os-test

Measuring POSIX compliance on every single OS

https://sortix.org/os-test/

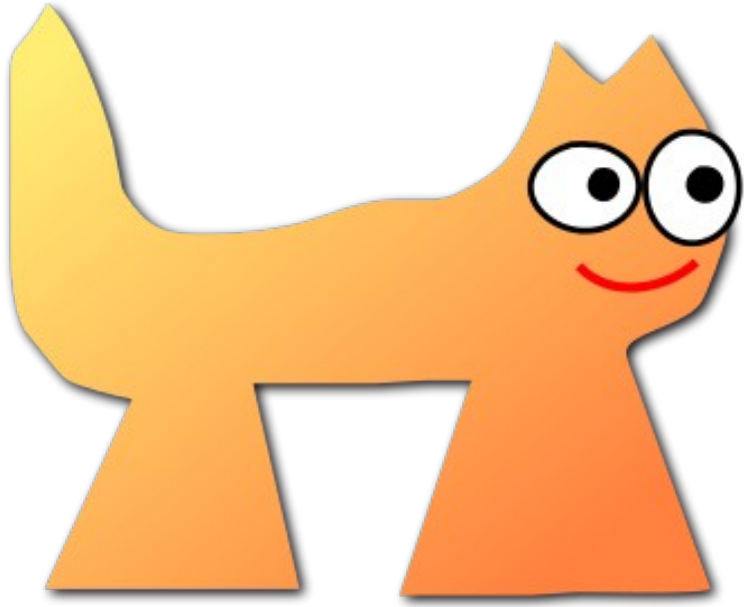
Jonas ‘Sortie’ Termansen, FOSDEM 2026

Hi, I’m Jonas Termansen

- I worked on Dart for Google:
  - Testing
  - Release engineering
  - Supply chain security
- Now I’m making the Sortix operating system full time and developing os-test.

Sortix

- I implemented 90% of POSIX libc in Sortix.
- It’s self-hosting & installable.
- Runs its own sortix.org infrastructure and everything is built natively.

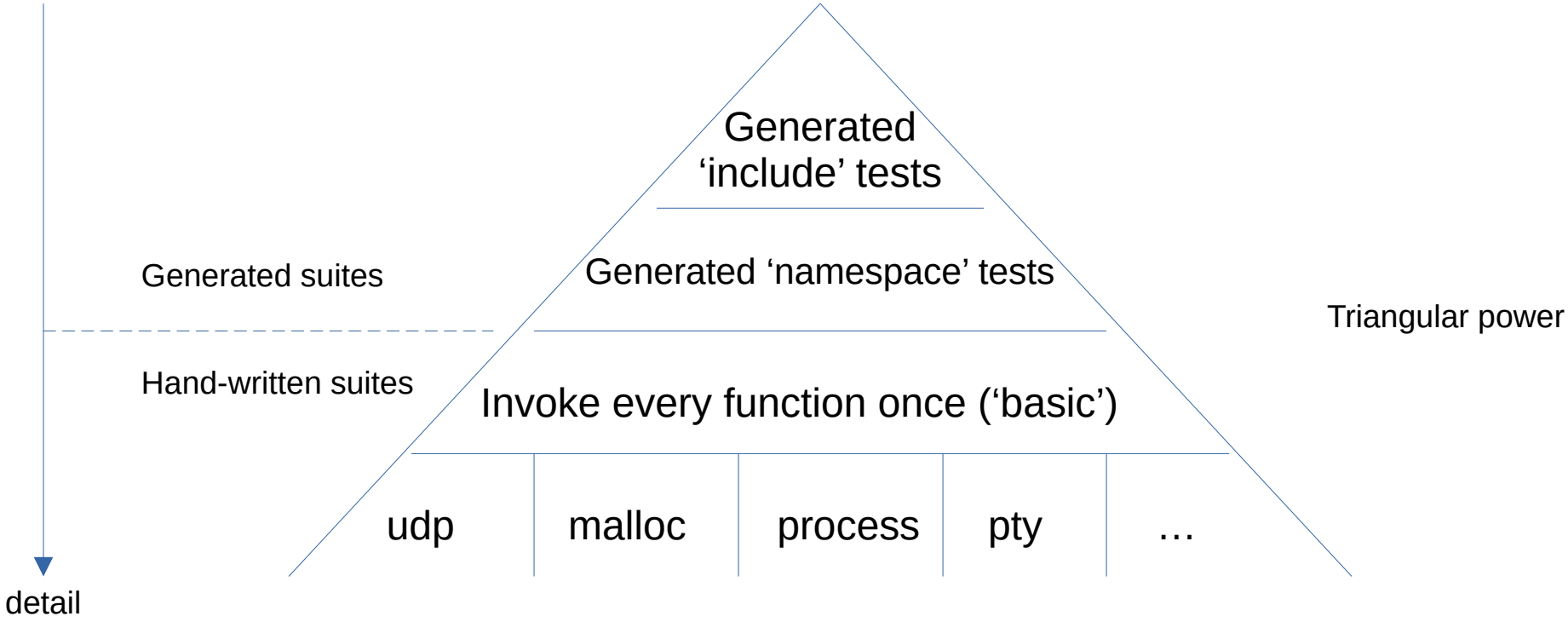


Did you know there is a new POSIX 2024 standard?

POSIX.1-2024

- Exciting new features… but
- The changelog is very incomplete.
- See https://sortix.org/blog/posix-2024/ for all changes.
- The official test suite is proprietary and out of date.
- So I decided to make a public test suite!
- NGI Zero Commons were kind enough to fund it!

100% coverage, in increasing detail



POSIX english is… almost code!

The <time.h> header shall declare the tm structure, which shall include at least the following members:

```
int         tm_sec    Seconds [0,60].
int         tm_min    Minutes [0,59].
int         tm_hour   Hour [0,23].
int         tm_mday   Day of month [1,31].
int         tm_mon    Month of year [0,11].
int         tm_year   Years since 1900.
int         tm_wday   Day of week [0,6] (Sunday =0).
int         tm_yday   Day of year [0,365].
int         tm_isdst  Daylight Saving flag.
long        tm_gmtoff Seconds east of UTC.
const char *tm_zone   Timezone abbreviation.
```

```
The <time.h> header shall declare the timespec structure, which shall include at least the following members:
time_t  tv_sec    Whole seconds.
long    tv_nsec   Nanoseconds [0, 999999999].

[CX] The <time.h> header shall also declare the itimerspec structure, which shall include at least the following
members:
struct timespec  it_interval  Timer period.
struct timespec  it_value     Timer expiration.
```

So I parsed POSIX's html with regexps!

- Custom POSIX parser in C with state machinery.
- Lots and lots and lots of regular expressions.
- And special cases.
- And it worked!

POSIX → Machine readable .api files

```
#include <time.h>
typedef clock_t;
[CX] typedef locale_t;
[CPT] typedef pid_t;
struct timespec;
parent struct timespec struct_member tv_sec: time_t tv_sec;
parent struct timespec struct_member tv_nsec: long tv_nsec;
[CX] struct itimerspec;
[CX] parent struct itimerspec struct_member it_interval: struct timespec it_interval;
[CX] parent struct itimerspec struct_member it_value: struct timespec it_value;
define NULL;
[CX] symbolic_constant CLOCK_MONOTONIC;
[CX] maybe_define function clock_gettime: int clock_gettime(clockid_t, struct timespec *);
[CX] namespace _t$;
```

- One line per declaration with name and allowed types.
- Declarations are tagged with POSIX options groups.

Generated ‘include’ Test Suite

```
#include <time.h>
#ifdef clock_gettime
#undef clock_gettime
#endif
int (*foo)(clockid_t, struct timespec *) = clock_gettime;
int main(void) { return 0; }
```

- Generated 3758 tests.
- A declaration is only allowed if it has the correct type.
- I found a lot of missing features.

```
#include <time.h>
void foo(struct timespec* bar)
{
    time_t *qux = &bar->tv_sec;
    (void) qux;
}
int main(void) { return 0; }
```

‘namespace’ Test Suite

- I preprocessed every system header.
- I used the .api files to check if the headers have declarations that are not allowed by POSIX
- I found a lot of namespace pollution.

```
pollution: CMSG_ALIGN
#define CMSG_ALIGN(len) (((len) + sizeof (size_t) - 1) & (size_t) ~(sizeof (size_t) - 1))
pollution: PF_INET
#define PF_INET 2
pollution: PF_RXRPC
#define PF_RXRPC 33
```

‘basic’ Test Suite

- A function might not work even if it can be linked with.
  - ENOSYS, EINVAL, or buggy
- So, I’m invoking all 1188 functions in POSIX on hello world inputs.
- I invoked 522 so far. Turns out a lot of functions are stubbed and don’t actually work.

```
/* Test whether a basic strlen invocation
works. */

#include <string.h>

#include "../basic.h"

int main(void)
{
     if ( strlen("foo") != 3 )
          errx(1, "strlen did not return 3");
     return 0;
}
```

Detailed Test Suites

- I’m also writing a lot of detailed test suites.
- So far we have: io, limits, malloc, process, pty, signal, stdio, termios, and udp.
- Turns out there are a lot of bugs and platform differences.

I built a lab with every POSIX OS

- I just need ssh + tar + make + sh + cc

  AIX, DragonFly, FreeBSD, Haiku, Hurd, Linux (glibc), Linux (musl), macOS, Managarm, Minix, NetBSD, OmniOS, OpenBSD, Solaris, and Sortix

Thank you to cfarm for access to rare proprietary Unix systems.

If you have access to more systems, do get in touch. I want all the systems I can get my hands on, at least if they're still maintained and relevant. sortie@maxsi.org

So, I ran the tests **everywhere**.

And now I have a **data problem**.

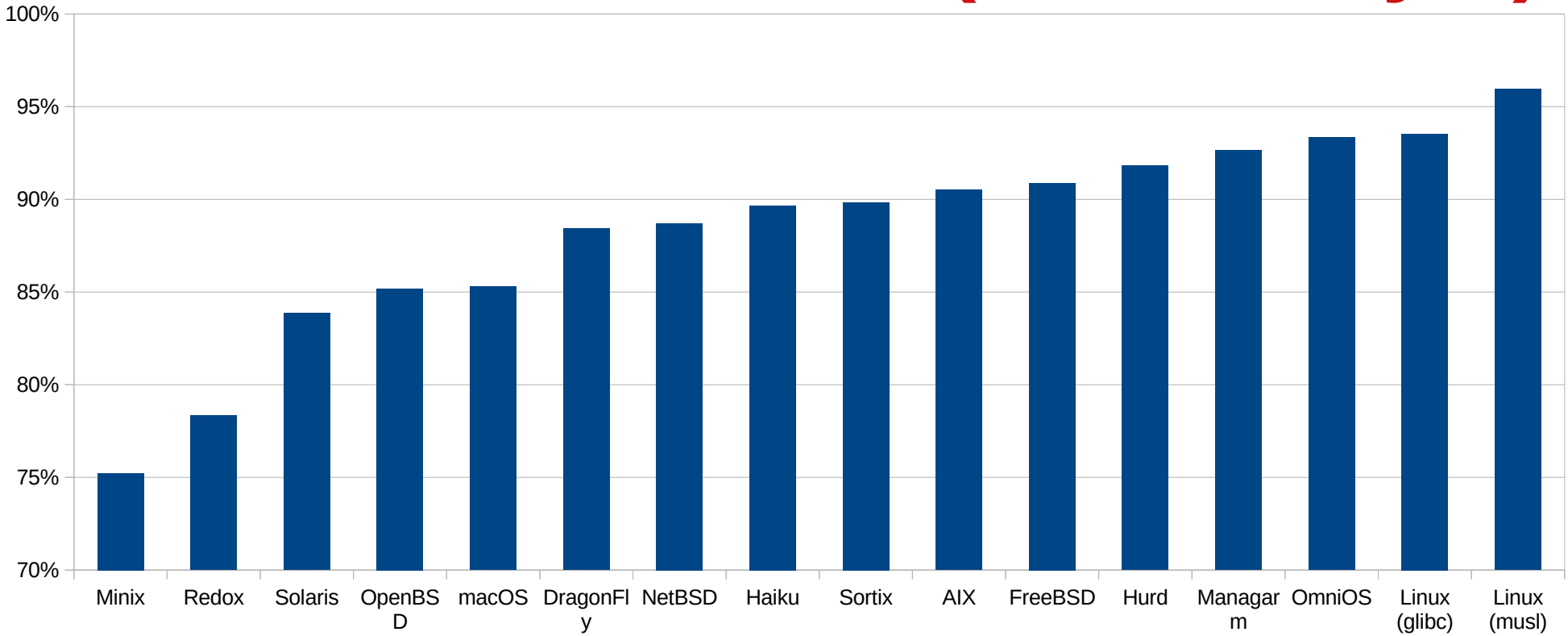
- 75,000 data points and more bugs than I can possibly report upstream.

https://sortix.org/os-test/

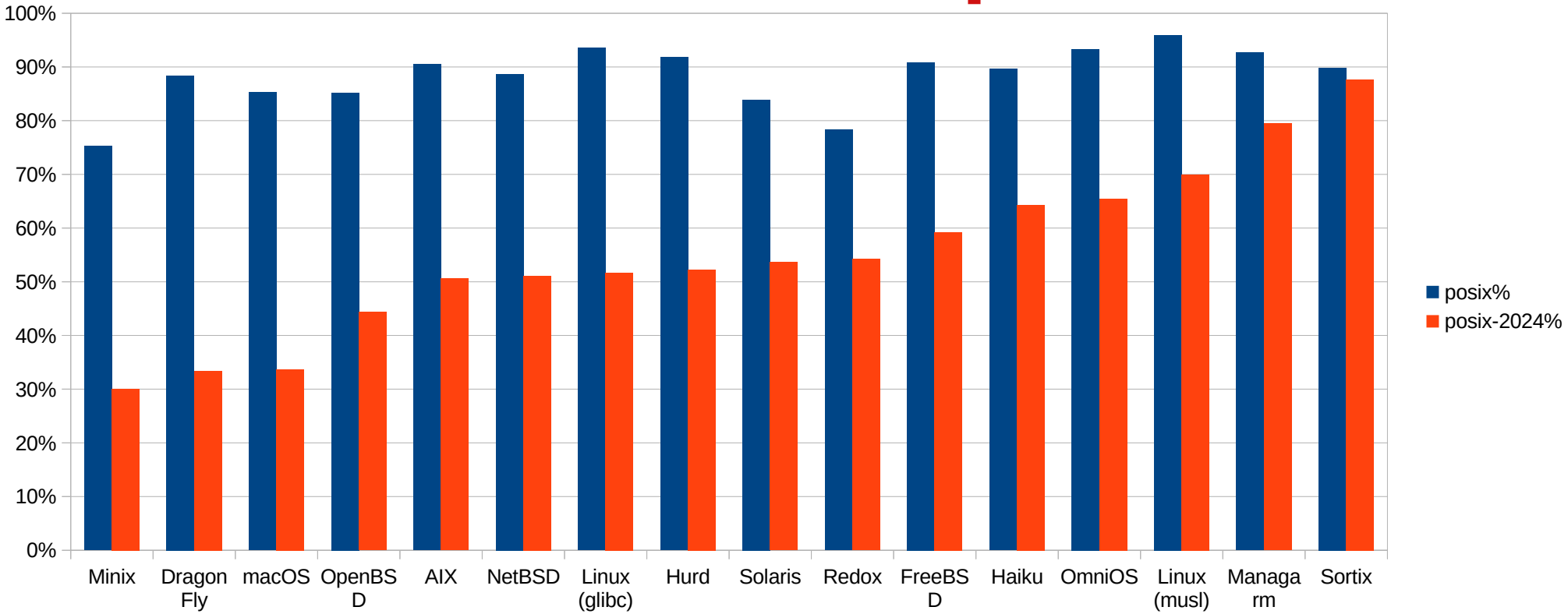
- Overall report
- Sliced along many dimensions
- Click non-green boxes to explore your detailed results

	aix AIX 7.3 powerpc	dragonfly DragonFly 6.4-RELEASE x86_64	freebsd FreeBSD 15.0-RELEASE amd64	haiku Haiku hrev59326 Jan 20 2026 x86_64	hurd GNU 0.9 i686-AT386	linux Linux 6.17.13+deb14-amd64 x86_64	macos macOS 26.2 Darwin 25.2.0 arm64	managarm Managarm 0.0.1-rolling x86_64	minix Minix 3.4.0 i386	musl Linux 6.12.54-0-lts x86_64	netbsd NetBSD 10.1 amd64	omnios SunOS 5.11 r151056 i386	openbsd OpenBSD 7.8 amd64	redox Redox 0.5.12 x86_64	solaris SunOS 5.11 11.4.89.207.2 sparc	sortix Sortix 1.1.0-dev Dec 23 2025 x86_64
§ overall	aix: 90% (3471/3829)	dragonfly: 88% (3397/3834) ~ 91% = +2% (93) as extension	freebsd: 91% (3502/3846) ~ 92% = +1% (72) as extension	haiku: 89% (3441/3833) ~ 90% = +0% (12) as extension	hurd: 92% (3533/3839) ~ 93% = +1% (74) as extension	linux: 93% (3589/3829) ~ 95% = +2% (86) as extension	macos: 85% (3260/3829) ~ 86% = +1% (51) as extension	managarm: 92% (3430/3702)	minix: 74% (2834/3824) ~ 77% = +3% (117) as extension	musl: 95% (3676/3830) ~ 96% = +0% (21) as extension	netbsd: 88% (3416/3846) ~ 90% = +1% (67) as extension	omnios: 93% (3586/3832) ~ 95% = +1% (59) as extension	openbsd: 85% (3286/3846) ~ 85% = +0% (19) as extension	redox: 79% (3034/3801)	solaris: 84% (3234/3838) ~ 93% = +0% (36) as extension +8% (316) as previous_posix	sortix: 89% (3432/3832)
§ basic	aix: 92% (395/426)	dragonfly: 87% (373/426)	freebsd: 98% (419/426)	haiku: 93% (397/426)	hurd: 96% (410/426)	linux: 98% (420/426)	macos: 80% (342/426)	managarm: 89% (316/353)	minix: 63% (272/426)	musl: 98% (418/426)	netbsd: 95% (405/426)	omnios: 96% (409/426)	openbsd: 86% (369/426)	redox: 72% (257/353)	solaris: 96% (411/426)	sortix: 91% (390/426)
§ include	aix: 93% (2790/2995)	dragonfly: 90% (2704/2995) ~ 93% = +3% (93) as extension	freebsd: 92% (2776/2995) ~ 95% = +2% (72) as extension	haiku: 91% (2748/2995) ~ 92% = +0% (12) as extension	hurd: 93% (2788/2995) ~ 95% = +2% (74) as extension	linux: 93% (2790/2995) ~ 96% = +2% (86) as extension	macos: 87% (2611/2979) ~ 89% = +1% (51) as extension	managarm: 93% (2799/2995)	minix: 77% (2310/2995) ~ 81% = +3% (117) as extension	musl: 96% (2882/2995) ~ 96% = +0% (21) as extension	netbsd: 90% (2708/2995) ~ 92% = +2% (67) as extension	omnios: 94% (2822/2995) ~ 96% = +1% (59) as extension	openbsd: 87% (2615/2995) ~ 87% = +0% (19) as extension	redox: 85% (2562/2995)	solaris: 83% (2500/2995) ~ 95% = +1% (36) as extension +10% (316) as previous_posix	sortix: 90% (2699/2995)
§ io	aix: 36% (20/55)	dragonfly: 32% (18/55)	freebsd: 41% (23/55)	haiku: 40% (22/55)	hurd: 36% (20/55)	linux: 96% (53/55)	macos: 94% (52/55)	managarm: 63% (35/55)	minix: 29% (16/55)	musl: 96% (53/55)	netbsd: 36% (20/55)	omnios: 98% (54/55)	openbsd: 40% (22/55)	redox: 36% (20/55)	solaris: 36% (20/55)	sortix: 41% (23/55)
§ limits	aix: 100% (40/40)	dragonfly: 95% (38/40)	freebsd: 100% (40/40)	haiku: 95% (38/40)	hurd: 100% (40/40)	linux: 100% (40/40)	macos: 100% (40/40)	managarm: 100% (40/40)	minix: 85% (34/40)	musl: 100% (40/40)	netbsd: 92% (37/40)	omnios: 87% (35/40)	openbsd: 82% (33/40)	redox: 100% (40/40)	solaris: 87% (35/40)	sortix: 100% (40/40)

How each OS scores (POSIX any%)



POSIX.1-2024 compliance%



Examples of bugs on DragonFlyBSD

Wrong:

```
int setpriority(int, int, int);
int getnameinfo(const struct sockaddr *restrict, socklen_t, char *restrict,
                size_t, char *restrict, size_t, int);
```

Correct:

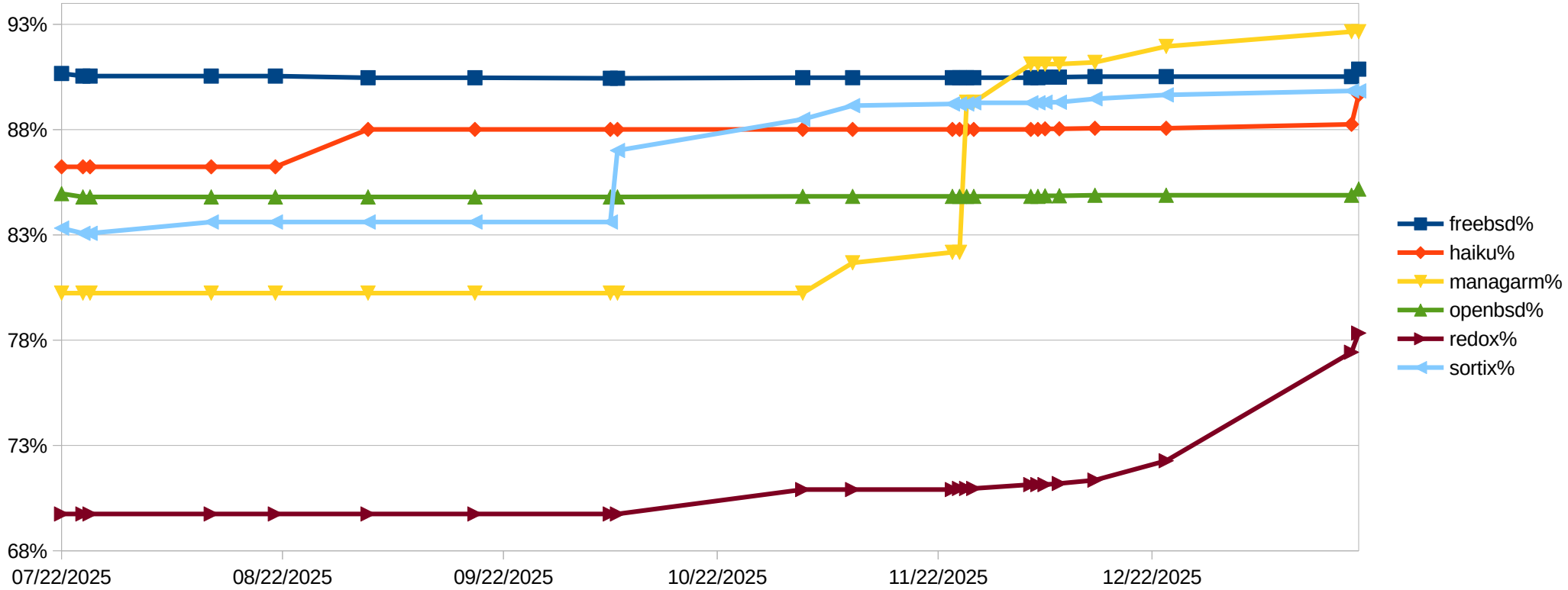
```
int setpriority(int, id_t, int);
int getnameinfo(const struct sockaddr *restrict, socklen_t, char *restrict,
                socklen_t, char *restrict, socklen_t, int);
```

Examples of bugs, sigaltstack

- What happens if you set up an alternate signal stack and execute a new program?
- POSIX says the stack is unset and the SA_ONSTACK bit is unset.
- But: DragonFly, FreeBSD, and OpenBSD preserves the bit on exec => **boom**

```
stack_t ss = { .ss_size = SIGSTKSZ, .ss_sp = malloc(SIGSTKSZ) };
sigaltstack(&ss, NULL);
struct sigaction sa = { .sa_handler = sa_handler, .sa_flags = SA_ONSTACK };
sigaction(SIGUSR1, &sa, NULL);
execve(…);   /* ? */
```

POSIX improvement over time



/dev/random

- On topic, I’m happy to announce:
  - /dev/random is universal.
  - /dev/urandom is universal.
- They both exist *everywhere*.

Critical Mass

- My idea is to have enough useful data.
- Operating systems developers will find it.
- All results are open data published as .json.
- Developers will convince themselves and I won’t need to report + debate on thousands of bugs.
- And all systems will become slightly better. :) !

Community project

- Send a MR if you want portability data for any topic you're interested in! Send a .c and I'll run it everywhere!
- os-test is now used for Sortix, Managarm, and Redox development. We're competing for top scores.
- os-test makes it possible to develop new quality POSIX systems and improve the existing ones.

The best POSIX system?

- Yep. It’s musl libc + the Linux kernel.
- But other systems score very highly too.
- If you want to try out the POSIX 2024 additions, Sortix has 90% of them.

Thank You

- Go check out os-test, tell your osdev friends :)
- Jonas Termansen, sortie@maxsi.org
- https://sortix.org/os-test/
- https://gitlab.com/sortix/os-test/
- @sortiecat on Twitter, @jonastermansen on IG
- Meet me in the hallway for a chat! :)