Instrument and Unit Test an Asm-only OS Kernel by Turning it into an Anykernel

or:

Move Parts of KolibriOS Kernel to Userspace

and Build some Tools around them

Ivan Baravy

FOSDEM'26

01 February 2026

Brussels, Belgium

Scope. The bleeding edge of the marginal activity

- Writing in assembly
  - Reading
  - Generating
  - Media codecs, data compressors, etc
- Writing an OS in assembly
  - HelloWorldOS'es, over 9000 (mostly dead)
  - MenuetOS (thanks Ville), Visopsys (hi Andy), a few others
- Writing an OS in assembly in 2026
  - fun, hobby, art, demo, dream, you name it

KolibriOS Introduction

- Website: https://kolibrios.org
- Requirements: i586 CPU, 6 MiB RAM
- Kernel
  - Size: $<$ 100kiB compressed
  - LOC: ~50k
  - Architecture: monolithic?
- Languages
  - Kernel: fasm (a.k.a. flat assembler)
  - Drivers: fasm, C
  - Apps: fasm, nasm, C, C++, C--, Oberon, etc.
- Google Summer of Code: 2014, 2016, 2024,

Virtual Kernel, Rump Kernel, Anykernel, Unikernel, etc

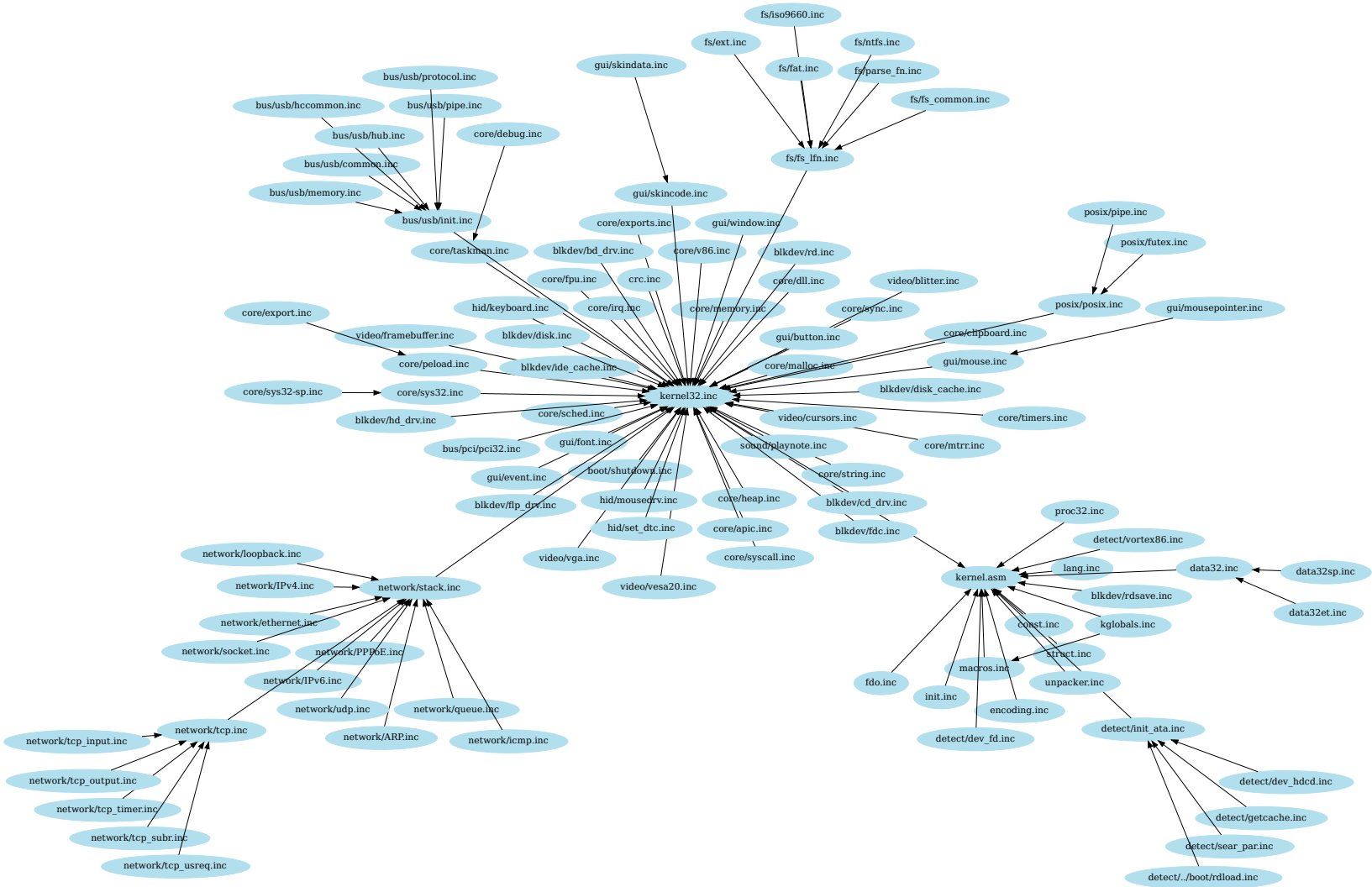
- User-mode Linux
- DragonflyBSD vkernel
- NetBSD rump kernels
- ...
- KolibriOS UMKa

- 'The Anykernel and Rump Kernels',
  by Antti Kantee, FOSDEM'13

Developers, developers, developers. Myths and Truths

- Branch coverage
- Line coverage
- Debugging
- Backtrace
- Automatic testing
- Unit testing
- Anykernel <--- you are here
- ...
- Profit!

Map of Kernel Files



What's in our Anykernel

- Automatic instrumentation, coverage, and unit testing of assembly code
  - File systems, disk cache, partition scan, etc
  - Most of UI and graphics
  - Basic network
  - User input (HID)
  - Injected reproducible 'hardware' errors
  - Multiple threads*
  - In-kernel crypto, hash, (de)compression algorithms
  - String manipulation, datetime conversion, other minor procs
  - WiP: ACPI/AML interpreter

Kernel Dependencies / Host Services

- Memory management: good old *malloc* and *free* from libc
- Block devices: regular files
- Network: TUN/TAP interfaces
- Graphics framebuffer: static aligned array(s)
- Keyboard, mouse: keyboard, mouse (interactive or from a file)
- FS: FUSE

Asm-specific ~~hacks~~ approach

- Calling convention: wrappers
- Static addresses: linker scripts
- Privileged/ring0 instructions: macros
- Real mode code: don't call it
- Hardware access: stubs
- Source code instrumentation: FASM preprocessor
- Binary code analysis: CPU-specific ISA extensions

Coverage: Lines and Branches

Branch tracing:

- Generic LBR (Last Branch Record), MSR_IA32_DEBUGCTL, EFLAGS Trap Flag
- AMD LWP (LightWeight Profiling)
- Intel PT (Processor Trace)
- Intel PEBS (Processor Event Based Sampling)

```
                          :
                    119   :          movzx   eax, [ebx+xfs_inode.di_core.di_format]
                          :          ; switch directory ondisk format and jump to corresponding label
                    119   :          cmp     eax, XFS_DINODE_FMT_LOCAL
          82/37     119   :          jnz     @f
                     37   :          add     ebx, [ebp+XFS.inode_core_size]
                     37   :          stdcall xfs._.readdir_sf, ebx, [_dst]
                     37   :          test    eax, eax
-          0/37      37   :          jnz     .error
                     37   :          jmp     .quit
                          : @@:
                     82   :          cmp     eax, XFS_DINODE_FMT_BTREE
```

umka_shell — an interactive shell (what about fuzzing?)

```
$ umka_shell < 004_#f70_#f70s5_#xfs_stat.t

/> umka_boot

/> disk_add ../img/xfs_v5_files_s05k_b4k_n8k.img hd0 -c 0
/hd0/1: xfs

/> stat70 /hd0/1/
status = 0 success
attr: ----f

/> stat70 /hd0/1/some_file
status = 0 success
attr: -----
size: 65536

/> disk_del hd0
```

Insripation: xfs_db, xfs_io, debugfs, libxfs, xfstests

umka_shell — example commands

```
umka_set_boot_params

send_scancode
set_keyboard_layout
mouse_move

disk_add
disk_del

draw_line
draw_rect

i40 (syscall entry)
```

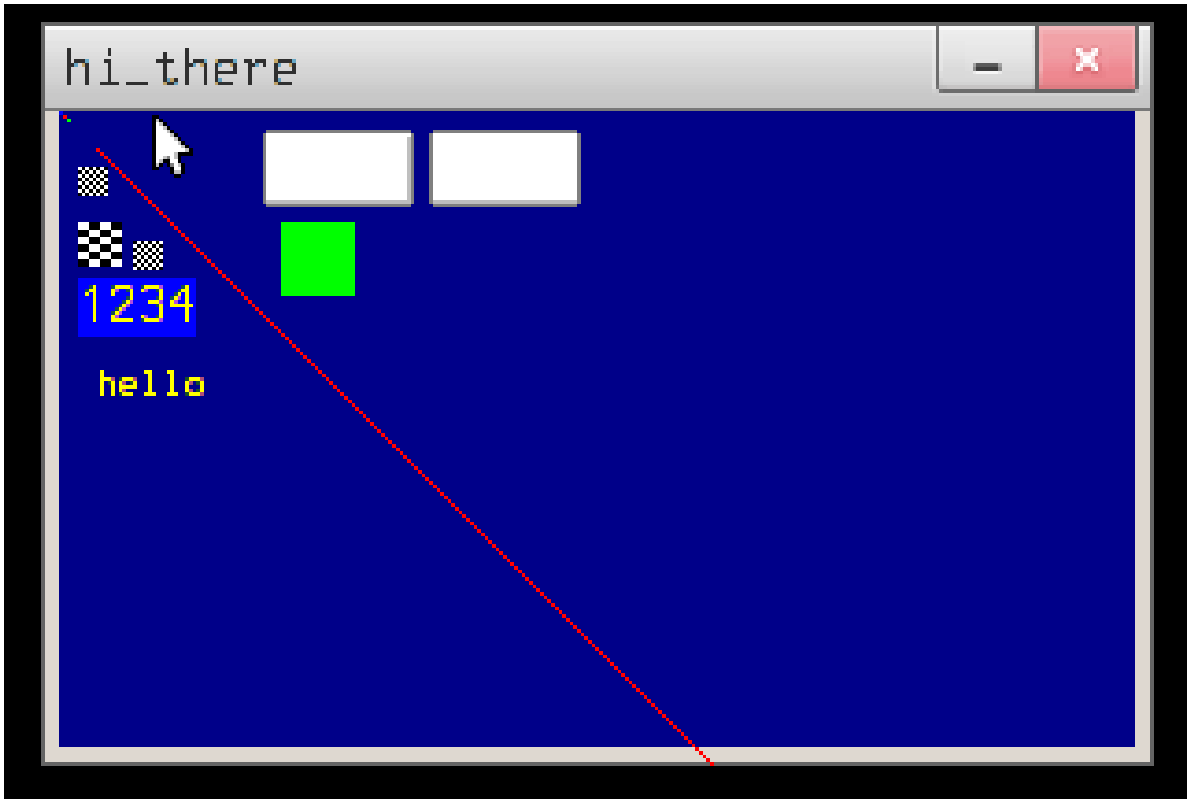
```
dump_dlls
dump_shmem


net_accept
net_bind
net_connect
net_listen
net_ipv4_get_addr
net_ipv4_set_addr


wait_for_idle
wait_for_window
```

umka_fuse — UMKa gets commands via FUSE/libfuse

- FUSE – Filesystem in Userspace
- High-level API, low-level API
- Mount disk images
- Use your favourite shell and tools
- Run xfs-tests / cross-fs tests / generic
- compare/validate against reference FS implementation in Linux(R)

umka_os — Finally, KolibriOS Kernel in Linux® Userspace

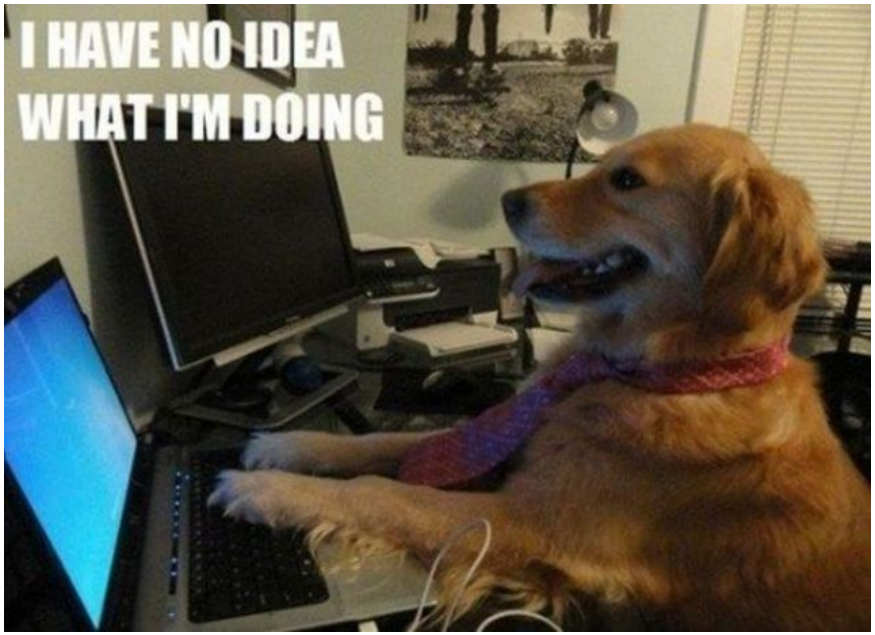
- Task switching via signal handler
- Graphics via SDL
- Input as usual
- Network via TUN/TAP



Future Directions

- Finish ACPI/AML interpreter
- Support new FS feature flags
- Full network stack
- USB via libusb

Conclusion



Audio player?
Sound card driver?
UEFI loader?
AML interpreter?
Unit testing framework?
Code coverage tool?
Anykernel!

Thank you for your attention!

Also thanks to

- Tomasz Grysztar
  for flat assembler
- CleverMouse
  for block/vfs KolibriOS redesign
- Vladimir Solontsov
  for asking me why I develop KolibriOS

Bonus. What about amd64?

- Userspace: mostly fine
- Drivers: not great
- Kernel: not terrible