What is UNIX, Anyway?

Marc Rochkind
19-June-2003
rochkind@basepath.com

Topics

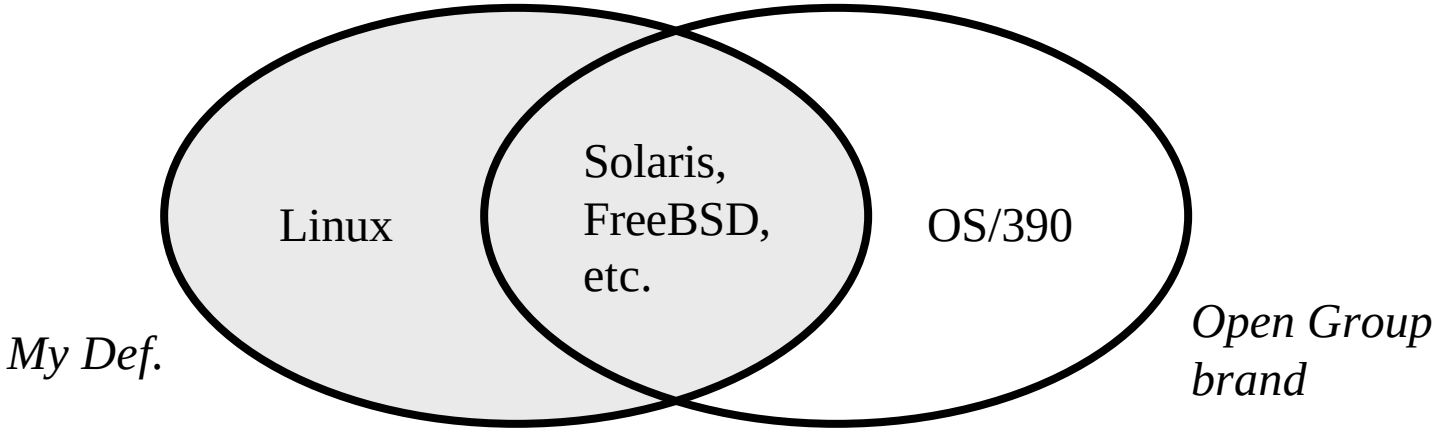
- Definition of UNIX
- Growth of kernel API
- Scope of Advanced UNIX Programming, 2nd Edition
- Problems with API and their origin
- Prospects for fixing or wrapping the API

Definition of UNIX

- Original Bell Labs system or successor? (Solaris, AIX, HP/UX, FreeBSD, ...)
- System that “substantially” conforms to above? (Linux?)
- System the qualifies for UNIX brand? (OS/390?)

My Definition

- A UNIX system is an OS that implements* some version of POSIX.1 as its lowest-level (native) API.



* Except for bugs and minor omissions.

Growth of Kernel API

"... the size constraint has encouraged not only economy, but also a certain elegance of design."

– Thompson and Ritchie, "The UNIX Time-Sharing System" (BSTJ, July/August 1978)

So much for constraints...

+

Þ

®

V7 (1978) – 70 system calls

SUSv3 (2001) – 500 system calls

Who Owns “UNIX?”

- Open Group: 1996 merger of Open Software Foundation (OSF) and X/Open
- Owns UNIX trademark (gift from Novell)
  - SCO (formerly Caldera) owns UNIX intellectual property – in the news!
- www.opengroup.org
- Jointly works with POSIX as Austin Group
  - www.opengroup.org/austin

Open Group Priority: Coverage

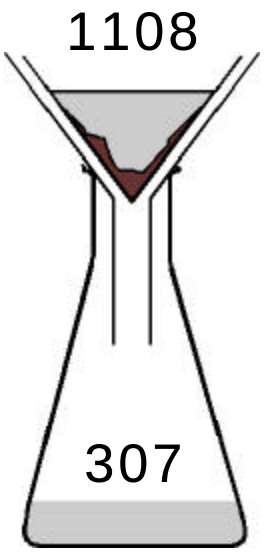
- "The focus of this initiative was to deliver the core application interfaces used by current application programs."
  - "The Single UNIX Specification: The Authorized Guide to Version 3"
- X/Open XPG4 Base + C standard library + APIs used by Top 10 apps + APIs used by 3 of Next 40 + APIs used by 7 of 3,500 modules = 926 APIs (v3 added 182)

SUS

Fantasy vs. Reality

Scope of AUP2

- Scanned SUSv3 with Perl program to create database sorted by header (1108)
- Reduced to 307 by removing:
  - Most ANSI C and other user-level functions (586)
  - 90% of threads (89)
  - Some realtime (9)
  - Spawn (21) and trace (50)
  - Accounting (20)
  - Obscure, obsolete, other (31)



AUP2 Laboratory

- SUS v3 as primary reference
- Four test systems:
  Solaris, Linux, FreeBSD,
  Mac OS X (FreeBSD)
- Writing and code editing on Windows
  (Word, Textpad)
- Source on FreeBSD
  (NFS & Samba)
- 4 Telnet/SSH windows
  (PuTTY)

API Problems

- Inconsistent error reporting
- Too many standards and options make porting difficult
- Blocking and signal handling; threads
- Inconsistent organization, naming, and arguments
- Missing features
- Defective calls

Checking an Error Return

```
void *p;

if ((p = shmat(shmid, NULL, 0)) == NULL)
  perror(“Can’t attach shared memory”);
```

Inconsistent error reporting

- Did error occur?
  - Reserved return value: `-1`, `NULL`, `>0`, special (e.g., `SEM_FAILED`)
  - Change in `errno`
  - Unreserved value (`nice`)
- What was it?
  - return value, `errno`, `h_errno`, `getdate_err`, `gai_strerror`
- Difficult to test error-checking – must get it right the first time

Too many standards

- POSIX1988, POSIX1990, POSIX1993, POSIX1996, XPG3 (X/Open Portability Guide), SUS1 (Single UNIX Specification, Version 1), SUS2, SUS3
- Also: C (included in SUS), C++, ...
- Some systems report erroneously (my Linux reports SUS2, but it isn't)

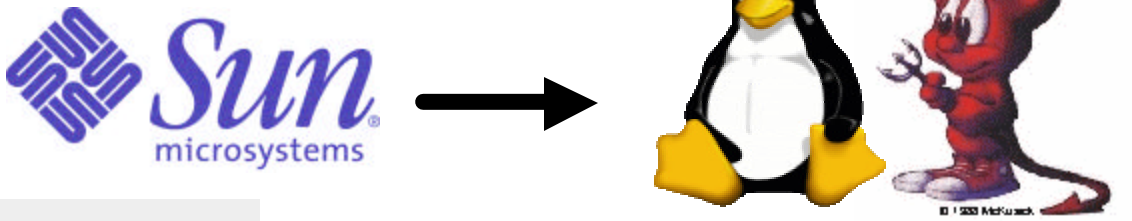
... and options

_POSIX_ADVISORY_INFO, _POSIX_ASYNCHRONOUS_IO, _POSIX_BARRIERS, _POSIX_CHOWN_RESTRICTED, _POSIX_CLOCK_SELECTION, _POSIX_CPUTIME, _POSIX_FSYNC, _POSIX_IPV6, _POSIX_JOB_CONTROL, _POSIX_MAPPED_FILES, _POSIX_MEMLOCK, _POSIX_MEMLOCK_RANGE, _POSIX_MEMORY_PROTECTION, _POSIX_MESSAGE_PASSING, _POSIX_MONOTONIC_CLOCK, _POSIX_NO_TRUNC, _POSIX_PRIORITIZED_IO, _POSIX_PRIORITY_SCHEDULING, _POSIX_RAW_SOCKETS, _POSIX_READER_WRITER_LOCKS, _POSIX_REALTIME_SIGNALS, _POSIX_REGEXP, _POSIX_SAVED_IDS, _POSIX_SEMAPHORES, _POSIX_SHARED_MEMORY_OBJECTS, _POSIX_SHELL, _POSIX_SPAWN, _POSIX_SPIN_LOCKS, _POSIX_SPORADIC_SERVER, _POSIX_SYNCHRONIZED_IO, _POSIX_THREAD_ATTR_STACKADDR, _POSIX_THREAD_ATTR_STACKSIZE, _POSIX_THREAD_CPUTIME, _POSIX_THREAD_PRIO_INHERIT, _POSIX_THREAD_PRIO_PROTECT, _POSIX_THREAD_PRIORITY_SCHEDULING, _POSIX_THREAD_PROCESS_SHARED, _POSIX_THREAD_SAFE_FUNCTIONS, _POSIX_THREAD_SPORADIC_SERVER, _POSIX_THREADS, _POSIX_TIMEOUTS, _POSIX_TIMERS, _POSIX_TRACE, _POSIX_TRACE_EVENT_FILTER, _POSIX_TRACE_INHERIT, _POSIX_TRACE_LOG, _POSIX_TYPED_MEMORY_OBJECTS, _POSIX_VDISABLE, _POSIX2_C_BIND, _POSIX2_C_DEV, _POSIX2_CHAR_TERM, _POSIX2_FORT_DEV, _POSIX2_FORT_RUN, _POSIX2_LOCALEDEF, _POSIX2_PBS, _POSIX2_PBS_ACCOUNTING, _POSIX2_PBS_CHECKPOINT, _POSIX2_PBS_LOCATE, _POSIX2_PBS_MESSAGE, _POSIX2_PBS_TRACK, _POSIX2_SW_DEV, _POSIX2_UPE

Method of testing changed in SUSes:

```
#ifdef _POSIX_ASYNCHRONOUS_IO // wrong
```

Conspiracy?



Last 3 Years

Conspiracy Theory #2

Blocking and Signal Handling

- Blocking
  - Not all resources are represented by file descriptors (message queues, semaphores, mutexes, processes, ...)
- Signal handling
  - Few functions are async signal safe
  - Can't even close FILEs from signal handler
- Threads solve these, but introduce complexity

Inconsistent organization, naming, and arguments

- `mktime`, `localtime`, `gmtime`, `ctime`, `time`, `asctime`, `strftime`, `strptime`
- Some functions grouped (`aio_*`); most not (`socket`, `bind`, `connect`, `listen`, `accept`)
- Weird clashes: `sigaction`, `signgam`
- Horrors: `posix_trace_attr_getmaxsystemeventsize`

Missing features

- A “problem” increasingly well “solved” by vendors, organizations, and Open Group
- 1108 functions in SUS3

Defective calls

- `mktemp`, `signal`, ...
- Obvious why they are kept in
- Chiefly a documentation issue

How did it get this way?

- “Open source” since day one
- IEEE and Open Group collect, but rarely change – aesthetics and usability not important
- Not breaking existing apps is the rule
- Screw-ups get turned to stone (`getdate`, `getaddrinfo`, `shmat`, `gethostbyname`)

Fixing the API?

- Improvement in the standard?
  - No incentive for anything other than missing features (my opinion)
- Living with it?
  - Too hazardous
- Wrapping the API in C or C++?

Experiment 1: C Wrapper

```
NuErrno NuFd_chmod(NuFd fd, mode_t mode);

NuErrno NuFd_chown(NuFd fd, uid_t uid, gid_t gid);

NuErrno NuFd_close(NuFd fd);

NuErrno NuFd_dup(NuFd fd, NuFd *newfd);

NuErrno NuFd_dup2(NuFd fd, NuFd fd2);

...

NuErrno NuFd_write(NuFd fd, const void *buf, size_t nbytes, ssize_t
*nwitten);

NuErrno NuFd_writev(NuFd fd, const struct iovec *iov, int iovcnt, ssize_t
*nwitten);

NuErrno NuPath_access(const NuPath path, int what);

NuErrno NuPath_chmod(const NuPath path, mode_t mode);

NuErrno NuPath_chown(const NuPath path, uid_t uid, gid_t gid);

...
```

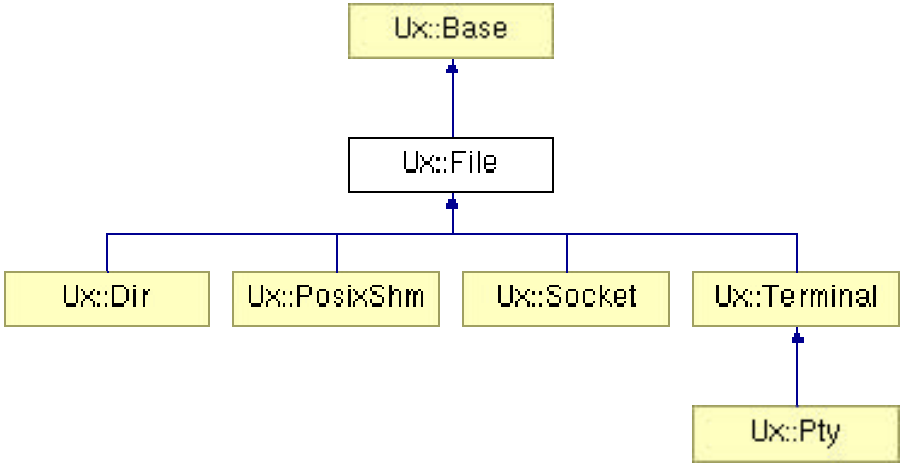
C Wrapper: Results

- Minimal benefits compared to cost of learning and supporting another interface
- Abandoned

Experiment 2: C++ Wrapper

1. 100% uniform error handling for all functions (exceptions).
2. 100% functionality for included functions.
3. Organization into UNIX objects; very thin.
4. Elimination of redundant, obsolete, or defective functions (`readdir`, `signal`, `mktemp`) where there is an alternative.
5. As close to native-C-interface speed as possible.

C++ Wrapper



Looks promising...

Judge for yourself: www.basepath.com/aup/ux

Assigning Blame*

- Thompson & Ritchie: developing useful, simple, popular system and releasing it as open source (virus)
- AT&T: sponsoring T&R; distributing UNIX; fighting OS wars
- Sun: making UNIX commercial; conspiring with AT&T
- OSF: OS wars
- X/Open, Open Group: emphasizing “coverage” over reliability and usability
- IEEE: limiting distribution of standards
- Linus Torvalds: just as “bad” as T&R
- Me,** FRUUG: blameless; listed to increase attendance at talk

* Chill out – this is a joke!

** Not completely, but, after all, this is *my* talk!

Conclusion

- Evolution will continue; mess will get worse
- Not a bad thing. True of all such things.
- SUS API should not be used for AP.
- Use a wrapper.
- Would a standardized wrapper help? Hurt?


LIVE FREE OR DIE
http://www.UNIX-systems.org
UNIX®
UNIX is a Registered Trademark of The Open Group