

The State of the Kconfig Ecosystem

Julian Braha



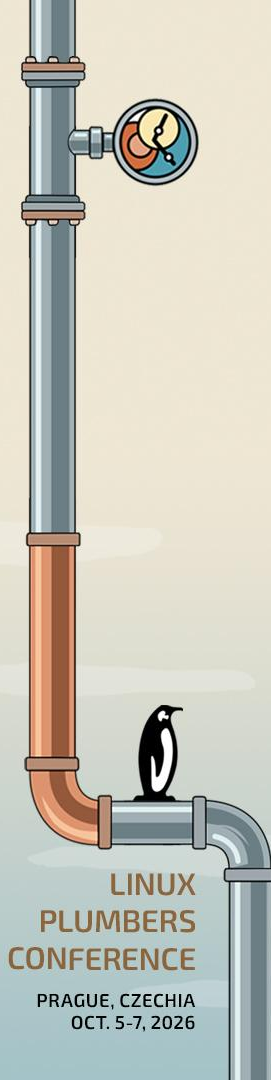
LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Part 1: The State of Tooling

In-tree:

- localmodconfig
- kconfig-sym-check.pl
- checkkconfigsymbols.py



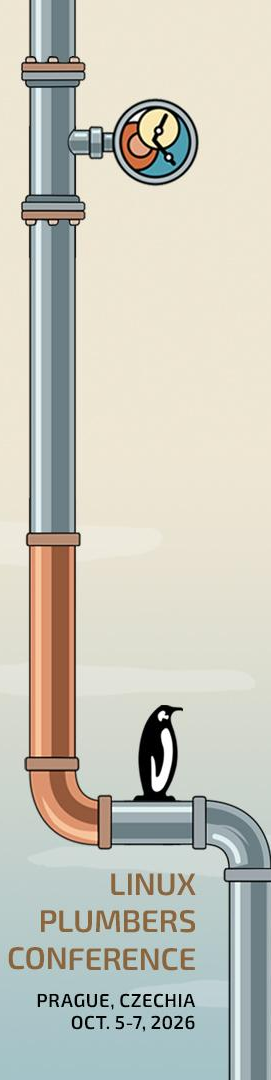
LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Part 1: The State of Tooling

Out-of-tree:

- kmax
 - klocalizer
 - koverage
 - krepair
 - kismet
- kconfirm
 - kconfirm-smt
- nom-kconfig



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

The Kmax Suite

klocalizer:

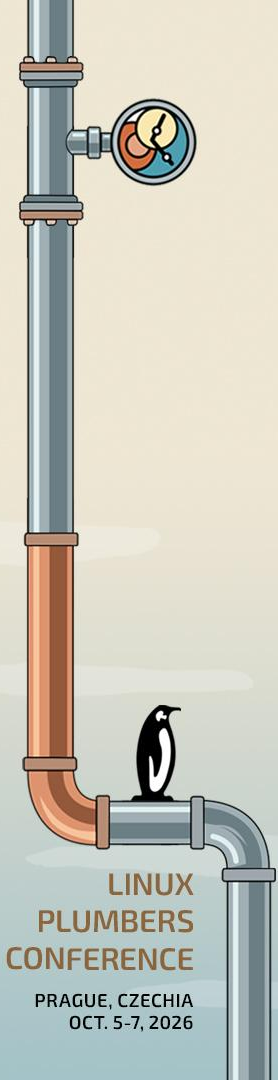
- Generate a configuration that will compile a file / line of C code

koverage:

- Checks if your configuration will compile the lines affected by a patch

krepair:

- Tweaks your configuration to compile the lines affected by a patch



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

kconfirm: a Static Analyzer for Kconfig

Linting:

- Duplicate dependencies
- Dead code
- “select-visible”

Outdated documentation

- Dead links

```
1 config SYS_L3_SIZE
2     int
3     default 262144 if SYS_L3_SIZE_256KB
4     default 524288 if SYS_L3_SIZE_512KB
5     default 1048576 if SYS_L3_SIZE_512KB
```

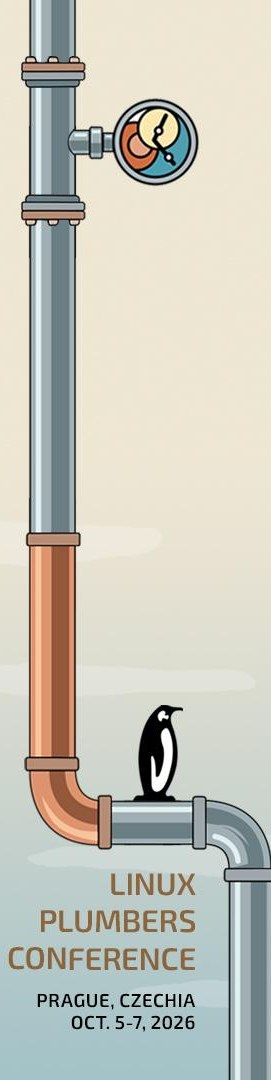
kconfirm-smt: Not Just Heuristics

SMT-solving Kconfig:

- Detect dead options
- Detect “undead” options
- Dead subranges for int/hex options
- Detect wrong types (e.g. tristate vs bool)
- Detect Unmet Dependency bugs

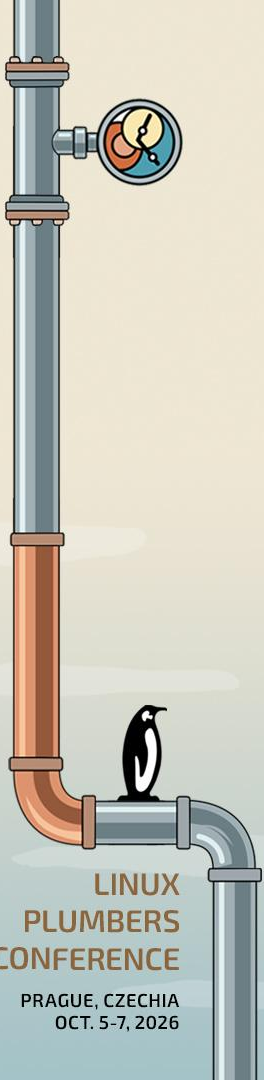
What's Still Missing

1. Revive dead tools
 - a. Vampyr
 - b. tree-sitter-kconfig
2. A style guide?
3. Formal semantics



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026



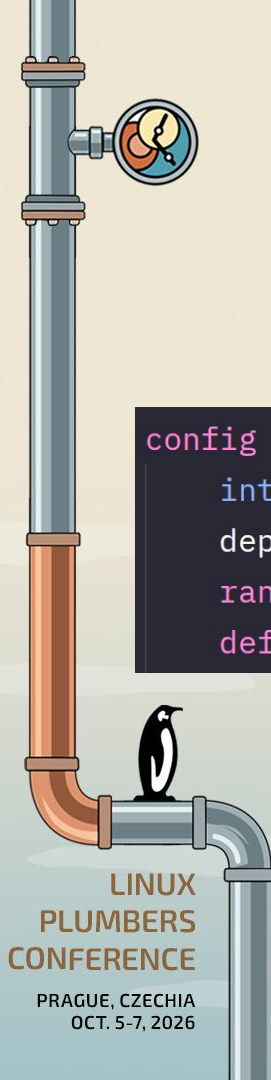
Style: Improve Readability by Grouping?

```
config GUEST_STATE_BUFFER_TEST
    def_tristate n
    prompt "Enable Guest State Buffer unit tests"
    depends on KUNIT
    depends on KVM_BOOK3S_HV_POSSIBLE
    default KUNIT_ALL_TESTS
```

Unifying Different Kconfig Styles?

```
config RV_PER_TASK_MONITORS
    int "Maximum number of per-task monitor"
    depends on RV
    range 1 8
    default 2
```

```
config RV_MON_SCHED
    depends on RV
    depends on RV_PER_TASK_MONITORS ≥ 3
    bool "sched monitor"
```



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Indent the Code like the Frontend?

```
<*> Second extended fs support (DEPRECATED)
[*]   Ext2 extended attributes
[*]   Ext2 POSIX Access Control Lists
[*]   Ext2 Security Labels
```

```
config EXT2_FS
    tristate "Second extended fs support (DEPRECATED)"
    select BUFFER_HEAD
    select FS_IOMAP

config EXT2_FS_XATTR
    bool "Ext2 extended attributes"
    depends on EXT2_FS

config EXT2_FS_POSIX_ACL
    bool "Ext2 POSIX Access Control Lists"
    depends on EXT2_FS_XATTR
    select FS_POSIX_ACL

config EXT2_FS_SECURITY
    bool "Ext2 Security Labels"
    depends on EXT2_FS_XATTR
```

```
config EXT2_FS
    tristate "Second extended fs support (DEPRECATED)"
    select BUFFER_HEAD
    select FS_IOMAP

config EXT2_FS_XATTR
    bool "Ext2 extended attributes"
    depends on EXT2_FS

config EXT2_FS_POSIX_ACL
    bool "Ext2 POSIX Access Control Lists"
    depends on EXT2_FS_XATTR
    select FS_POSIX_ACL

config EXT2_FS_SECURITY
    bool "Ext2 Security Labels"
    depends on EXT2_FS_XATTR
```

Semantics of Kconfig

Future kconfig work

Work on kconfig is welcomed on both areas of clarifying semantics

Semantics of Kconfig

The use of Kconfig is broad, Linux is now only one of Kconfig's users



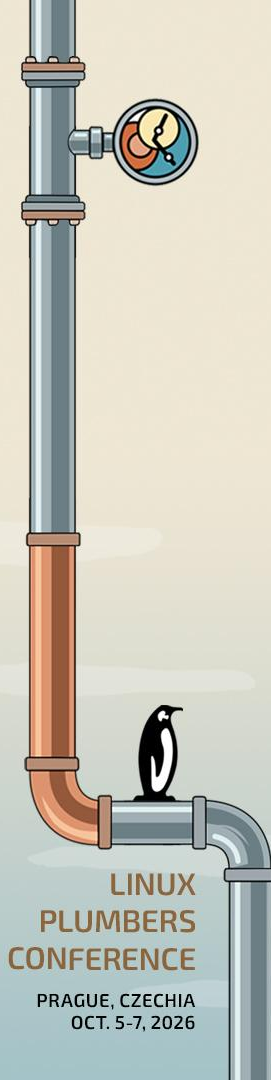
**LINUX
PLUMBERS CONFERENCE**

PRAGUE, CZECHIA / OCT. 5-7, 2026



But What Kind of Semantics?

- Denotational
- Mechanized
- Operational



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Semantics of Kconfig: Denotational?

- (E₁) $E[[\text{prompt } word \text{ if } expr]]\sigma \triangleq E[[expr]]\sigma$
- (E₂) $E[[depends+]]\sigma \triangleq \bigwedge_{expr_i \in depends+} (E[[expr_i]]\sigma)$
- (E₃) $E[[\text{default } val \text{ if } expr \text{ default}^*]]\sigma \triangleq \begin{cases} val = y & \text{if } E[[expr]]\sigma \\ E[[default^*]]\sigma & \text{if } default^* \neq \emptyset \\ \text{false} & \text{otherwise} \end{cases}$
- (E₄) $E[[expr_1 \&\& expr_2]]\sigma \triangleq E[[expr_1]]\sigma \wedge E[[expr_2]]\sigma$
- (E₅) $E[[expr_1 || expr_2]]\sigma \triangleq E[[expr_1]]\sigma \vee E[[expr_2]]\sigma$
- (E₆) $E[[! expr]]\sigma \triangleq \neg E[[expr]]\sigma$
- (E₇) $E[[sym]]\sigma \triangleq (\sigma(sym) = y)$

(e) Expression evaluation rules.

Semantics of Kconfig: Mechanized?

linux/tools/memory-model/

```
(* Visibility and executes-before for plain accesses *)
let ww-vis = fence | (strong-fence ; xbstar ; w-pre-bounded) |
  (w-post-bounded ; vis ; w-pre-bounded)
let wr-vis = fence | (strong-fence ; xbstar ; r-pre-bounded) |
  (w-post-bounded ; vis ; r-pre-bounded)
let rw-xbstar = fence | (r-post-bounded ; xbstar ; w-pre-bounded)

(* Potential races *)
let pre-race = ext & ((Plain * M) | ((M \ IW) * Plain))

(* Coherence requirements for plain accesses *)
let wr-incoh = pre-race & rf & rw-xbstar^-1
let rw-incoh = pre-race & fr & wr-vis^-1
let ww-incoh = pre-race & co & ww-vis^-1
empty (wr-incoh | rw-incoh | ww-incoh) as plain-coherence
```

Semantics of Kconfig: Operational

```
/*@  
  requires \valid(p);  
  requires \valid(q);  
  assigns *p, *q;  
  ensures *p == \old(*q);  
  ensures *q == \old(*p);  
*/  
static void swap(int *p, int *q) {  
    int tmp = *p;  
    *p = *q;  
    *q = tmp;  
}
```

Language Versioning

Linux v7.1: *depends on X if Y*

Linux v7.2 - v7.3: *fixes*

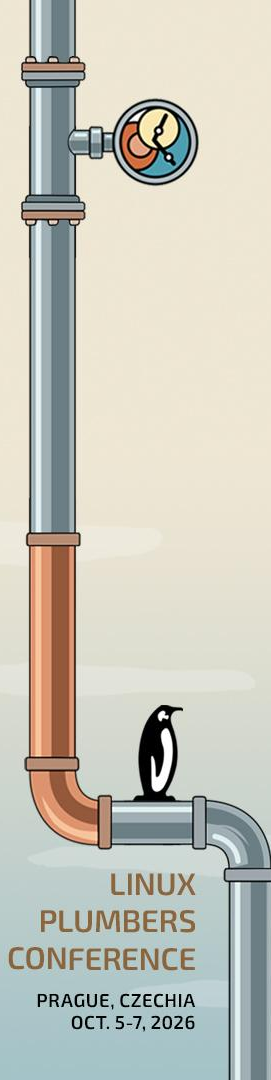
Linux v7.4: *def_int, def_hex, def_str*

What about...

Kconfig v19: *depends on X if Y*

Kconfig v19.1 - v19.2: *fixes*

Kconfig v20: *def_int, def_hex, def_str*



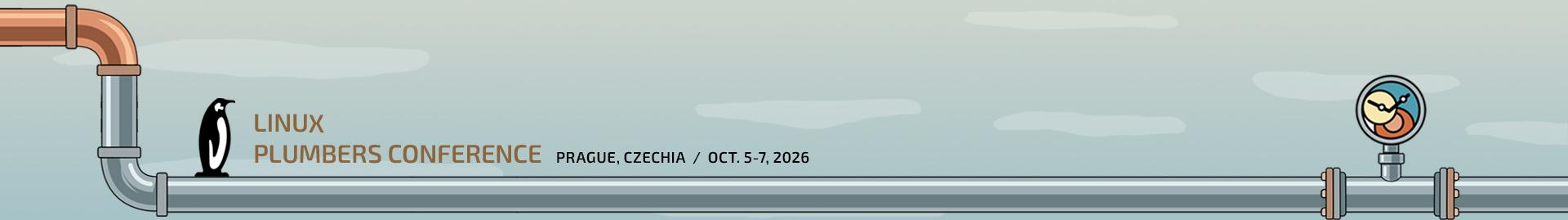
LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Part 2: Fragmentation

At least 44 known open source projects using Kconfig, including:

1. coreboot
2. Das U-boot
3. Zephyr
4. uClibc
5. BusyBox
6. ToyBox
7. QEMU
8. OpenWrt
9. L4Re / Fiasco
10. ESP-IDF



coreboot's Extension to Kconfig: *

Example:

```
choice
    prompt "Mainboard vendor"
    default VENDOR_EMULATION

    source "src/mainboard/*/Kconfig.name"

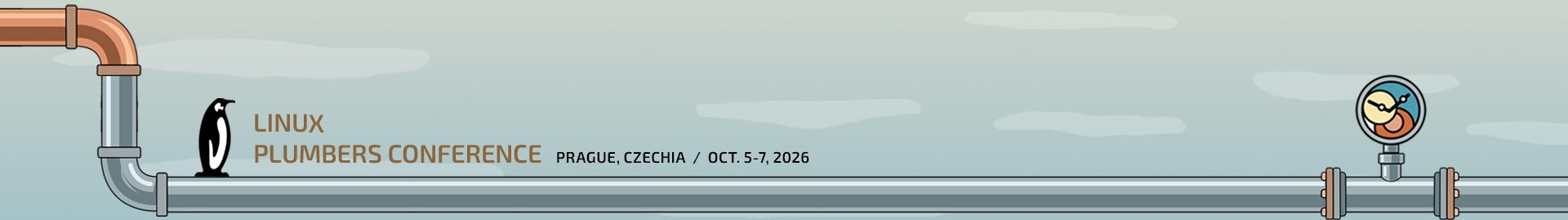
endchoice

source "src/mainboard/*/Kconfig"
```



U-Boot: Closer to an Old Kconfig

- Retains named choices
- Retains tristate choices
- Lacks 'transitional'
- Impl works differently



Kconfiglib's Extension to Kconfig

An `rsource` statement is available for including files specified with a relative path. The path is relative to the directory of the `kconfig` file that contains the `rsource` statement.

```
config F00
    bool "F00"
    depends on BAR

configdefault F00
    default y if FIZZ
```

For cases where it's okay for a pattern to match no files (or for a plain filename to not exist), a separate `osource` (*optional source*) statement is available. `osource` is a no-op if no file matches.

The *named choices* feature originates from Linux, but it is no longer supported in Linux since kernel release 6.9, and has thus become a Kconfiglib language extension.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



How do we Reconcile the Different Forks?

- Should the official Linux Kconfig support features for the rest of the ecosystem?
 - Other projects need types like floating-point, so they need to use an alternative to Kconfig
- Should Kbuild support other implementations of Kconfig?
- Should we package Kconfig for other projects to use?



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

