Curriculum Vitae — Nicholas Alcock

Phone: +44 7853 694629
Email Address: nix-cv-enquiry@esperi.org.uk

Home Address:
4 Little London Gardens,
Lynn Road, Ely,
Cambridgeshire,
CB6 1BF

Job Requirements

I have enjoyed working on debuggers, tracers, and toolchain software for decades; my ideal job would leverage this knowledge in some way.

I learn new programming languages fast and follow language standards meticulously. I tend to be the person that is consulted when developers have tricky technical problems they cannot solve. I believe in good documentation, and am a strong advocate of writing documentation before code, and of open systems and free software in general.

I am strongly self-motivating; I would prefer to work remotely, as I have for the last fifteen years.

Employment History

For 15 years, Principal Software Engineer working in the Oracle Linux Toolchain and Tracing team.

Immediately previously, Senior Analyst Programmer on the RIMS financial markets trading system.

Experience Summary

Linux and Unix Focus of attention for my entire career: most experience running Debian and Oracle Linux; also experience in RHEL 2–5, Solaris, Tru64, IRIX, and AIX.

C Extensive experience throughout my career. In my current employment, and to some considerable degree before that, GNU toolchain work; mostly GNU `binutils`, but also to some degree glibc, GCC, the GNU Build System, and other GNU tools as needed. Work on DTrace for Linux. (See below.)

Even before my current employment, worked mostly on server-side and infrastructural programs (without a user interface; not web applications). Substantial experience with library interface design.

Scripting Languages Writing scripts (large and small) in the POSIX shell, `zsh`, and Emacs Lisp. Also Python 2.1+, Lua, Guile, scsh, Common Lisp, and some Perl.

BPF/BTF/CTF Comprehensive knowledge of the CTF Solaris-era cousin of BTF, and considerable knowledge of BTF. (See below.)

Quite a lot of experience with raw BPF code generation (without going through `clang`).

Version Control `git`, and in the past, Mercurial, `darcs`, Subversion, CVS, RCS, SCCS.

Rust A bit rusty at this: my last real experience was with version 1.40, since when I've been keeping up with the release notes. I plan to pick this up again.

C++ Recent experience with the restricted form of C++ used in GCC. Previous experience with more conventional C++ (with Boost, STL, and a little Qt) but this was some time ago, before C++11. I will probably refresh my skills in this area too.

Database Systems Until 2011: Oracle Database versions 7–11; PostgreSQL 6 and above.

Selected Employment Projects: Oracle

My involvement in these projects included requirements specification (where the requirements weren't already known), design, programming, testing and documentation, usually as part of a team.

CTF and `libctf` This C type-system debuginfo library has been my major focus for the last few years, initially as a requirement of DTrace (see below) but later as a library in its own right.

Substantially rewrote the Solaris `libctf` library and extended it dramatically with many new features and improvements, including automatic type deduplication by `GNU ld`, via a recursive-hashing strategy, with reliable detection of ambiguously-defined types; efficient lookup of ELF symbol types; and much more. `libctf` was contributed to the GNU `binutils` in 2019. See https://sourceware.org/git/?p=binutils-gdb.git;a=history;f=libctf;hb=HEAD

Further extensions are underway that implement a new version of the CTF file format which is upwardly compatible with BTF, so that existing BTF tools will work on CTF as well. The library API is also much improved. See https://sourceware.org/git?p=binutils-gdb.git;a=shortlog;h=refs/heads/users/nalcock/road-to-ctfv4.

Supervised a young software engineer who worked on the above area.

I am (and will continue to be) the maintainer of this component of GNU `binutils`.

Previously, wrote a different (kernel-specific) CTF deduplicator for the Oracle UEK kernel (never upstreamed).

CTF Presentations 2019: Linux Plumber's Conference (LPC)

2020: LPC, OSS Europe, and GNU Tools Cauldron

2024: LPC and Cauldron

2025: Cauldron

DTrace for Linux Despite calling itself a tracer, DTrace is a mix of a tracer, a compiler, a debugger and a graphing and statistics tool, and I've dealt with nearly all of this except for the statistics part and the old kernel component it used to have before 2019. DTrace for Linux is now included in Debian and Gentoo.

The most challenging single part of this was probably porting Solaris `libproc` to Linux, including live inspection of the internals of glibc to permit symbol lookup of alternate lmids; but the port involved much other work, including an implementation of userspace statically defined probes.

Also made a good few changes to the kernel build system, and added code to handle things like kernel symbol lookup more reliably, in addition to tying in CTF

generation. (Largely not upstreamed in the end: the plan is to be able to use BTF for everything, with `libctf` doing the deduplication better than the existing machinery in `pahole` can.)

Wrote most of the build system and test harness for DTrace for Linux (with support for all the usual things like test tagging, logging, triggering, etc).

See https://github.com/oracle/dtrace/ (most of my work on this was 2018 and earlier, after which I focused on `libctf`, above).

Selected Employment Projects: RIMS (previous)

File transfer system A flexible, protocol-independent library to transfer files to and from arbitrary destinations, customizable by clients on-site via shell scripts.

Messlisp A multiprocessing, type-inferencing, soft-realtime optimizing Lisp interpreter specialized for document transformation.

Tools Numerous tools written to automate testing, building and release procedures, some at the request of others, some at my own initiative, including an automated testing system modelled on DejaGNU, and a linter based on GCC with substantial enhancements for (largely flow-based) verification of various important invariants.

Infrastructure Implemented a lot of generic infrastructure, mostly providing convenient interfaces to do things in C, with documentation and testsuites so that others might benefit.

Optimization Optimization work, taking a system which had not had significant speed optimizations for fifteen years and speeding it up by approximately two orders of magnitude via extensive use of profile-guided transparent caching (automatically injecting caching operations via a preprocessor), some explicit caching, and the occasional algorithmic improvement.

Education

B.Sc. (Hons.) Computer Science, Brunel University