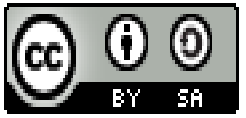
Not breaking userspace: the evolving Linux ABI

Alison Chaiken
Southern California Linux Expo 2022
alison@she-devel.com

Slides: http://she-devel.com/Chaiken_LinuxABI.pdf



Blue text indicates hyperlinks.

Why do kernel upgrades make users nervous?

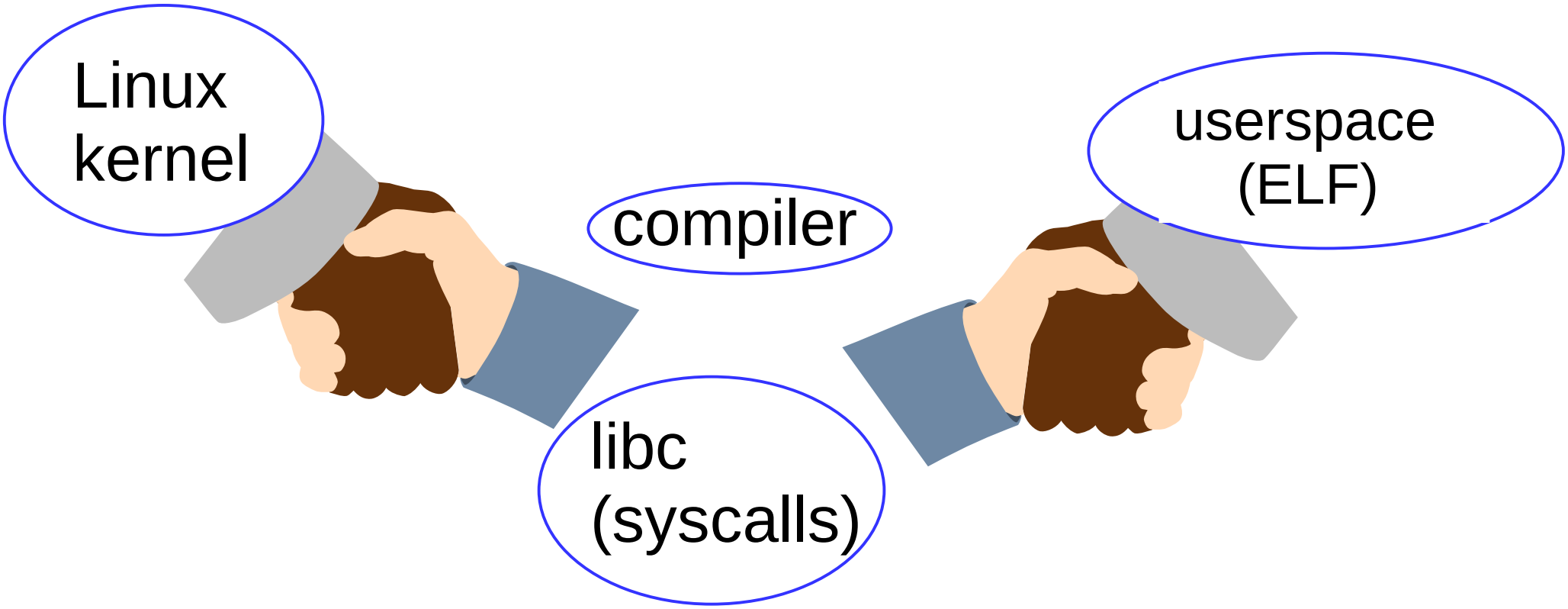
Contents

- What is an ABI and what is Linux's?
- Why does it matter?
  - 64-bit time (Y2038)
  - Priority inversions and robotics
  - Writing and maintaining log monitors
  - Mounting old filesystems
  - Mixing BPF programs and udev rules
- How to avoid ABI breaks
- Summary

What is an Application Binary Interface?

- Library headers and documentation are the API.
- For C programs in general:
  - return types of functions;
  - parameter lists of functions;
  - layout of structs
  - meaning, range or ordering of enums.
- Lack of “name mangling” in C is a problem.

ABI Guarantees are a Contract




The Linux Kernel Warranty


Seriously, the "we don't break user space" is the #1 rule in the kernel, and people should _know_ it's the #1 rule.

If somebody ignores that rule, it needs to be escalated to me. Immediately. Because I need to know.

I need to know so that I can override the bogus NAK, and so that we can fix the breakage ASAP. The absolute last thing we need is some other user space then starting to rely on the new behavior, which just compounds the problem and makes it a *much* bigger problem.

But I also need to know so that I can then make sure I know not to trust the person who broke rule #1.

This is not some odd corner case for the kernel. This is literally the rule we have lived with for *decades*.

So please escalate to me whenever you feel a kernel developer doesn't follow the first rule. Because the code that broke things *will* be reverted (*).

Y2038 and 64-bit time

tinyurl.com/mw94ktpw

32-bit IoT devices have a long service life

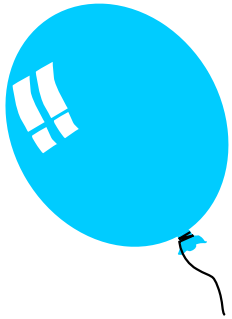
Disclaimer: these devices may not run Linux, or be 64-bit, or already updated.

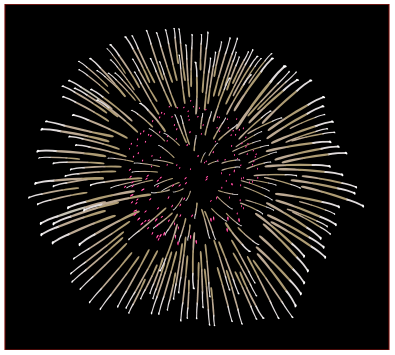
Good News: Kernel and Glibc are ready!

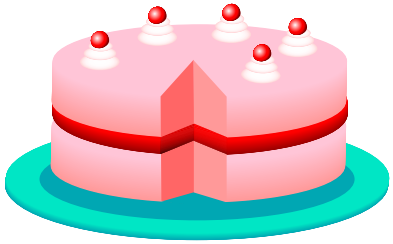
GNU C Library Lands Year 2038 Handling For Legacy ABIs

Written by Michael Larabel in GNU on 16 June 2021 at 12:00 AM EDT. 24 Comments

With recent versions of the Linux kernel and other key user-space software components, the Year 2038 handling is in relatively good shape and now this Glibc work for proper handling with the legacy ABIs now more thoroughly handles it. There is still seventeen years to go until the "Epochalypse" but particularly on the embedded front it remains to be seen how many systems/hardware will be updated by the vendor to the mitigate the issue.







Two bad choices for distros with 32-bit arches

A. complete set of 32- **and** 64-bit-time *packages*;

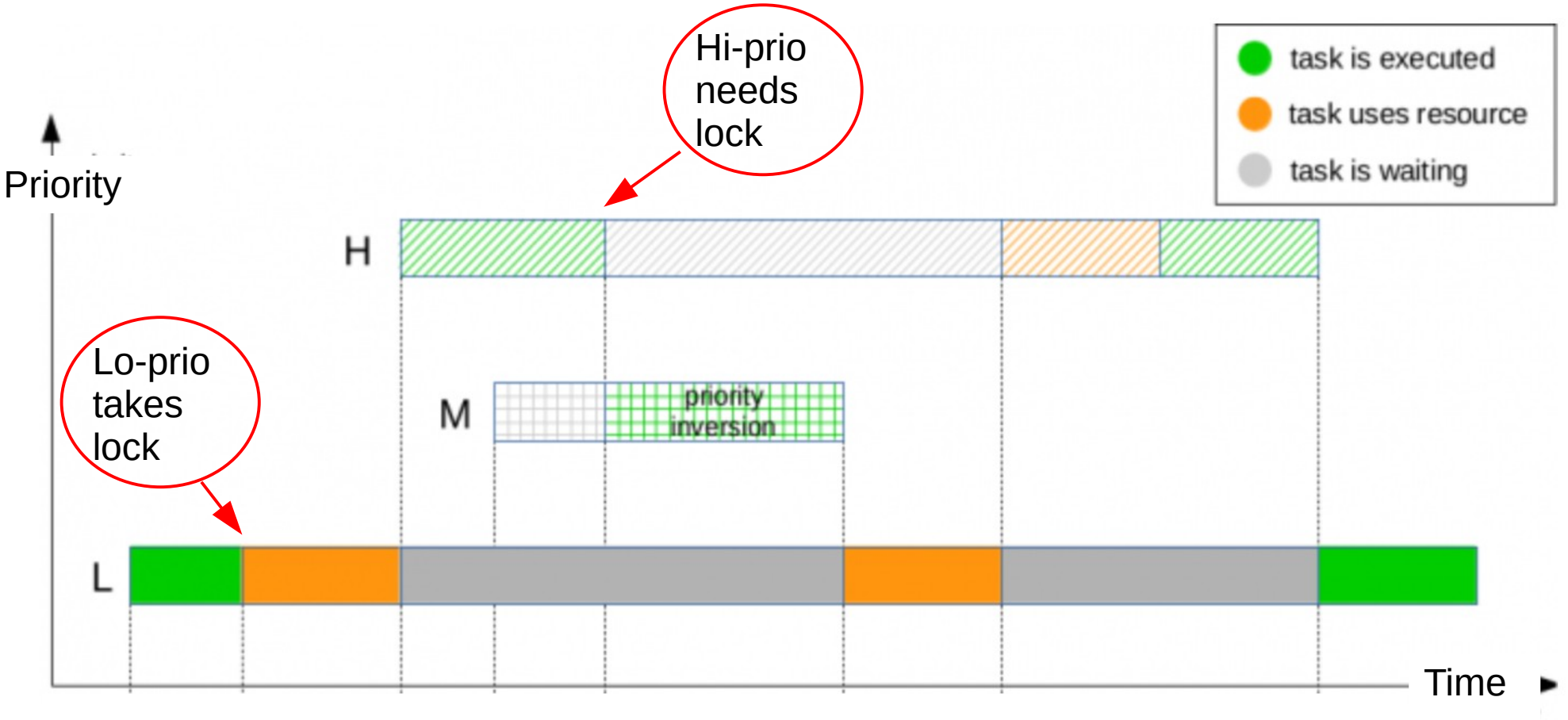
OR

B. distinct 32- and 64-bit-time *images*: i386 and i386t

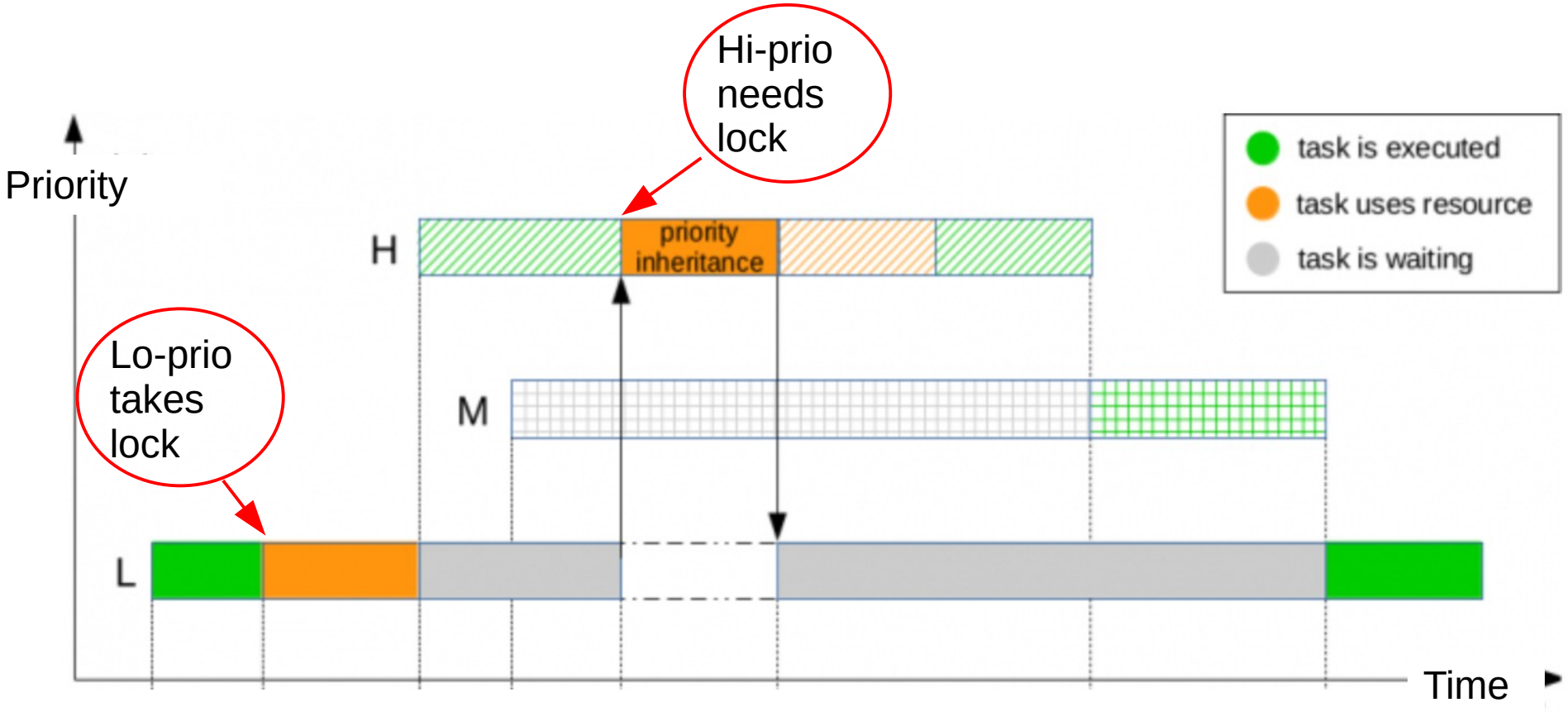
https://tinyurl.com/mryr56ee

Servers 1, Robots 0

Priority inversion occurs when L-prio thread holds a lock that H-prio thread needs

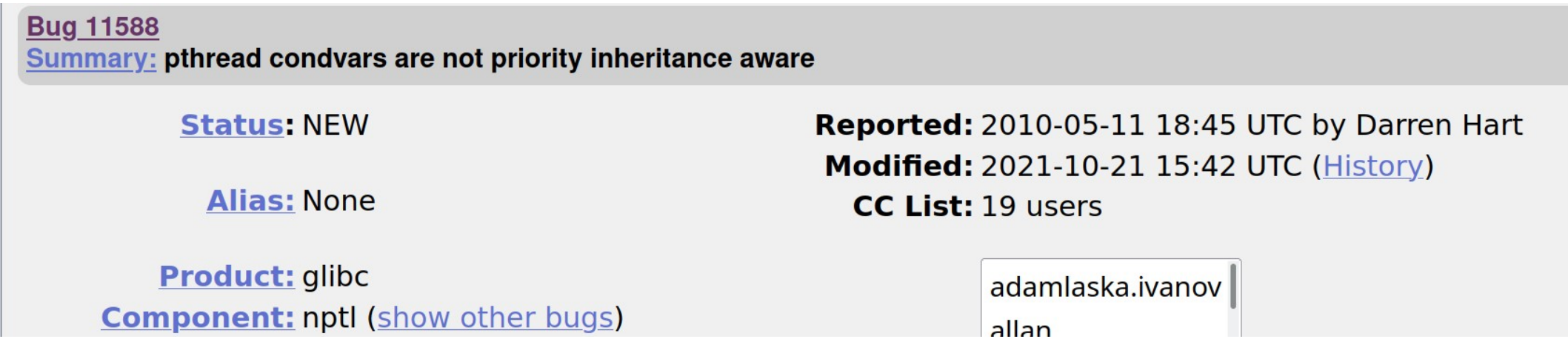


Priority Inheritance in Realtime Kernel Boosts the L-prio thread temporarily



Vital for userspace tasks as well as kernel!

Problem: glibc threads do not support PI



Bug 11588
Summary: **pthread condvars are not priority inheritance aware**

Status: NEW

Alias: None

Product: glibc
Component: nptl (show other bugs)

Reported: 2010-05-11 18:45 UTC by Darren Hart
Modified: 2021-10-21 15:42 UTC (History)
CC List: 19 users

adamlaska.ivanov
allan

Bug report posted in 2010; still “NEW” in 2022.

Fix Requires an ABI Break

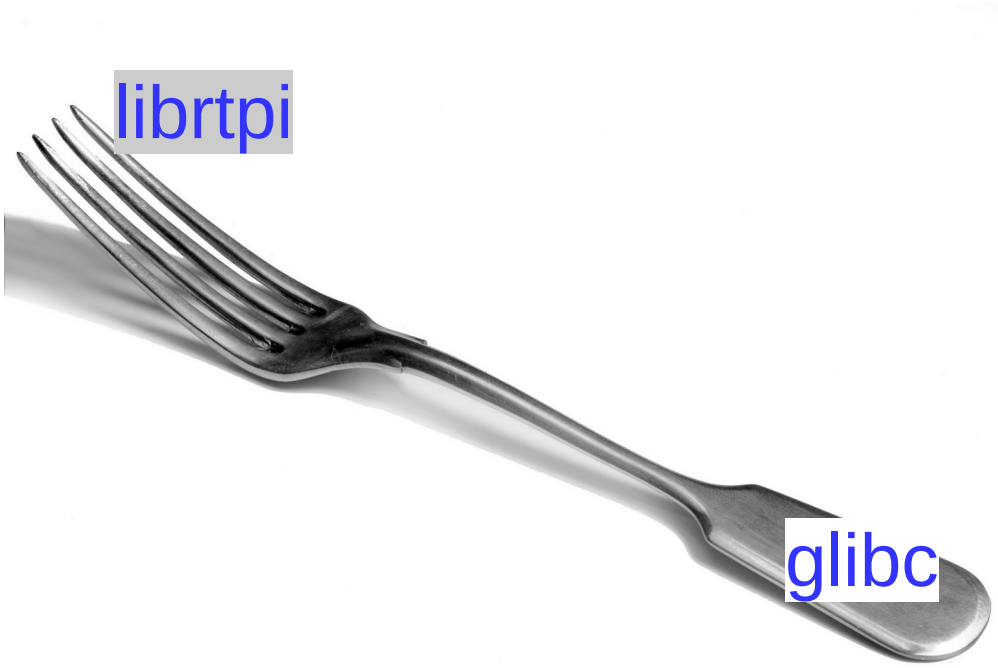
Torvald Riegel 2017-01-11 11:50:41 UTC **Comment 56**

```
So far, there is no known solution for how to achieve PI support given the current
constraints we have (eg, available futex operations, POSIX requirements, ...).
```

Either:

- a syscall must take an additional argument;
  -or-
- a function arg (the “futex word”) must change from 32- to 64-bit.

Robotics projects need a different pthread library



Or an entirely different libc!

tinyurl.com/2p99dtan, https://tinyurl.com/2rmmmban

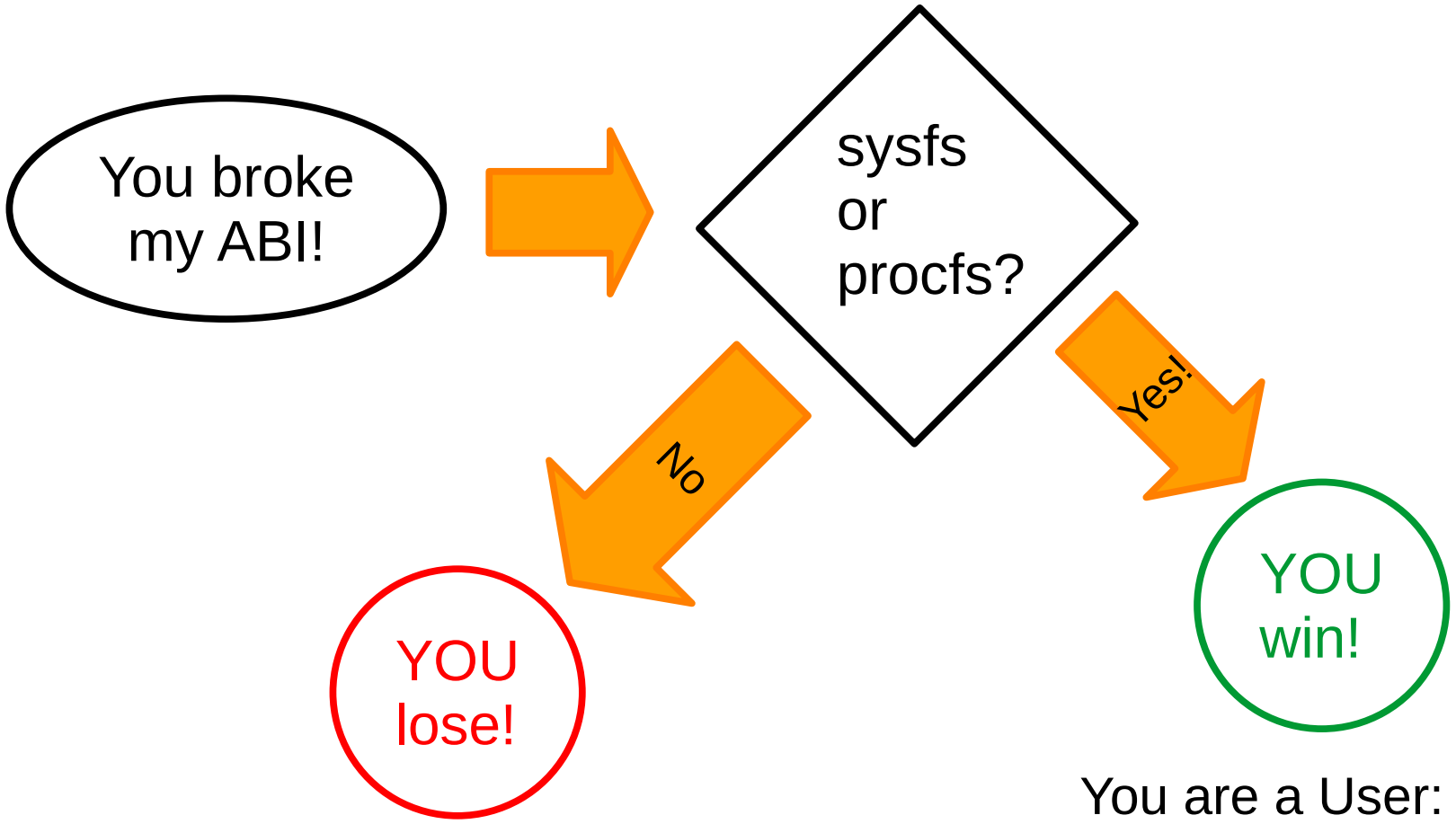
What is in the Linux Kernel Stable ABI?

Advertised: ELF, /proc, /sys and device-tree.

What about :

- Devices in /dev?
- printk output (dmesg)?
- Bootargs (cmdline)?
- Module params?
- Netlink sockets?
- Tracepoints?
- Valid BPF programs?
- Filesystem metadata?
- *Meaning* of constants in headers?

Developers are **not** covered by the Warranty



You are a *Developer:*
“I want to debug and optimize”.

You are a User:
“I want it to *just work*!”

dmesg, GDB and log monitors

Threading Kernel Log Messages

From Tetsuo Handa <>

Subject [PATCH] printk: Add caller information to printk() output.

Date Sat, 24 Nov 2018 16:37:55 +0900

Sometimes we want to print a whole line without being disturbed by
concurrent printk() from interrupts and/or other threads, for printk()
which does not end with '\n' can be disturbed.

Some examples for /dev/kmsg output:

```
6,21,0,-,from=T0;NX (Execute Disable) protection: active
6,22,0,-,from=T0;SMBIOS 2.7 present.
6,23,0,-,from=T0;DMI: VMware, Inc. VMware Virtual Platform/440BX Desktop Reference Platform, BIOS 6.00 05/19/2017
6,24,0,-,from=T0;tsc: Fast TSC calibration using PIT
6,25,0,-,from=T0;tsc: Detected 2793.552 MHz processor
7,26,858,-,from=T0;e820: update [mem 0x00000000-0x00000fff] usable ==> reserved
7,27,864,-,from=T0;e820: remove [mem 0x000a0000-0x000fffff] usable
6,28,882,-,from=T0;last_pfn = 0x140000 max_arch_pfn = 0x400000000
7,29,953,-,from=T0;MTRR default type: uncachable
6,512,753757,-,from=T1;PCI: Using host bridge windows from ACPI; if necessary, use "pci=nocrs" and report a bug
6,513,757721,-,from=T1;ACPI: Enabled 4 GPEs in block 00 to 0F
```

And then fix GDB

From Joel Colledge <>

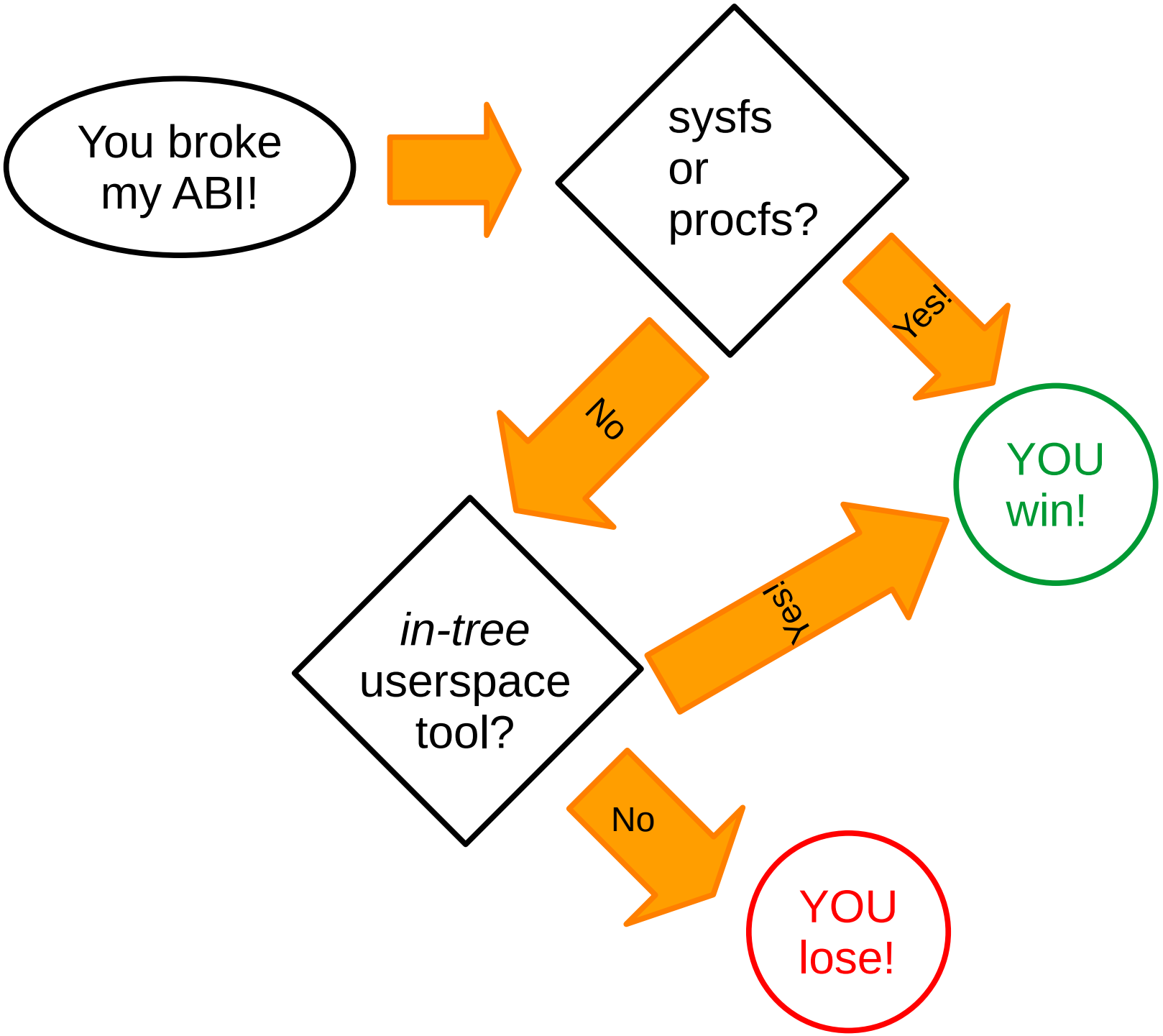
Subject [PATCH] scripts/gdb: fix lx-dmesg when CONFIG_PRINTK_CALLER is set

Date Wed, 25 Sep 2019 17:03:08 +0200

When CONFIG_PRINTK_CALLER is set, struct printk_log contains an
additional member caller_id. As a result, the offset of the log text is
different.

This fixes the following error:

```
(gdb) lx-dmesg
Python Exception <class 'ValueError'> embedded null character:
Error occurred in Python command: embedded null character
```


You broke my ABI!
sysfs or procfs?
Yes!
No
YOU win!
in-tree userspace tool?
Yes!
No
YOU lose!

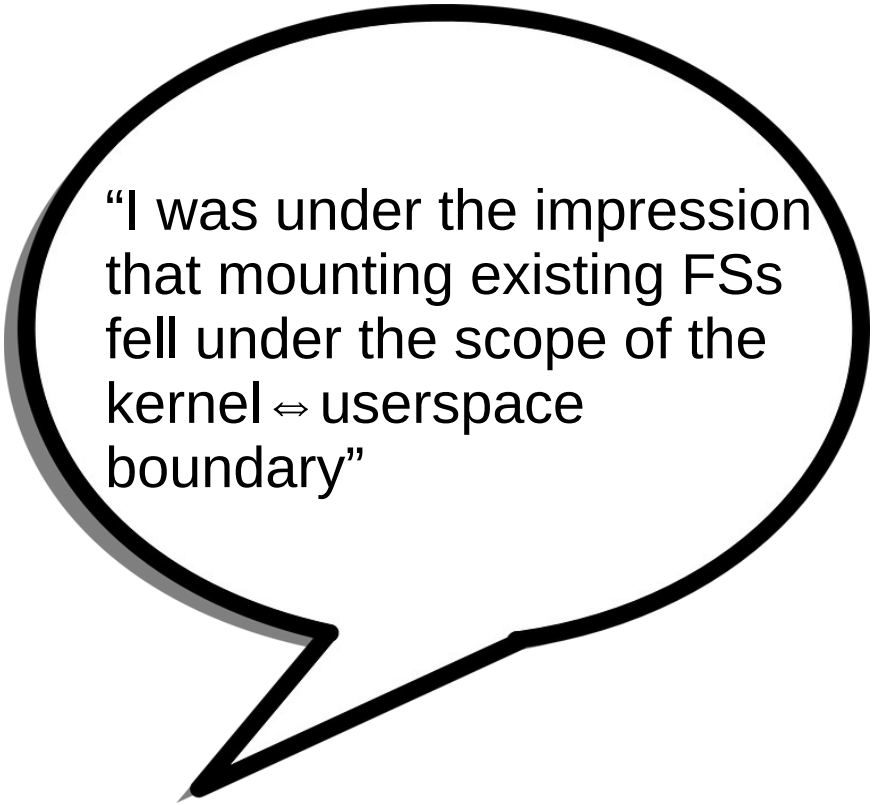
Kernel ABI Stability Checklist

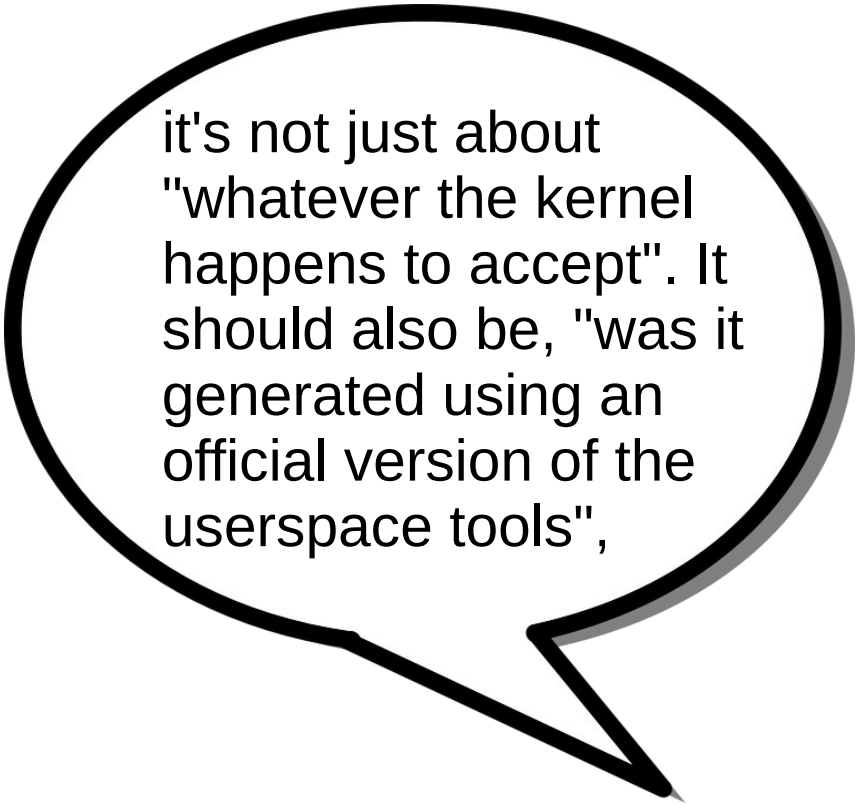
- ELF format: yes, /proc: yes, /sys: yes†
- ~~Device files in *dev?* (*not* /dev/kmsg)~~
- ~~printk output (dmesg)?~~
- Filesystem metadata?
- Valid BPF programs?
- Tracepoints?
- Bootargs (cmdline)?
- Module parameters?

†except /sys/kernel/debug, /sys/kernel/btf . . .

Mounting Old Filesystems

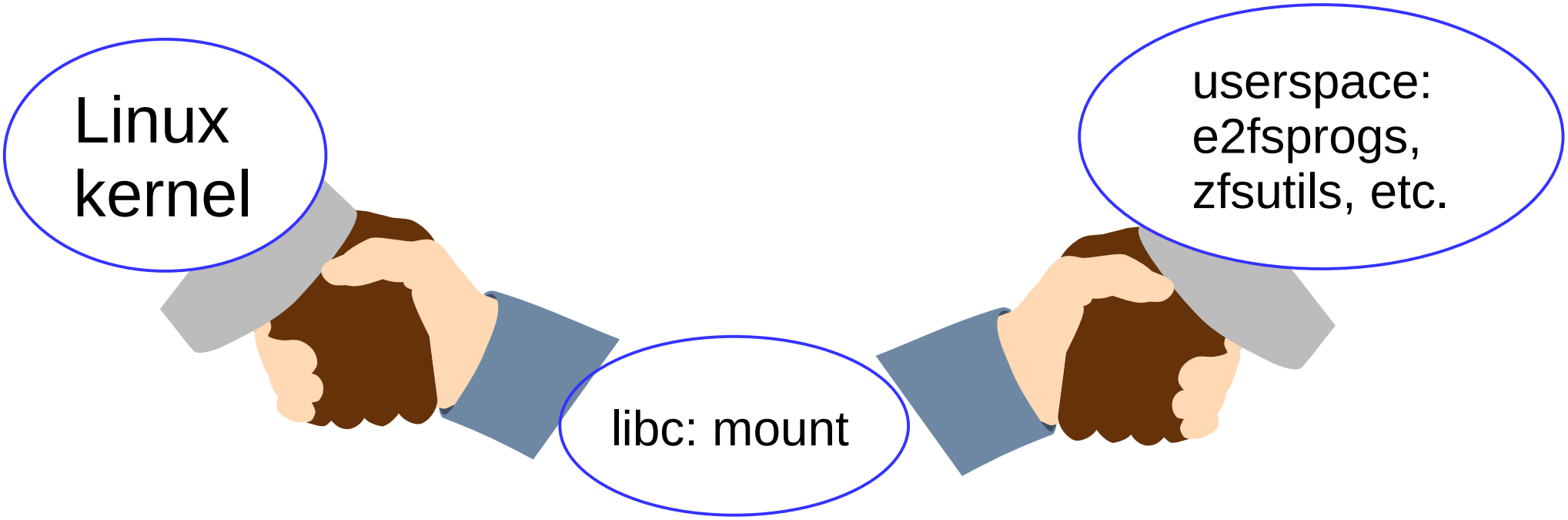
Will old filesystems mount with new kernels?





Manipulating FS metadata with custom tool relies on *Undefined Behavior*.

ABI Contract holds if *official* userspace tools are used



What about FS from Windows, iOS, BSD?

Kernel ABI Stability Checklist

- ELF format: yes, /proc: yes, /sys: yes†
- ~~Device files in *dev?*~~ (*not* /dev/kmsg)
- ~~printk output (dmesg)?~~
- ~~Filesystem metadata?~~ only with official tools
- Valid BPF programs?
- Bootargs (cmdline)?
- Module parameters?
- Tracepoints?

†except /sys/kernel/debug, /sys/kernel/btf . . .

Are BPF Programs Part of Stable ABI?

BPF's ABI Guarantee

Q: Directly calling kernel function is an ABI?

Q: Some kernel functions (e.g. tcp_slow_start) can be called by BPF programs. Do these kernel functions become an ABI?

A: NO.

Controversy: can an *ordinary user* tool like a udev rule depend on a *developer* tool like bpftrace?

What about Kernel Features on which BPF Programs Rely?

“out-of-tree modules will have the technical debt of changing every time a new kernel release is out, and so should out-of-tree bpf programs.”

Kernel Summit mailing list
6/16/22

Stable ABI is any “"**user** workflow", whether it uses BPF or any other kernel feature’

Photo: https://lwn.net/Articles/899182/

OSS 2022,
6/21/22


Avoiding ABI Breaks


Method 0: generic syscalls

prctl()

ioctl()

https://tinyurl.com/bdd5yeys

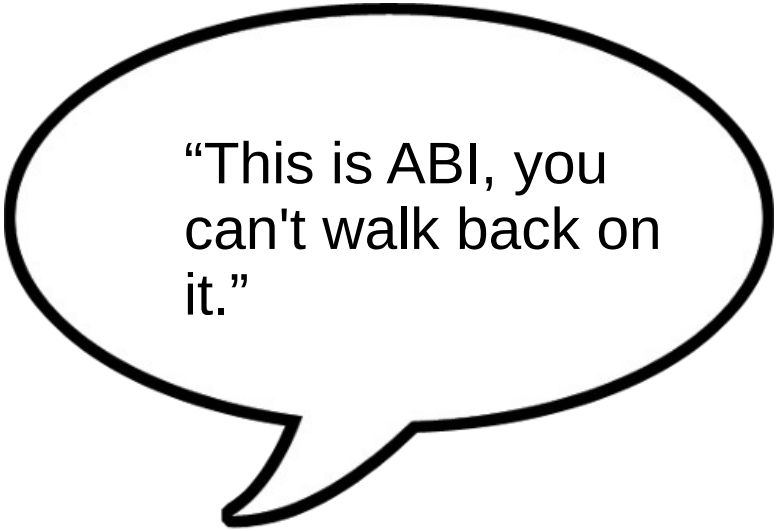
```
int prctl(int option, unsigned long arg2, unsigned long arg3,
          unsigned long arg4, unsigned long arg5);

int ioctl(int fd, unsigned long request, ...);
```

Method 1: Unused Function Parameters

- Retain *deprecated* params: *man shmctl*
- "*Reserved for future use*" params: *man pid_getfd*
- Bitmasks with *unused bits*: *man statx*
- Multiple versions of the same syscall*: apropos clone*
- Some *unwrapped* syscalls have no glibc implementation.

Method 2: avoid procfs and sysfs changes



https://pixabay.com/en/speech-bubble-ellipse-shape-arrow-25917/

Summary

- The Linux ABI definition is both fuzzy and controversial.
- Some ABI breaks are unavoidable.
- Y2038 fix will be painful for distros.
- glibc’s thread library NPTL is *not* suitable for robotics.
- Trouble arises when *users* rely on *developer* tools.
- Think carefully when writing userspace applications!

Acknowledgements

Thanks to Akkana Peck and Sarah Newman for their feedback.

Live Patching: a Down in the Trenches View

Sarah Newman

Today, Saturday July 30, at 6 PM in this same room!

Many full-time-remote Linux jobs with Aurora Innovation (aurora.tech)

Software Engineer - Simulation Web Tools
Mountain View

HIL Software Engineer
Mountain View

HIL Software Engineer
Pittsburgh

Linux Kernel Engineer
Pittsburgh

Linux Kernel Engineer
Mountain View

Senior Software Engineer, Visualization Team
Pittsburgh

Staff/ Senior Fullstack Engineer- Perception Datasets
Mountain View

Staff Software Engineer
Pittsburgh

Staff Software Engineer - Performance Engineering Optimization
Mountain View

Vehicle Platform Engineer
Seattle

Vehicle Platform Engineer
Pittsburgh

Resources

- Please support LWN!
- “To Save C, We Must Save ABI,” and “
  Binary Banshees and Digital Daemons,” JeanHeyd Meneide.
- “It’s About Time,” Mike Crowe, *Overload*, 28(156):28-30, April 2020. (about pthreads and condition variables)
- Gratian Crisan,
  librtpi: Conditional Variables for Real-Time Applications,
  Embedded Linux Conference 2020 presentation
- “BPF CO-RE (Compile Once, Run Everywhere),” Andrii Nakryiko’s blog

Changing *Meanings* of Names

Behavior of Clocks during Suspension

Should CLOCK_MONOTONIC advance when the system is suspended?

```
There might be side effects in applications, which rely on the
(unfortunately) well documented behaviour of the MONOTONIC clock, but the
downsides of the existing behaviour are probably worse.
```

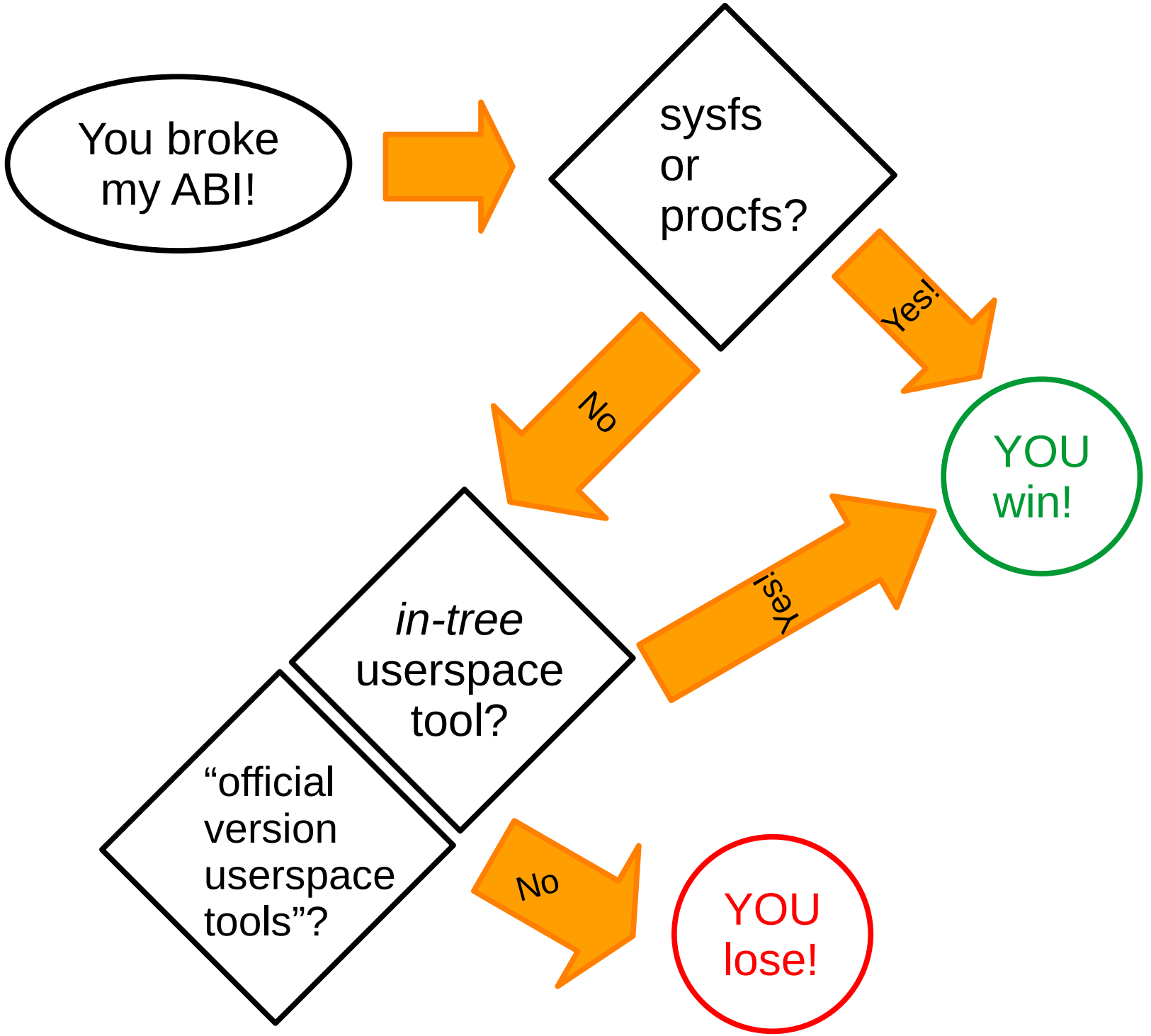
> **Possible side-effects of CLOCK_MONOTONIC change?**
> Posted Apr 17, 2018 5:51 UTC (Tue) by **epa** (subscriber, #39769) [Link]
>
> If I read the article correctly, it could be summarized as “CLOCK_MONOTONIC has been renamed to CLOCK_MONOTONIC_ACTIVE, but the old name has now become an alias for something else”. It all seems a bit strange, especially given the declared intention never to break user space.

Changing the meaning of bootargs

- See bootargs with sudo dmesg | grep -i “command line”
- Should bootarg A means something different alone than in conjunction with bootarg B?

  $ git log -n 1 --grep=consistent – kernel/rcu/tree_nocb.h
- Someone preferred the old behavior:

  $ git log --oneline -n 1 --grep="option to opt" – kernel/rcu


You broke my ABI!
sysfs or procfs?
Yes!
No
YOU win!
in-tree userspace tool?
Yes!
“official version userspace tools”?
No
YOU lose!

Trace Events are “stable API”

- The *list* of tracepoints is (mostly) stable.
- The *format* of trace events (the ABI!) changes.

The difference between a tracepoint and trace event is that a *tracepoint* is the "trace_foo()" in the kernel, whereas the *trace event* is the data extracted from the *tracepoint* via the TRACE_EVENT() macro and listed in the format files in the tracefs events directory.

Why kernel tracepoints disappear: optimizing compilers

#linux-rt IRC chat:

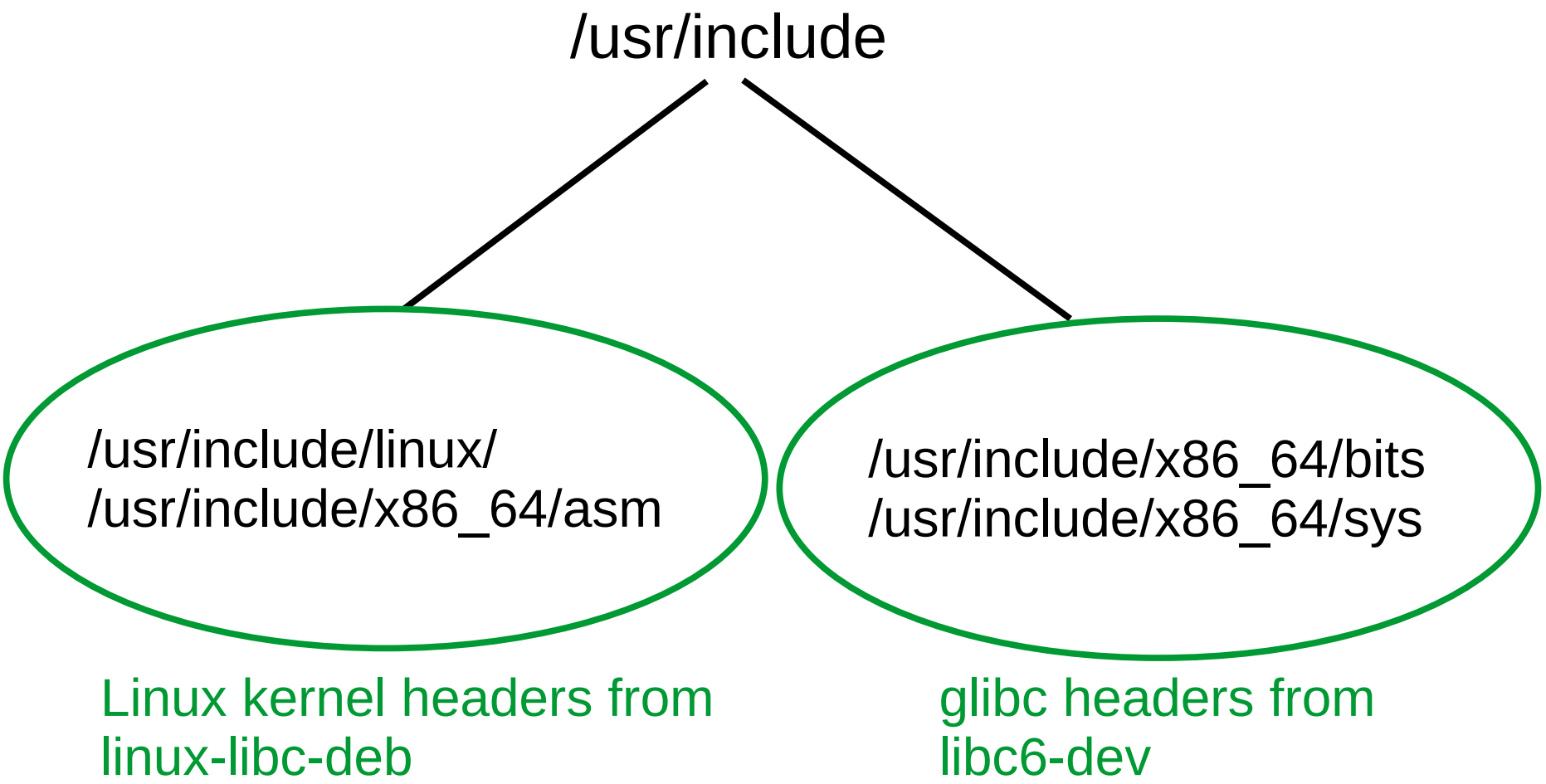
<rostedt> I already get people asking me why a function is traced in one kernel, but not the other

<rostedt> and I have to tell them it was because gcc decided to inline it in the other kernel

<rostedt> now we are going to lose tracing, because a function was modified slightly where gcc can do a tail call?

<rostedt> and tail calls are quite common

The libc ⇔ kernel ABI boundary



BPF Programs and "Compile Once, Run Everywhere"

- Problem: BPF programs rely on *internal* kernel ABI.
- Solution: BPF Type Format, with debug info exported to /sys/kernel/btf:
  - $ bpftool btf dump file /sys/kernel/btf/battery format c
- CO-RE further has LINUX_KERNEL_VERSION and CONFIG_FOO_FEATURE macros to help with more complex changes.

Printk indexing will ease log-monitor maintenance

```
+============
+Printk Index
+============
+
+There are many ways how to monitor the state of the system. One important
+source of information is the system log. It provides a lot of information,
+including more or less important warnings and error messages.
+
+There are monitoring tools that filter and take action based on messages
+logged.
+
+The kernel messages are evolving together with the code.  As a result,
+particular kernel messages are not KABI and never will be!
```

Solution: export kernel format strings to /sys/kernel/debug for use by log monitors.

Method 3: Export info to sysfs (BTF and printk indexing)

How to access the ABI of the *currently running* kernel:

```
$ ls /sys/kernel/btf

$ bpftool btf dump file /sys/kernel/btf/vfat

$ bpftool btf dump file /sys/kernel/btf/battery format c
```

Not part of stable ABI!

Method 4: JIT Compilation

Works great, but expensive:

- requires kernel headers to be updated along with kernel;
- requires compilation with libclang or libgcc;
- slows startup of applications while JIT runs.

Who ever heard of futexes?

VERSIONS

Futexes were first made available in a stable kernel release with Linux 2.6.0.

CONFORMING TO

This system call is Linux-specific.

NOTES

Several higher-level programming abstractions are implemented via futexes, including POSIX semaphores and various POSIX threads synchronization mechanisms (mutexes, condition variables, read-write locks, and barriers).

musl has had PI mutexes since 2019

https://git.musl-libc.org/cgit/musl/tree/WHATSNEW

```
2040
2041 1.1.22 release notes
2042
2043 new features:
2044 - priority-inheritance mutexes
```

C language lacks features that make ABI stability easier

- There are no function *overload*s or *override*s.
  - No “name mangling.”
  - To safely change a function’s ABI, one must rename it.
- “Encapsulation” methods are limited to **static** keyword.
  - No notion of granular privacy.
  - Implementation details tend to “leak” from C programs.
- Maybe “assembly labels” like **__attribute__((alias()))** are a path forward?

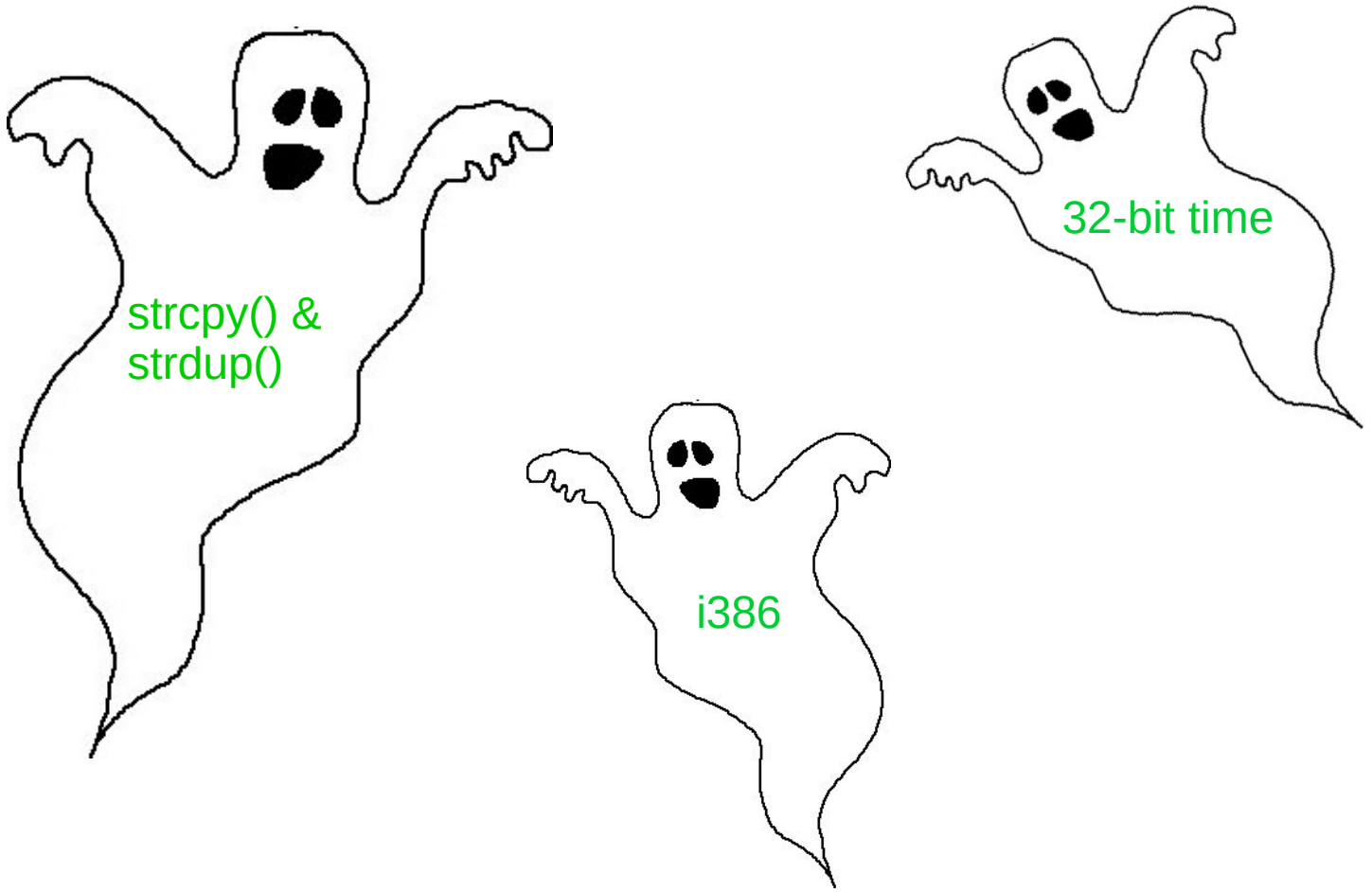
Undefined Behavior with FS Metadata

Ran into an ext4 regression when testing upgrades to 5.9-rc kernels:

Commit e7bfb5c9bb3d ("ext4: handle add_system_zone() failure in ext4_setup_system_zone()") breaks mounting of read-only ext4 filesystems with intentionally overlapping bitmap blocks.

On an always-read-only filesystem explicitly marked with EXT4_FEATURE_RO_COMPAT_SHARED_BLOCKS, prior to that commit, it's safe to point all the block and inode bitmaps to a single block of all 1s, because a read-only filesystem will never allocate or free any blocks or inodes.

Backward compatibility is hard



http://stackoverflow.com/questions/28986125/double-curved-shape

Why not just recompile?

- Source code is unavailable.
- Need to recompile propagates to other libraries/applications.
- Proprietary toolchain is too old to support needed features.
- Code would need new FDA, ISO262 … certification.
- Suppose recompiling is ***not enough***?

ABI-Break Demo

$ git clone https://github.com/chaiken/SCALE2022-demo-code

$ cd SCALE2022-demo-code/C-ABI

$ make right-abi-lib

$ $ make wrong-abi-lib

Makefile

is_negative_64.c

is_negative_128.c

main.c

What about the asterix?

(*) Yes, there are exceptions. We have had situations where some interface was simply just a huge security issue or had some other fundamental issue. And we've had cases where the breakage was just so old that it was no longer fixable. So even rule #1 can sometimes have things that hold it back. But it is *very* rare. Certainly nothing like this.

How *should* libc's respond to kernel ABI Break?

Security, Performance, Stability, POSIX:
pick ~~2~~3

(forgetting readability, logical consistency . . .)

Debian and Out-of-Tree Kernel Modules

The kernel ABI

An ABI (Application Binary Interface) is an interface between two software components, considered at the level of register allocation and memory layout. The ABI between the kernel and user-space is generally maintained carefully, and is not a concern here. However, the ABI between the kernel and its modules is not. In order to support out-of-tree modules, the kernel version should be changed when the ABI between the kernel and modules changes.