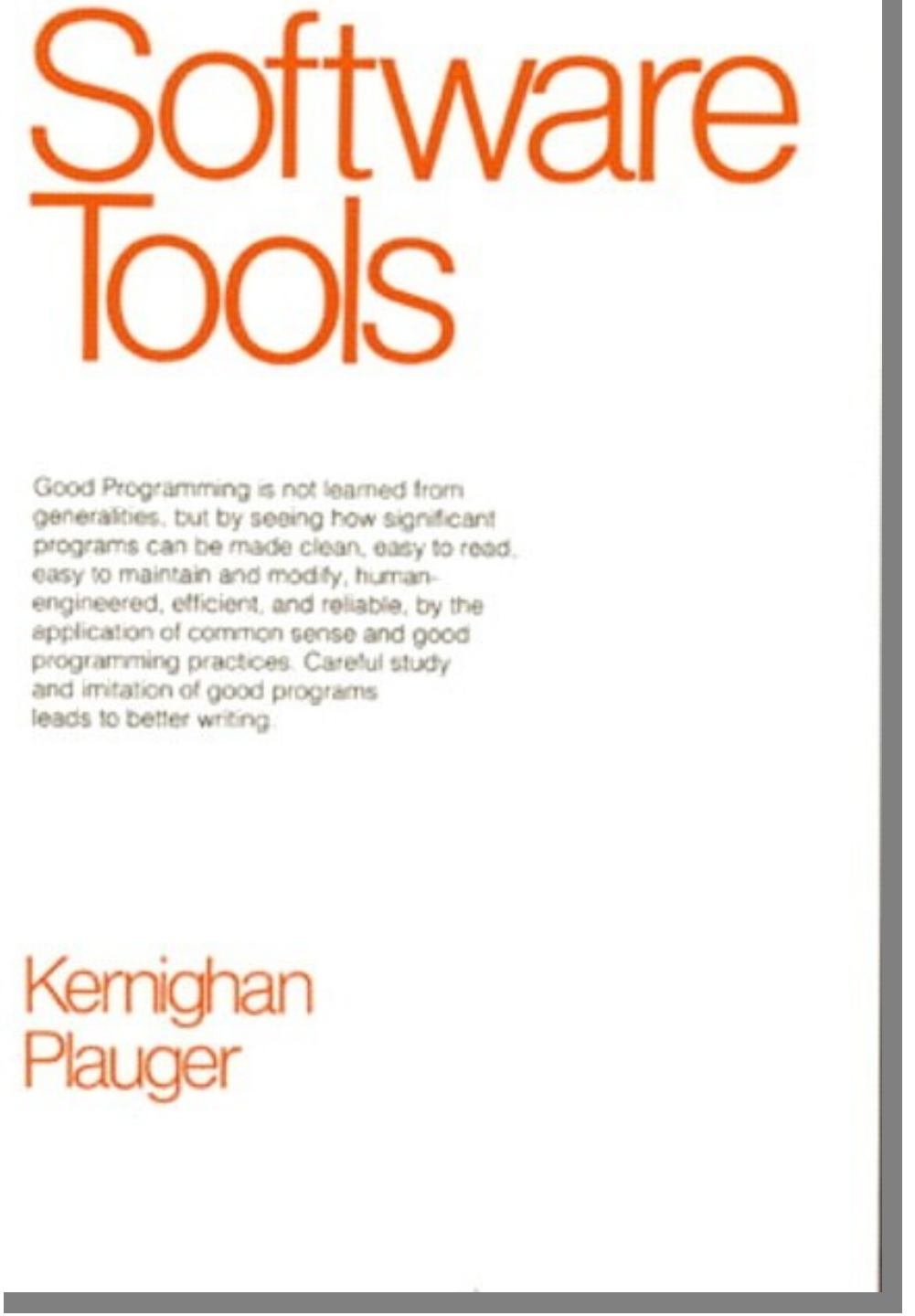
Book: Kernighan & Plauger

RATFOR, Pascal &c.

Argument:

Programs should be tools

(And well written)



Software Tools

Programs should be tools which do one job well

Tools, not policies

Programs should work together

Programs and source code are data

Interaction is programming

```
grep '#include' .c | sed -e 's/.*<//' -e 's/>.*//' | xargs grep foo
```

SwTools => Files are Flat

Files are flat and typeless == One dimensional array of bytes

Name files by/in free hierarchy

Compare IBM S/3:

A Disk has
- Volumes which contains
  - Files, which can be a
    - Library, which contains
      - Members, which have
        - Types, for instance
          - RPG/II Source code

Types of files

File type information conventionally encoded in name

...inconsistently:

*.y *.l *.f *.c *.o *.a *.s

*.Z *.gz *.bz2 *.tar *.tgz

Makefile

.login

rc.*

lib*.so.%d

That Neumann/Turing detail

Metadata X-bit not marks executability

Execution method by magic sequences in file data

```
#!/bin/sh

#!/usr/local/bin/python3.1

\0407\206\000\000

\x7fELF
```

The fine print...

”Programs and source code are data”

is not a transitive: Data is not a program.

<jedi_handwave>
This is not the Turing Machine You Are Looking For
</jedi_handwave>

...unless you say so

(Tools, not policy)

```
python3.1 really_not_advisable.gif
```

But notably not:

```
cc -o hello hello.c
chmod -x hello
a.out hello
```

(although, you could do that: see ld.so)

Somebody didn't pay attention...

Microsoft messed this fine detail up BIG time:

Name-extension controls presentation
= which icon, if not hidden

File content controls activation (=> & executability)
= what happens when you click on it.

```
mv bad.exe playboy.jpg
```

=> Perfect tool for malicious deception & malware

=> User has no means of defense

SwTool facility: pipes

”Programs should work together”

Pipes just a short-hand notation for temporary files
    -> Also how MS-DOS implemented them

Pipes are strictly speaking not mandatory for SwTools

But often considered the litmus test of ”UNIX-like”

How well do pipes work ?

Nice paradigm if everything is in ASCII files

Works well horizontally:

```
awk '{print $2 + $3}' file.dat
```

Works badly vertically:

```
sed '1258,2834s/^/>> /'
```

When do pipes work ?

Pipes only work if programs agree on data structure

The ASCII text-line is a very strong abstraction.

... But ultimately insufficiently expressive

(see also: troff(1), LaTex(1), XML, but also ReST)

Pipes are 1 dimensional

UNIX pipes are one-dimensional:

Inherent in ”stdin, stdout + stderr” and CLI models

```
grep ”^#” dat1 | foobar > dat2
```

Real Pipes are N dimensional

```
grep "^#" dat1 | foobar
  |                 |
 snafu      |      stitch > dat2
```

...but only on IBM mainframes with TSO/Pipes

(See: Jon Hartman, EuroBSDcon2007 Keynote)

SwTool => Processes are cheap

Fundamental consequence of tool-based model

Processes are not cheap by definition

High process creation overhead is a valid design choice

-> resource reservation

-> service guarantees

-> resource prestaging (spin up disks, tapes)

-> authorization/clearance/accounting

```
000001 //W9401RRR  JOB 'NCS.ACS.J9401',FORSTER,MSGLEVEL=1,MSGCLASS=Q,CLASS=N
000002 /*JOBPARM ROOM=#ACS
000003 //*PROCLIB=NCSACS.PROCLIB
001401 //STP1  EXEC PGM=FTP,COND=(0,NE)
001402 //SYSIN    DD DISP=SHR,DSN=NCSACS.J9826.MLF.SOURCLIB(Z9401RR1)
001403 //SYSPRINT DD SYSOUT=*
001404 //*
001500 //*STP2  EXEC P9401RRR
001600 //*
001700 //*STP3  EXEC PGM=FTP,COND=(0,NE)
001800 //*SYSIN    DD DISP=SHR,DSN=NCSACS.J9826.MLF.SOURCLIB(Z9401RR2)
001900 //*SYSPRINT DD SYSOUT=*
```

Processes are cheap(er) now

Even on mainframes
- But only for POSIX compliant env.
- Still slow for COBOL/JCL style stuff.

Requirement for graphical desktops:
- It's not called *”point, click and wait”*

SwTools => Interaction is programming

The UNIX shell is a programming language

Promotes procedural approach to computer usage

Invites automation of common and uncommon tasks

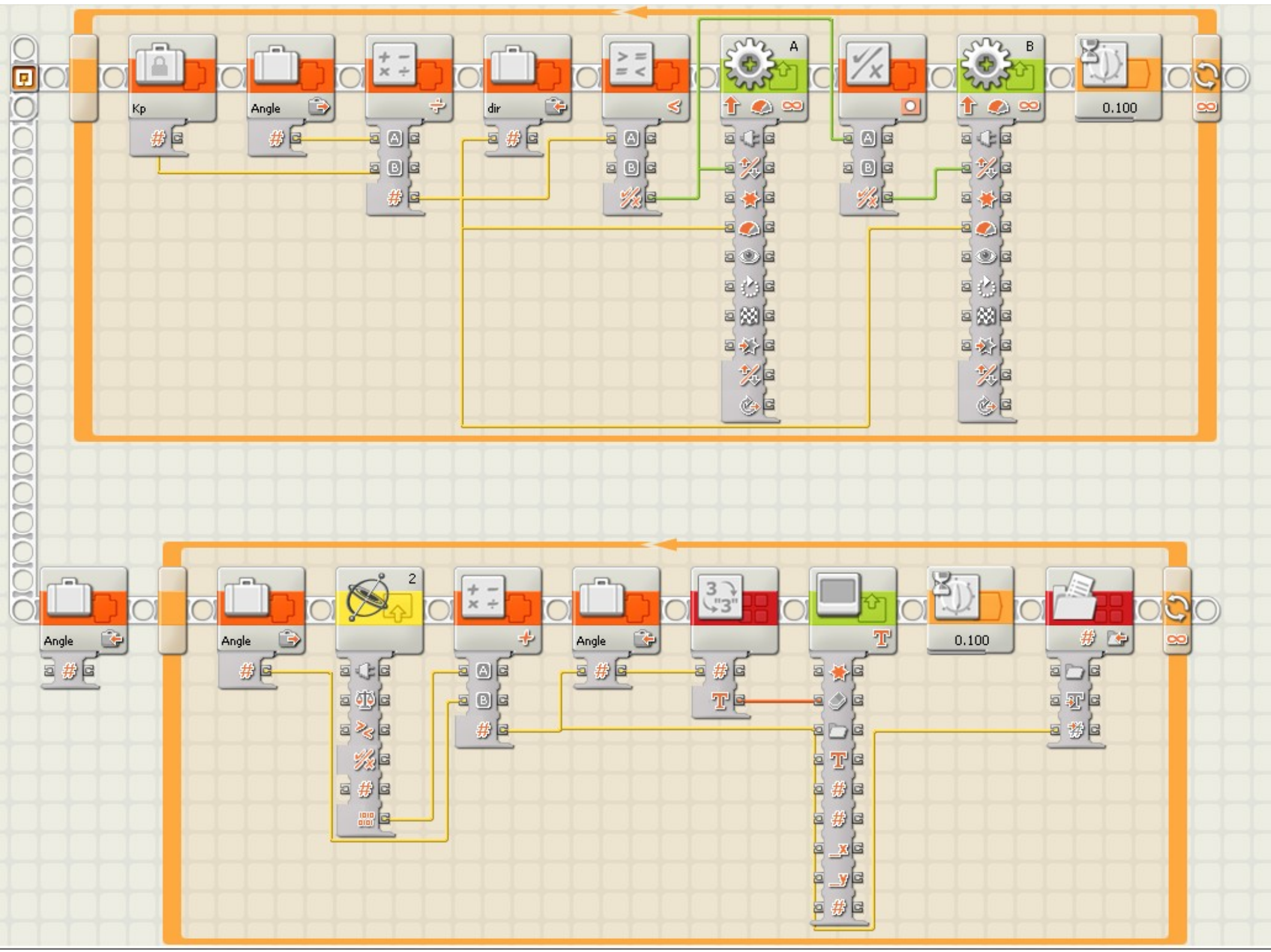
```
#!/bin/sh
find . -type f -print |
        egrep '\.orig|\.rej|\.#' | xargs rm -f
```

SwTools => Interaction is programming

Total defeat

WIMP/desktop metaphor won

You cannot program with a mouse


Kp
Angle
dir
A
B
0.100
Angle
Angle
2
Angle
3
"3"
0.100

SwTools => Interaction is programming

Total defeat

WIMP/desktop metaphor won.

You cannot program with a mouse

Very limited what you can do with scripts

Macro-replay no substitute for scripts

SwTools => Tools not policies

Freedom with responsibility

System comes with compilers and development tools

Root-bit offers unhindered foot-shooting

Protection limited to system integrity

-> fails: `ls -l > /`

-> works: `ls > my_phd_thesis.texi`

AKA: The castor-oil[1] cure for sore throat

(Cough, and you will suffer the consequences)

[1] Also known as ”american oil”

WIMP Design promotes user failure

If your only tool is a mouse,
everything looks like an icon

SOP: *"Try clicking on it and see what happens"*

Executable icons look exactly like any other icon

Automatic execution of code, without users request
-> Datamedia auto-execution trapdoors
-> Rich data format extensions
-> Vendor convenience features

=> We cannot trust the user.

WIMP does not trust users judgement

”Undo” function mandatory

-> Rodents harder to control than keyboards ?

> Despite having clicked yes already three times, are you *really sure* you want to exit our great program?
>
> [Probably not] [Of course not]

Many legitimate & useful operations prevented by program designers lack of imagination

UNIX userland is dead(-ish)

Only used to build programs and start daemons

Everything else:

- Graphical (Mega-)apps
  - Even program development: IDE's

UNIX = Kernel with benefits ?

No developer interest in userland:

```
$ cd /usr/src/sys
$ svn log | grep -c ' 2010-'
3604
$ cd /usr/src/usr.bin
$ svn log | grep -c ' 2010-'
335
$ cd /usr/src/bin
$ svn log | grep -c ' 2010-'
113
```

SwTools in the kernel ?

Device drivers

Vision: ”make it easy to add device drivers”

Reality: Getting there

- NewBus
- BusDMA
- DEVFS
- sysctl

SwTools in the kernel ?

Virtual File Systems

Vision: FS as tools for transforming namespace
    UNIONFS, NULLFS, UMAPFS etc.

Reality: Not happening
    Mounts restricted to root,
    Weak userland tool support
    (see Plan9)

SwTools in the kernel ?

Network stack

Vision: Multiple protocols on same computer
    DECnet, NetWare, IP, X.25, OSI etc.

Reality: Not happening
    One protocol is enough for everybody

Dishonourable POSIX mention:
    All apps need be fixed for IPv6

SwTools in the kernel ?

Disk I/O (GEOM)

Vision: Pluggable storage transformations
    RAID, crypto, remote, slicing etc.

Reality: Probably Survives
    Everybody wants ZFS, which is not modular at all

SwTool wins:

Files are just bytes

Processes are cheap

SwTool losses:

Filename extensions are magical

Graphical desktop sole surviving model

Icons cannot pipe

Mainframes better at pipes than UNIX

WIMP restricts users by designers imagination

Nothing SwTools about (Open-)Office, OOXML, ODF

A different kind of scoreboard

Mobile phones before software tools

Product designer decided what and how you could use the device

Fixed placement of icons

Fixed hierarchy of menus

Software carved in stone.

Mobile phone with software tools

Desktop filled with programs you selected.

”Apps” are tools that do one thing well

Mobile phone with software tools

Product designer delivers tools, not policies

Write your own software

For instance: Use your $$$$ smartphone as a $0.25 flashlight

Mobile phone with software tools

Apps don't work together

(no pipes)

No procedural interaction

A Real SwTools smartphone could...

```
event(incoming_call c) {
   If (hour < 9 || hour > 17 || day == sat || day == sun) {
      if (c.a-number() on customer_list.txt) {
         v = c.accept_call()
         v.play(outside_hours.greeting)
         a = v.record_message(max=2min)
         mail(dst="helpdesk@company",
             subject="Called outside hours, please handle.",
             body=(c.a-number(), c.starttime(), a))
         exit(0)
      }
      if (c.a-number() on company_list.txt) {
         write(file="overtime.txt",
             (c.a-number(), c.starttime(), c.duration()))
         c.ring_tone("Pennies_From_Heaven.mp3")
      }
   }
}
```

Another scoreboard

Kernel/OS as SwTool

FreeBSD ”make release” turned into a tool:
    PicoBSD, NanoBSD, PCBSD, FreeNAS ...

FreeBSD build system turned into a tool:
    Whistle, Juniper, Cisco, IBM, NetApp ...

Sort of a Conclusion

Software Tools is the critical part of UNIX success

To survive:

- Write tools, not programs
- Deliver tools, not policies
- Ditch UNIX tradition, if it does not work

UNIX stuff that does not work

Challenge: I18N

Non-solution: Pretend I18N is sort-of-ascii

Have been tried for 20 years, still doesn't work

UNIX stuff that would work

Challenge: I18N

Solution: Change kernel & userland to understand XML instead of flat ASCII.

```
grep --tag H3 "crazy idea" index.html
```

So much code to hack, so little time...

Thank you for listening

PS: Don't miss in Karlsruhe:

The ZKM.de museum, approx 2km across the Zoo, has a **Konrad Zuse 22** computer