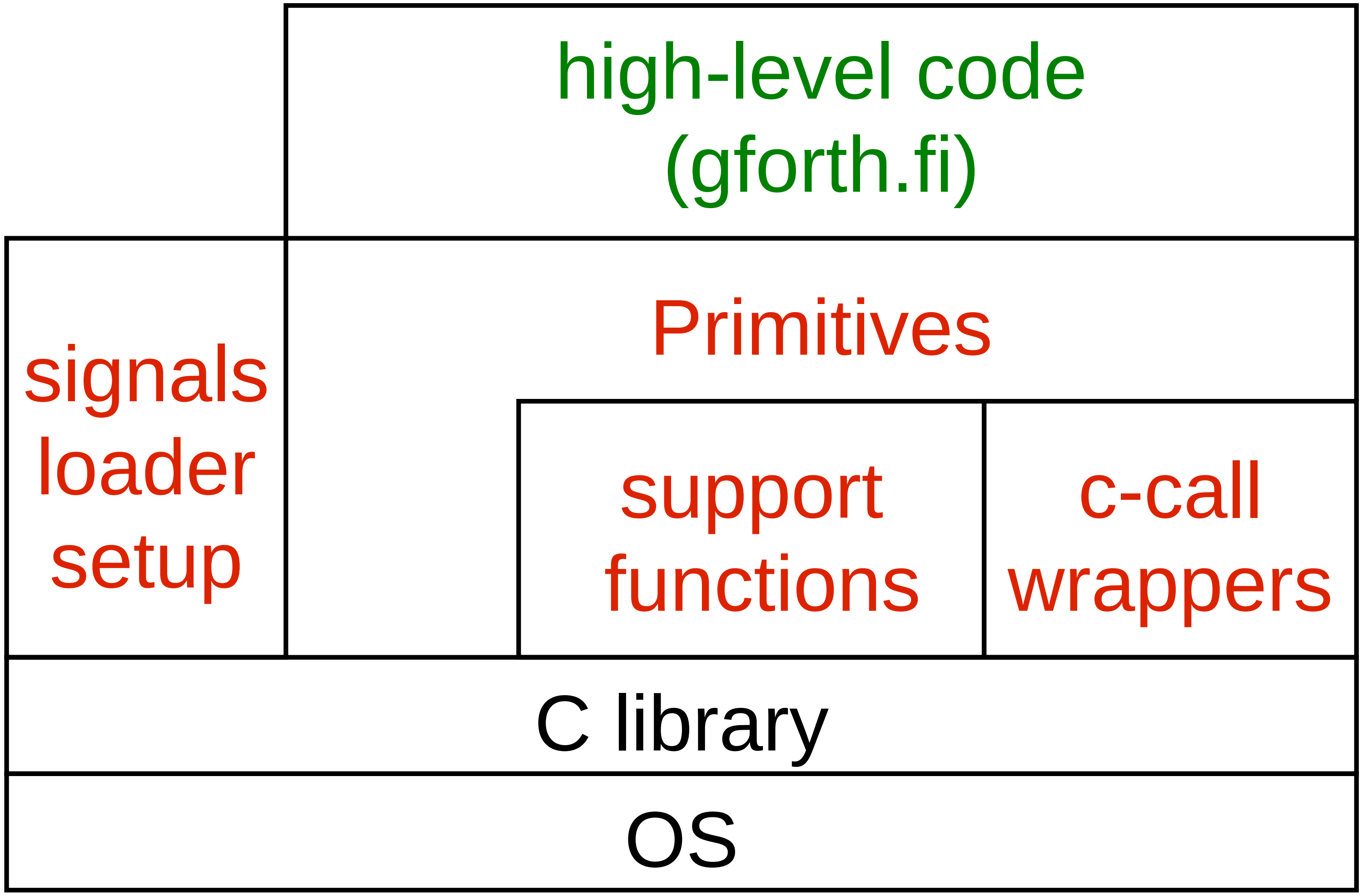
How to get rid of C

M. Anton Ertl
TU Wien

Problem: C has become unreliable

- 186 undefined behaviours in C standard
- every real-world program has them
- C compiler maintainers focus exclusively on
  programs without undefined behaviours
  benchmarks (SPEC)
- bug reports are not taken seriously
- $\Rightarrow$ We want to get rid of C

Gforth components



Primitives

- replace with native-code compiler on popular platforms

- keep existing primitives on other platforms
  $\Rightarrow$ we cannot get rid of C
  remove non-standard usage when gcc acts up
  no longer work around performance problems
  $\Rightarrow$ slowdown

- Or maybe some primitives in assembly language
  high-level replacement for others

Native-code compiler

- Still want to use image files

- Compiler from image files to native code

- For interactive use:
  Compiler from threaded-like code to native code
  threaded-like code allows storing image files

- For bootstrapping:
  Compiler from image files to assembly language

Support functions

- Called by primitives
  e.g. mixed division
- replaced by native-code compiler
- or high-level code

Calling C

- For system calls
  Alternative: direct system calls
  additional system-specific stuff to implement
  CPU-specific optimizations
- For library calls
- use wrappers like now?
- teach calling convention to native-code compiler
  Use `extern:` for specifying C functions

Setup, loader, signals

- Could be replaced with Forth code
  on systems with native-code compiler

- But: two versions to maintain

- not performance-sensitive
  Slowdown from C standards compliance should not be noticable

Conclusion

- Getting away from C is a long-term effort
- Is it worthwhile to get rid of C completely?