Warum Unix-Ports

Pascal bis Oberon in der Bremer Informatik

Günter Feldmann

Universität Bremen

fld@math.uni-bremen.de

1970th,
Dept. of Electrical Engineering

- 1974/75: first university computer

  CII-Honeywell Bull, IRIS-80

  - first Pascal port from a university in Paris.
  - learned Pascal by reading the compiler sources
  - engineers needed 64-bit REALS,
    compiler got modified accordingly
  - compiling and linking the compiler took 2 days
  - N. Wirth: Systematisches Programmieren

Since 1981,
Dept. of Mathematics and Computer Science

- first personal computers: DEC PDT 11
  - PDP11 instruction set, but some instructions were missing, these had to be emulated in software as the interpreters and compilers used them.
  - UCSD Pascal and some times a Modula (not Modula-2) compiler under RT11.
  - Small local area network via V24 connections

Computer Science

- A series of different computers
  - DEC PDP11/44, BSD Unix
  - DEC VAX 750 with 8 VT100 terminals, BSD Unix
  - 30 Atari 520 ST (M6800)
  - 20 Sun3 Workstations (M6820)
  - all machines were equipped with Pascal and/or Modula-2 compilers
- Some machines (Pascal Microengine, PERQ) were microprogrammed for Pascal (p-code, Q-code)

Computer Science

- workstation pool for students
  - 30 machines (1986), 100 machines today
  - in the beginning of 1990th we acquired Sun4 workstations (SPARC). No Modula-2 compiler!
  - ETHZ released the SPARC Oberon system hosted by SunOS. This system was used in the course “Software Projekt” until 1996. Then “Java” came …
  - programming courses got replaced by courses in “internet programming”

Keeping Oberon alive on our hardware

- OS change: SunOS (BSD) to Solaris (SYSVR4)
  - despite binary compatibility SPARC Oberon failed. Oberon compiler used registers reserved for usage by the system. Hard to find but easy to fix.
- New workstations: IBM RS6000 with Solaris
  - PowerPC processor
  - no software, Solaris got canceled after delivery
  - To bring up Oberon on these machines by cross-compiling looked easy as everything needed was available: POWER Oberon compiler, mixed sources

Keeping Oberon alive on our hardware

- Main problems encountered:
  - Solaris and the POWER Oberon compiler adhered different ABIs (compiler: AIX, MacOS, Solaris: SYSV).
    - The loader (coded in C) could not call Oberon procedures without assembler code
    - Oberon code could call ext. C procedures only with a limited number of parameters
  - The big endian PowerPC processor was used in little endian mode by Solaris.
    - Some of the instructions used by the compiler caused illegal instruction traps in little endian mode

Further developments at ETHZ

- Solaris (SPARC, PPC) Oberon: Oberon V4

- New developments:

  J. Marais: GUI, Gadgets, System 3

  M. Franz: Code generation on the fly, OMI, Slim Binaries

- Solaris PPC: compiler from MacOberon, SYS V ABI problem
- Solaris SPARC: compiler upgrade,

  Mac Oberon frontend + SPARC Oberon backend

- Extended structure “Module” & expandable heap

Additional and reanimated Ports

- Linux PPC: Solaris PPC had been abandoned by SUN
- Problems with Linux PPC:
  - explicit cache invalidation after code loading,
    got assembler code from kernel developers
  - Cross compiling (little endian, big endian) failed,
    compiler bug (WriteBytes instead of WriteLInt)
- Solaris x86 (OpenIndiana): my preferred system
- MacOS X: MacOberon failed on MacOS X and
  people from Amsterdam asked for it
- Linux x86: purchased 30 Linux systems for the public pool,
  the existing LinuxOberon failed, exception state

Preparing UnixOberon for code optimizing

- Generating code on the fly > profile executing code > optimize code in the background > reload optimized module(s).

- interruptable background thread needed for code optimizing

- New module Threads.Mod (based on Solaris threads, posix threads) posix threads missed "suspend" and "resume" (needed for GC)
- Display operations needed a mutex to avoid X server crashes.

- Reimplemented J. Gutknechts SortDemo program with concurrency to show that threads and "System 3 Gadgets" can work together.

Unix Ports of AOS

- T. Kistler left Irvine for working at Transmeta,
  code optimizer was never released to the public and
  the working group of M. Franz discontinued using Oberon

- As the Unix ports of Oberon now contained threads
  F. Friedrich animated me to try an Unix port of AOS

- AOS depended on the language extensions of paco,
  no paco for PPC and SPARC, abandoned these platforms

Unix Ports of AOS

- Started with own '.Tool' texts, module hierarchy by trial and error, later switched to S. Staubers Release.Build
- The hardest parts:
  - reliable implementing the kernel procedures Lock, Await and Unlock based on POSIX threads with round robin scheduling
  - keep the heap stable (adresses >= 2^31, recursive mark)
- UnixAOS supports applications which can be run without the AOS desktop

  aos -x M.P

  Restrictions: apps. may not import windowmanager and display

Linux specific Problems

- Frequent modifications of the system interface
- Missing sigaltstack, could not handle stack overflow.
- POSIX threads allow priorities only in FIFO mode and need to be run suid root, can totally block Linux!
  In Solaris and Darwin thread priorities had no problems
- 64-bit Linux: dynamic linking failed, C-adapters needed

- Biggest mistake:
  - Never installed the original AOS system to see the differences in behavior to UnixAos. An ugly bug in module XDisplay.Mod (vast of heap space) persisted undetected for a long time.
- Compiler extensions I would like:
  - Procedure inlining (crypto software)
  - Adjustable (small) stacksizes: timer, X-input polling, ...

    Problem with large stacks and compiler paco:
    - many parsers > many stacks > unix paging >
    - paging slows down compilation speed >
    - time barrier > compilation fails

Oberon in progamming curses ?

- The compiler cannot be used in a familar environment
- Oberon modules cannot be combined with third party software
- Documentation:

  A complete manual for the (current) compiler is missing.

  One has to read at least the following documents:

  H. Mossenbock, N. Wirth: The Programming Language Oberon2, 1993

  P. Reali: Active Oberon Language Report, 2004

  F. Friedrich, F. Negele: Proposal for Modules Contexts, 2008

  Friedrich, Glavitsch, Negele, Stauber:

  A2 Proramming Quickstart Guide, 2010

  still undocumented:

  Inline assembler, procedure inlining, current language (Fox)