KERNKONZEPT

THE MICROKERNEL LANDSCAPE IN 2023

Martin Decky

About the Speaker

- **Co-author of the HelenOS microkernel multiserver operating system**
  - Contributing to HelenOS since 2004
- **Operating systems researcher and engineer**
  - Charles University in Prague (2008 – 2017)
    - Ph.D. in 2015
  - Huawei Technologies (2017 – 2021)
    - Co-founder of the Dresden Research Center
  - Kernkonzept (since 2021)

Microkernel-based Operating Systems

- **Fundamental approach to achieve operating system reliability and dependability**
  - Via proper **software architecture** following clear **design principles**
    - Separation of concerns
    - Split of mechanism and policy
    - Least privilege
  - Results in design that is modular, customizable and verifiable
    - Minimality (i.e. the “*micro*” part) is a consequence, not an a priori goal
    - Perhaps “non-monolithic kernel” would be a more fitting (but less catchy) name
  - Architecture and design principles affect not just the kernel, but also the user space
    - Hence: “microkernel multiserver OS with fine-grained components”

History

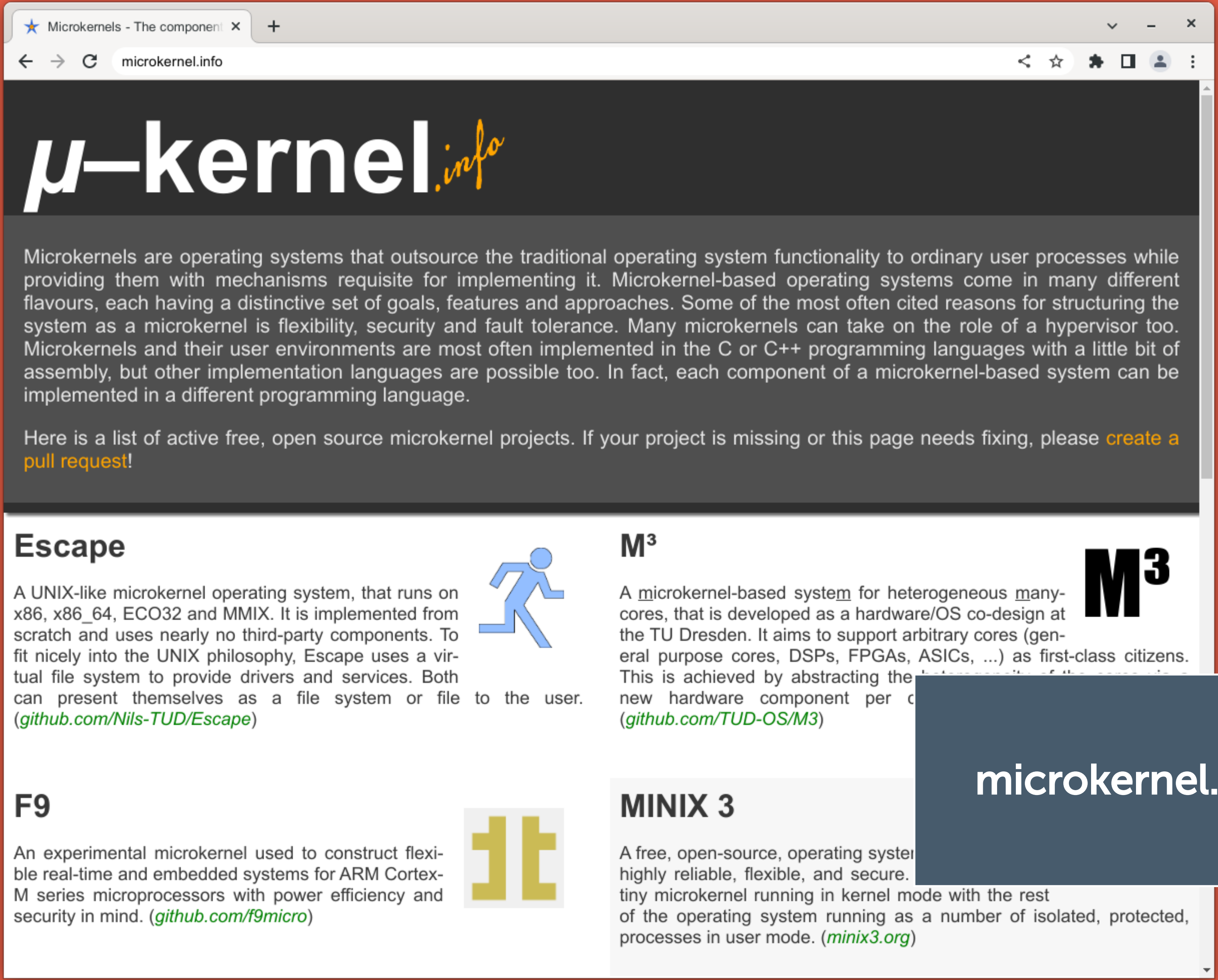
- **RC 4000 Multiprogramming System**
  - Per Brinch Hansen, Regnecentralen, 1969
    - Separation of mechanism and policy, isolated concurrently running processes, message passing
- **HYDRA**
  - William Wulf, Carnegie Mellon University, 1971
    - Capabilities, object orientation
- **EUMEL / L2**
  - Jochen Liedtke, University of Bielefeld, 1979
    - Proto-microkernel based on bitcode virtual machines

History

- **QNX**
  - Gordon Bell, Dan Dodge, 1982
    - Earliest commercially successful microkernel multiserver OS
- **CMU Mach**
  - Richard Rashid, Avie Tevanian, Carnegie Mellon University, 1985
    - Still physically present in the code base of macOS, iOS, Hurd, etc.
    - Highly infuential (e.g. on Windows NT) despite its well-publicized shortcomings

History

- **L4**
  - Jochen Liedtke, German National Research Center of Information Technology, 1993
    - Reflection of the design and performance shortcomings of CMU Mach
      - Successfully demonstrating the viability of the approach
    - Original implementation in non-portable x86 assembly
    - Started a large family of very loosely related (and more portable) microkernels
      - Contrary to popular belief, many state-of-the-art microkernels have very little to do with the original L4 design and implementation (sometimes even despite having “L4” in their name)



μ–kernel.info

Microkernels are operating systems that outsource the traditional operating system functionality to ordinary user processes while providing them with mechanisms requisite for implementing it. Microkernel-based operating systems come in many different flavours, each having a distinctive set of goals, features and approaches. Some of the most often cited reasons for structuring the system as a microkernel is flexibility, security and fault tolerance. Many microkernels can take on the role of a hypervisor too. Microkernels and their user environments are most often implemented in the C or C++ programming languages with a little bit of assembly, but other implementation languages are possible too. In fact, each component of a microkernel-based system can be implemented in a different programming language.

Here is a list of active free, open source microkernel projects. If your project is missing or this page needs fixing, please create a pull request!

Escape

A UNIX-like microkernel operating system, that runs on x86, x86_64, ECO32 and MMIX. It is implemented from scratch and uses nearly no third-party components. To fit nicely into the UNIX philosophy, Escape uses a virtual file system to provide drivers and services. Both can present themselves as a file system or file to the user. (*github.com/Nils-TUD/Escape*)

M^3

A microkernel-based system for heterogeneous many-cores, that is developed as a hardware/OS co-design at the TU Dresden. It aims to support arbitrary cores (general purpose cores, DSPs, FPGAs, ASICs, ...) as first-class citizens. This is achieved by abstracting the

new hardware component per c

(*github.com/TUD-OS/M3*)

F9

An experimental microkernel used to construct flexible real-time and embedded systems for ARM Cortex-M series microprocessors with power efficiency and security in mind. (*github.com/f9micro*)

MINIX 3

A free, open-source, operating syste

highly reliable, flexible, and secure.

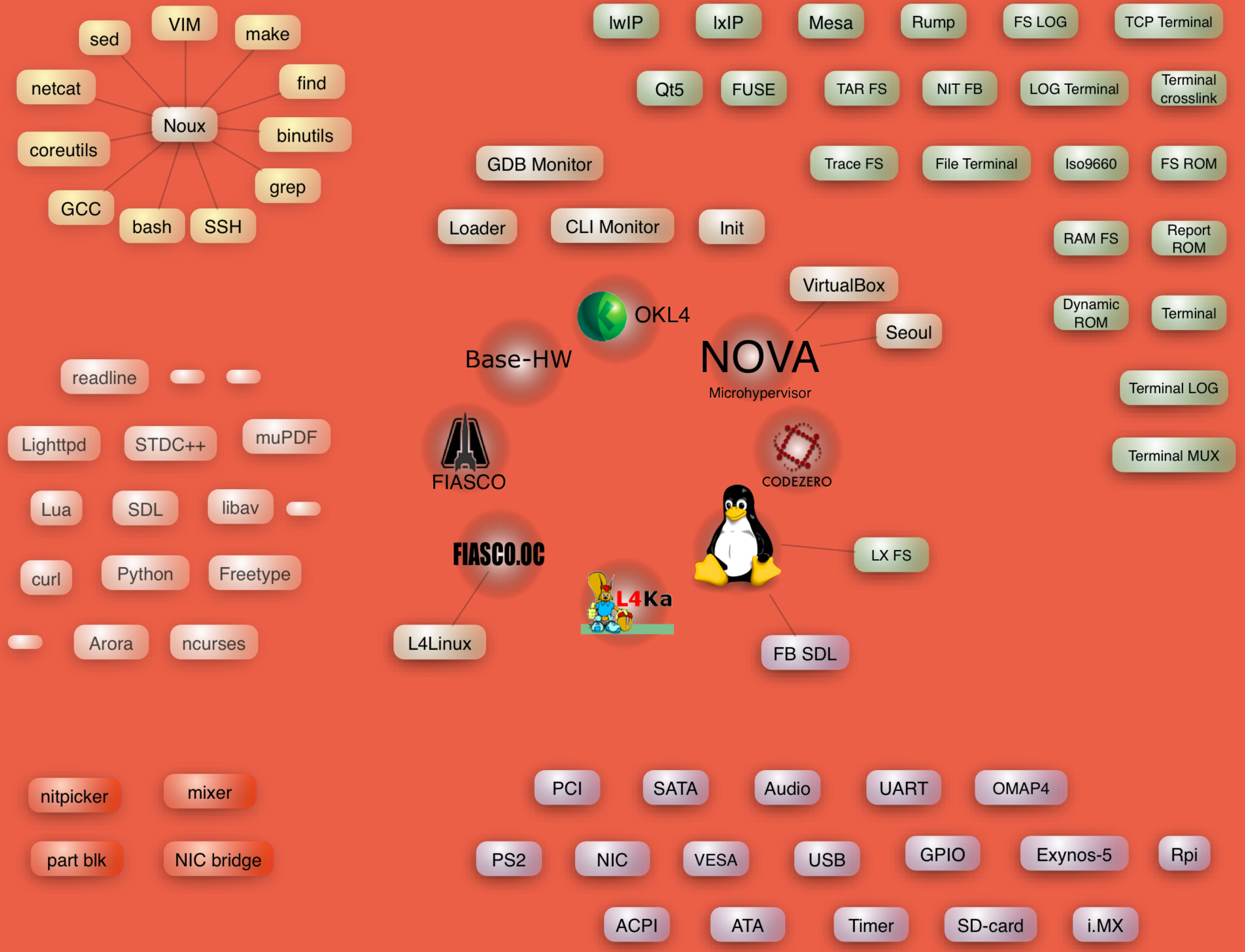
tiny microkernel running in kernel mode with the rest of the operating system running as a number of isolated, protected, processes in user mode. (*minix3.org*)

microkernel.info

Genode by Genode Labs

- **Operating systems construction kit**
  - Arguably the most versatile general-purpose desktop-oriented environment empowering microkernels
  - Used successfully in production (references are not public)
  - Supports multiple kernels
    - NOVA, seL4, Fiasco.OC, OKL4, L4Ka::Pistachio, L4/Fiasco, base-hw, Linux
  - Strong focus on resource management and accounting
- **Sculpt OS**
  - Prebuild distro of Genode

Genode
at a glance



sed
VIM
make
netcat
find
Noux
binutils
coreutils
grep
GCC
bash
SSH

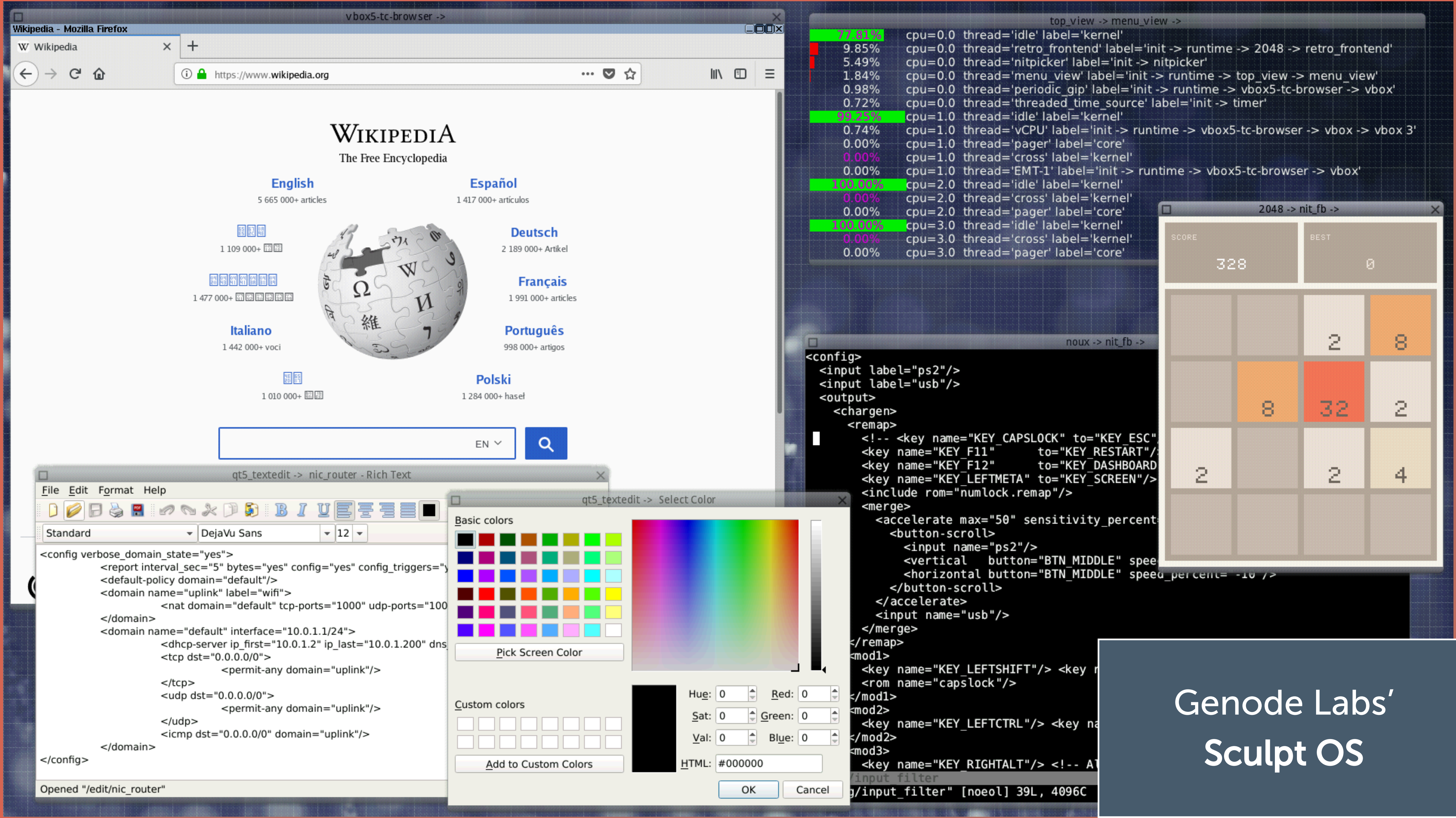
lwIP
lxIP
Mesa
Rump
FS LOG
TCP Terminal
Qt5
FUSE
TAR FS
NIT FB
LOG Terminal
Terminal crosslink
GDB Monitor
Trace FS
File Terminal
Iso9660
FS ROM
Loader
CLI Monitor
Init
RAM FS
Report ROM
VirtualBox
OKL4
Dynamic ROM
Terminal
Seoul
Base-HW
NOVA
Microhypervisor
Terminal LOG
FIASCO
CODEZERO
Terminal MUX
FIASCO.OC
L4Ka
LX FS
L4Linux
FB SDL

readline
Lighttpd
STDC++
muPDF
Lua
SDL
libav
curl
Python
Freetype
Arora
ncurses

nitpicker
mixer
part blk
NIC bridge

PCI
SATA
Audio
UART
OMAP4
PS2
NIC
VESA
USB
GPIO
Exynos-5
Rpi
ACPI
ATA
Timer
SD-card
i.MX

Source: Feske N.: *Introducing kernel-agnostic Genode executables*, Genode Labs, FOSDEM 2017, https://fosdem.org/2017/schedule/event/microkernel_kernel_agnostic_genode_executables/



Hue: 0 Red: 0
Sat: 0 Green: 0
Val: 0 Blue: 0
HTML: #000000

```
 77.81%   cpu=0.0  thread='idle' label='kernel'
  9.85%   cpu=0.0  thread='retro_frontend' label='init -> runtime -> 2048 -> retro_frontend'
  5.49%   cpu=0.0  thread='nitpicker' label='init -> nitpicker'
  1.84%   cpu=0.0  thread='menu_view' label='init -> runtime -> top_view -> menu_view'
  0.98%   cpu=0.0  thread='periodic_gip' label='init -> runtime -> vbox5-tc-browser -> vbox'
  0.72%   cpu=0.0  thread='threaded_time_source' label='init -> timer'
 99.25%   cpu=1.0  thread='idle' label='kernel'
  0.74%   cpu=1.0  thread='vCPU' label='init -> runtime -> vbox5-tc-browser -> vbox -> vbox 3'
  0.00%   cpu=1.0  thread='pager' label='core'
  0.00%   cpu=1.0  thread='cross' label='kernel'
  0.00%   cpu=1.0  thread='EMT-1' label='init -> runtime -> vbox5-tc-browser -> vbox'
100.00%   cpu=2.0  thread='idle' label='kernel'
  0.00%   cpu=2.0  thread='cross' label='kernel'
  0.00%   cpu=2.0  thread='pager' label='core'
100.00%   cpu=3.0  thread='idle' label='kernel'
  0.00%   cpu=3.0  thread='cross' label='kernel'
  0.00%   cpu=3.0  thread='pager' label='core'
```

```
<config verbose_domain_state="yes">
		<report interval_sec="5" bytes="yes" config="yes" config_triggers="y
		<default-policy domain="default"/>
		<domain name="uplink" label="wifi">
				<nat domain="default" tcp-ports="1000" udp-ports="100
		</domain>
		<domain name="default" interface="10.0.1.1/24">
				<dhcp-server ip_first="10.0.1.2" ip_last="10.0.1.200" dns
				<tcp dst="0.0.0.0/0">
						<permit-any domain="uplink"/>
				</tcp>
				<udp dst="0.0.0.0/0">
						<permit-any domain="uplink"/>
				</udp>
				<icmp dst="0.0.0.0/0" domain="uplink"/>
		</domain>
</config>
```

Opened "/edit/nic_router"

```
<config>
  <input label="ps2"/>
  <input label="usb"/>
  <output>
    <chargen>
      <remap>
        <!-- <key name="KEY_CAPSLOCK" to="KEY_ESC"
        <key name="KEY_F11"       to="KEY_RESTART"/
        <key name="KEY_F12"       to="KEY_DASHBOARD
        <key name="KEY_LEFTMETA" to="KEY_SCREEN"/>
        <include rom="numlock.remap"/>
        <merge>
          <accelerate max="50" sensitivity_percent
            <button-scroll>
              <input name="ps2"/>
              <vertical   button="BTN_MIDDLE" spee
              <horizontal button="BTN_MIDDLE" speed_percent="-10"/>
            </button-scroll>
          </accelerate>
          <input name="usb"/>
        </merge>
      </remap>
      <mod1>
        <key name="KEY_LEFTSHIFT"/> <key r
        <rom name="capslock"/>
      </mod1>
      <mod2>
        <key name="KEY_LEFTCTRL"/> <key na
      </mod2>
      <mod3>
        <key name="KEY_RIGHTALT"/> <!-- A
/input_filter
g/input_filter" [noeol] 39L, 4096C
```

Genode Labs’ Sculpt OS

Genode by Genode Labs

- **base-hw as a bespoke microkernel**
  - Nice integration, but does not have complete feature parity with some other kernels (e.g. with respect to hardware virtualization)
- **Somewhat steep learning curve**
  - Sculpt OS is a huge improvement, but still be prepared to read some documentation
- **https://genode.org**
- **https://genode-labs.com**

L4Re by Kernkonzept

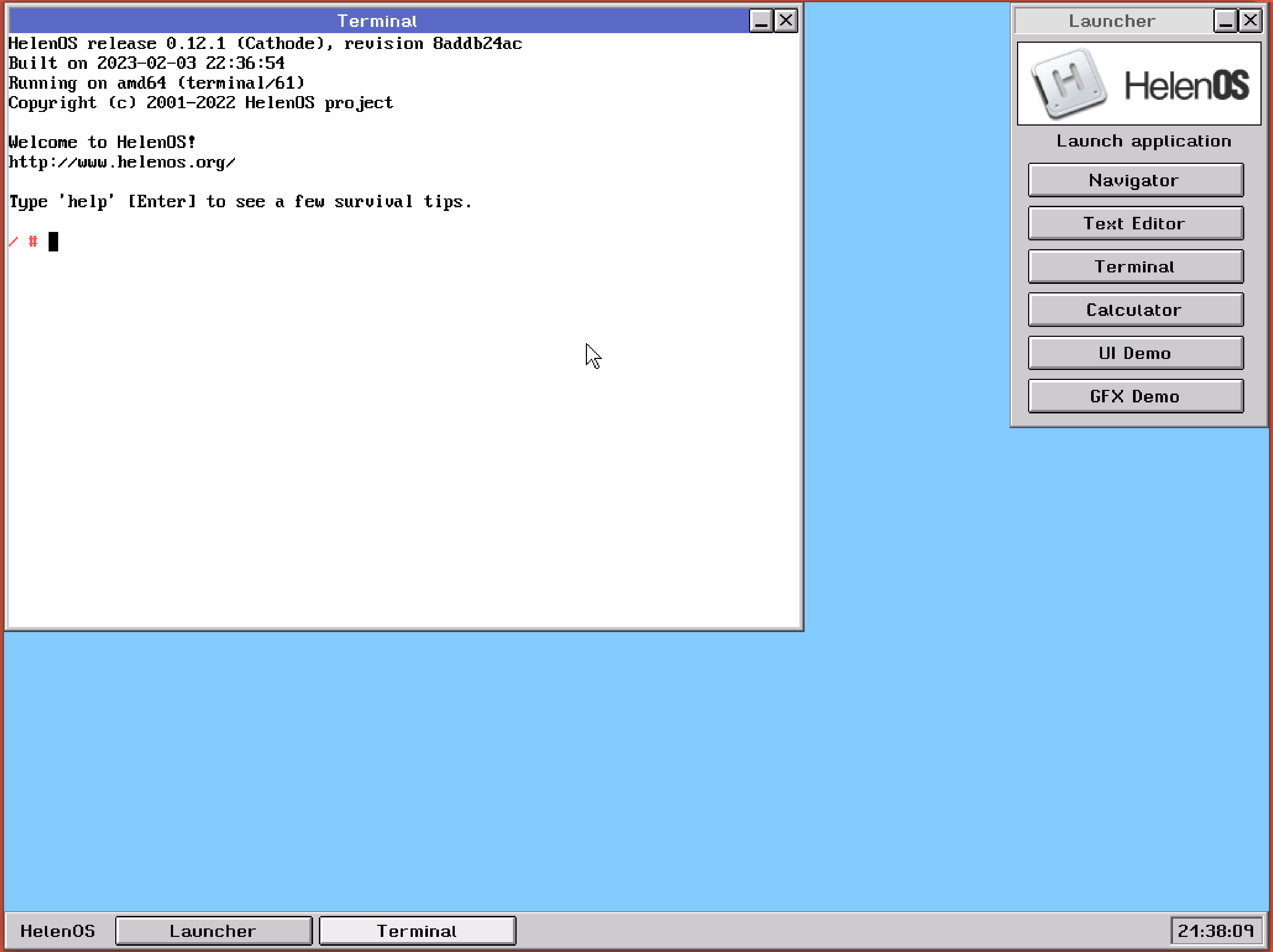
- **Production-grade microkernel-based environment**
  - Uses the L4Re Microkernel (a.k.a. Fiasco.OC)
  - Strong focus on virtualization
  - Targets safety (ISO 26262) and security (Common Criteria) certification
  - If you buy a new car from a German vendor, there is a high chance it will run code derived from L4Re in its software stack

L4Re by Kernkonzept

- **It is not the most verbosely-commented code base**
- **Somewhat steep learning curve**
  - Try building/downloading some example configurations (e.g. `l4linux-mag`)
- **https://l4re.org**
- **https://www.kernkonzept.com**

HelenOS

- **Integrated, general-purpose and desktop-oriented microkernel-based OS**
  - Arguably an ideal starting point with the lowest entry barrier
    - Portable, self-contained, well-structured, well-commented source code with no nasty hacks and surprises
    - Default configuration builds a ready-to-use OS distro
  - Uses only native OS components (no ported “franken-components”)

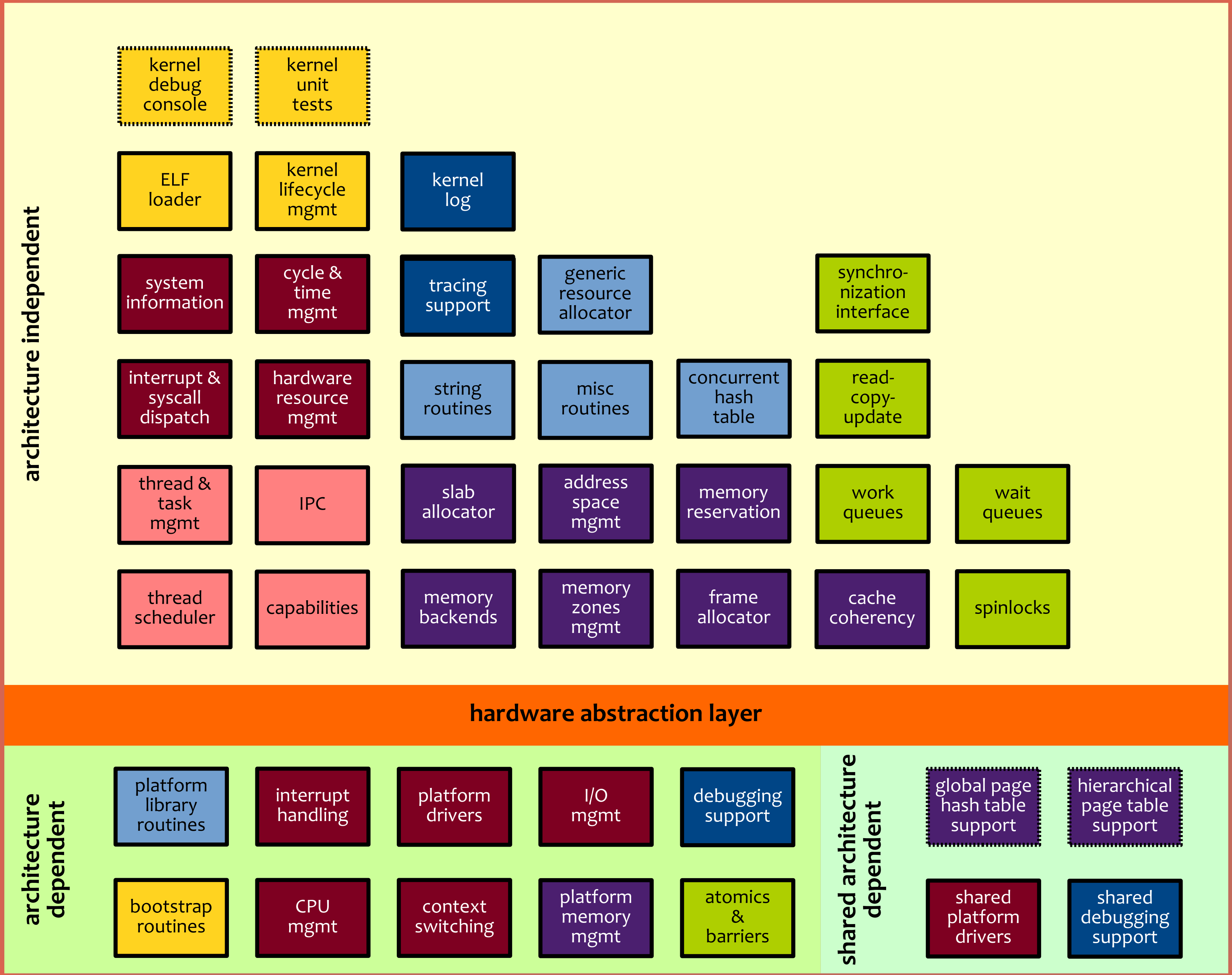


```
HelenOS release 0.12.1 (Cathode), revision 8addb24ac
Built on 2023-02-03 22:36:54
Running on amd64 (terminal/61)
Copyright (c) 2001-2022 HelenOS project

Welcome to HelenOS!
http://www.helenos.org/

Type 'help' [Enter] to see a few survival tips.

/ # 
```

HelenOS

- **Several unique features**
  - Support for 8+ CPU architectures
    - IA-32, AMD64 (x86-64), ARMv7, ARMv8, IA-64, MIPS, PowerPC, SPARCv9, RISC-V (work-in-progress)
  - Highly scalable asynchronous IPC using shared memory
  - Interrupt controller drivers in user space
  - Component-based TCP/IP networking stack (including IPv6 and Wi-Fi support)
  - USB 3.0 support
  - Sound stack

HelenOS
kernel architecture



architecture independent

kernel debug console | kernel unit tests

ELF loader | kernel lifecycle mgmt | kernel log

system information | cycle & time mgmt | tracing support | generic resource allocator | synchronization interface

interrupt & syscall dispatch | hardware resource mgmt | string routines | misc routines | concurrent hash table | read-copy-update

thread & task mgmt | IPC | slab allocator | address space mgmt | memory reservation | work queues | wait queues

thread scheduler | capabilities | memory backends | memory zones mgmt | frame allocator | cache coherency | spinlocks

hardware abstraction layer

architecture dependent

platform library routines | interrupt handling | platform drivers | I/O mgmt | debugging support

bootstrap routines | CPU mgmt | context switching | platform memory mgmt | atomics & barriers

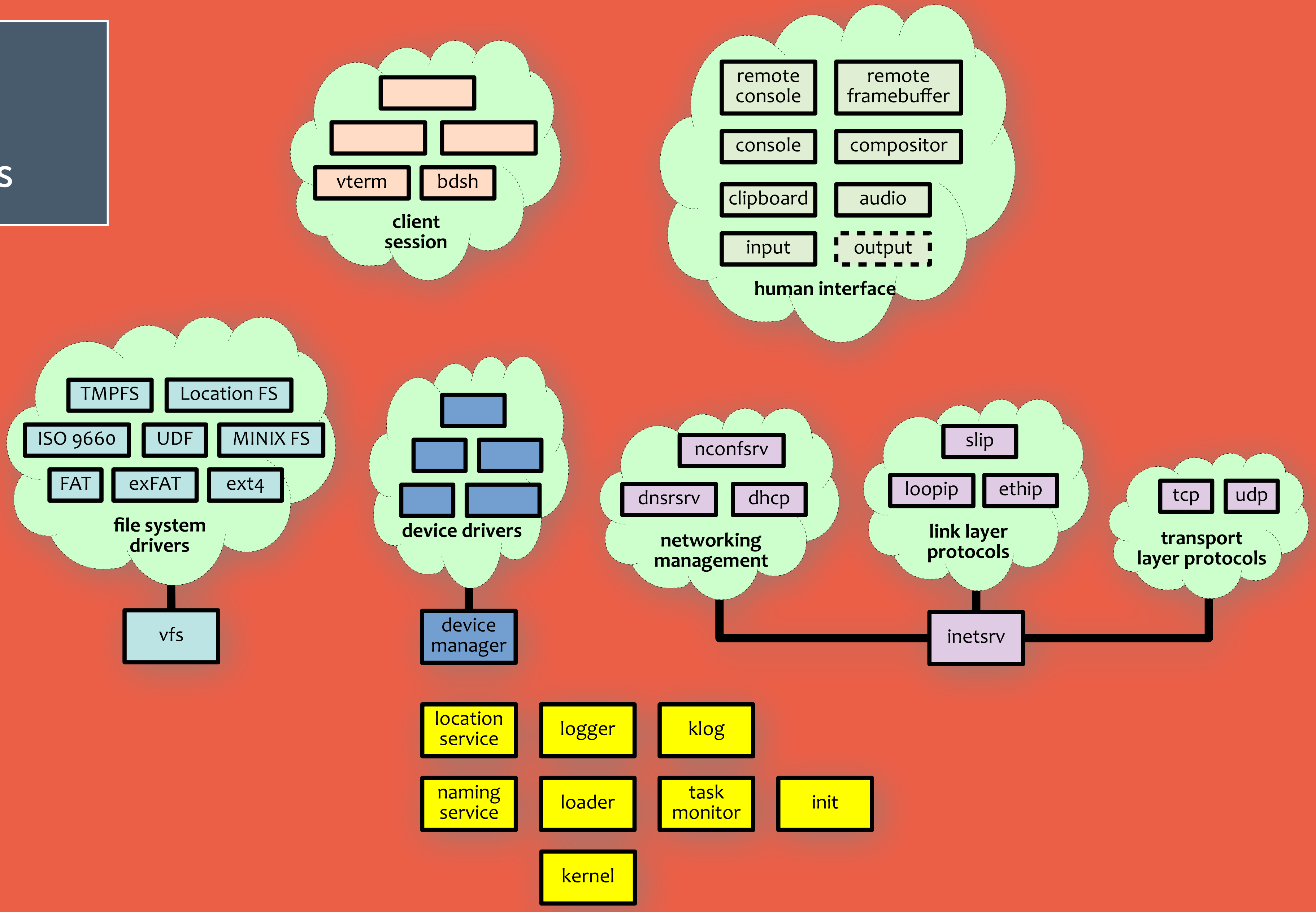
shared architecture dependent

global page hash table support | hierarchical page table support

shared platform drivers | shared debugging support

HelenOS

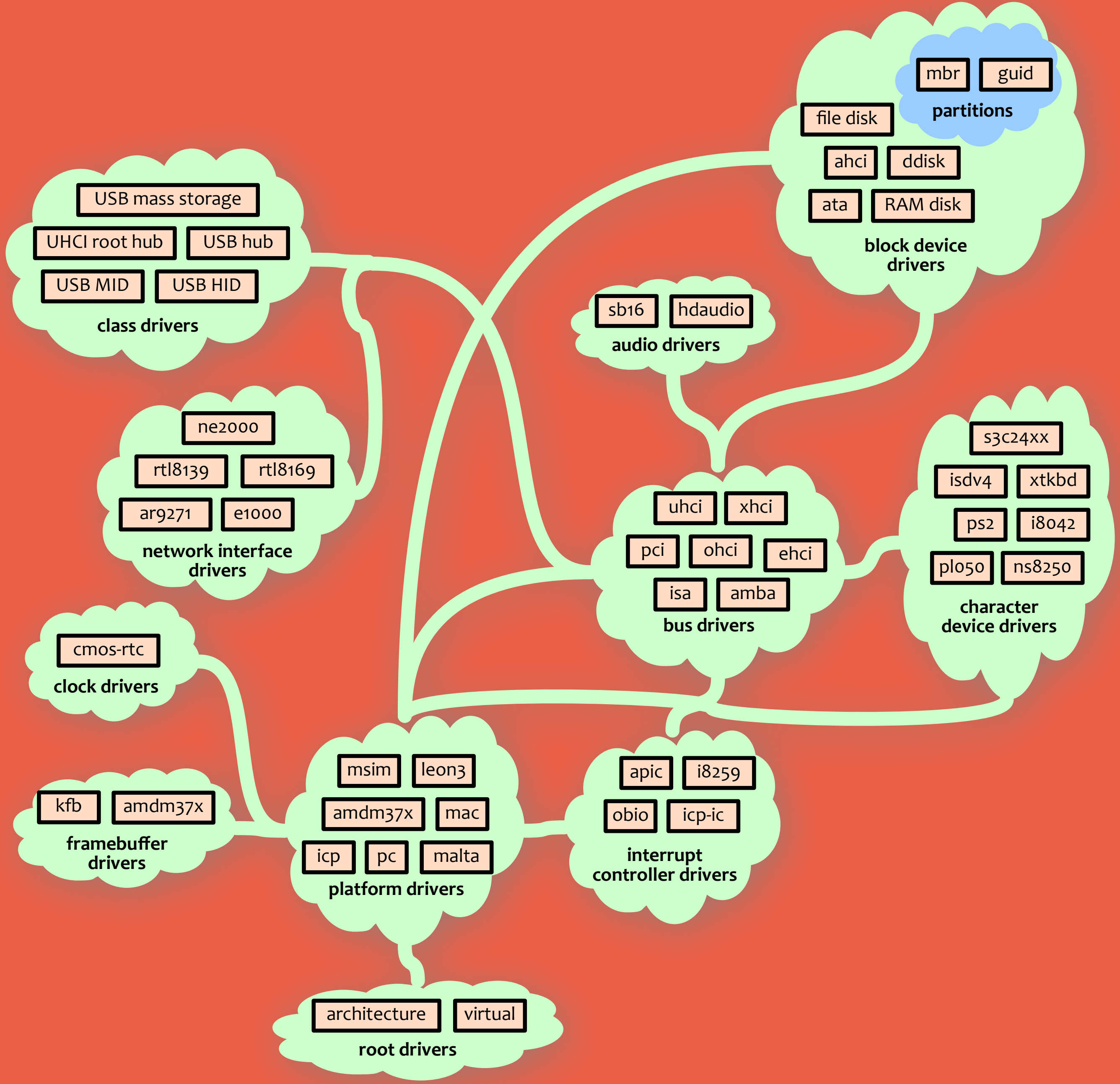
user space components



KERNKONZEPT

HelenOS

user space device drivers



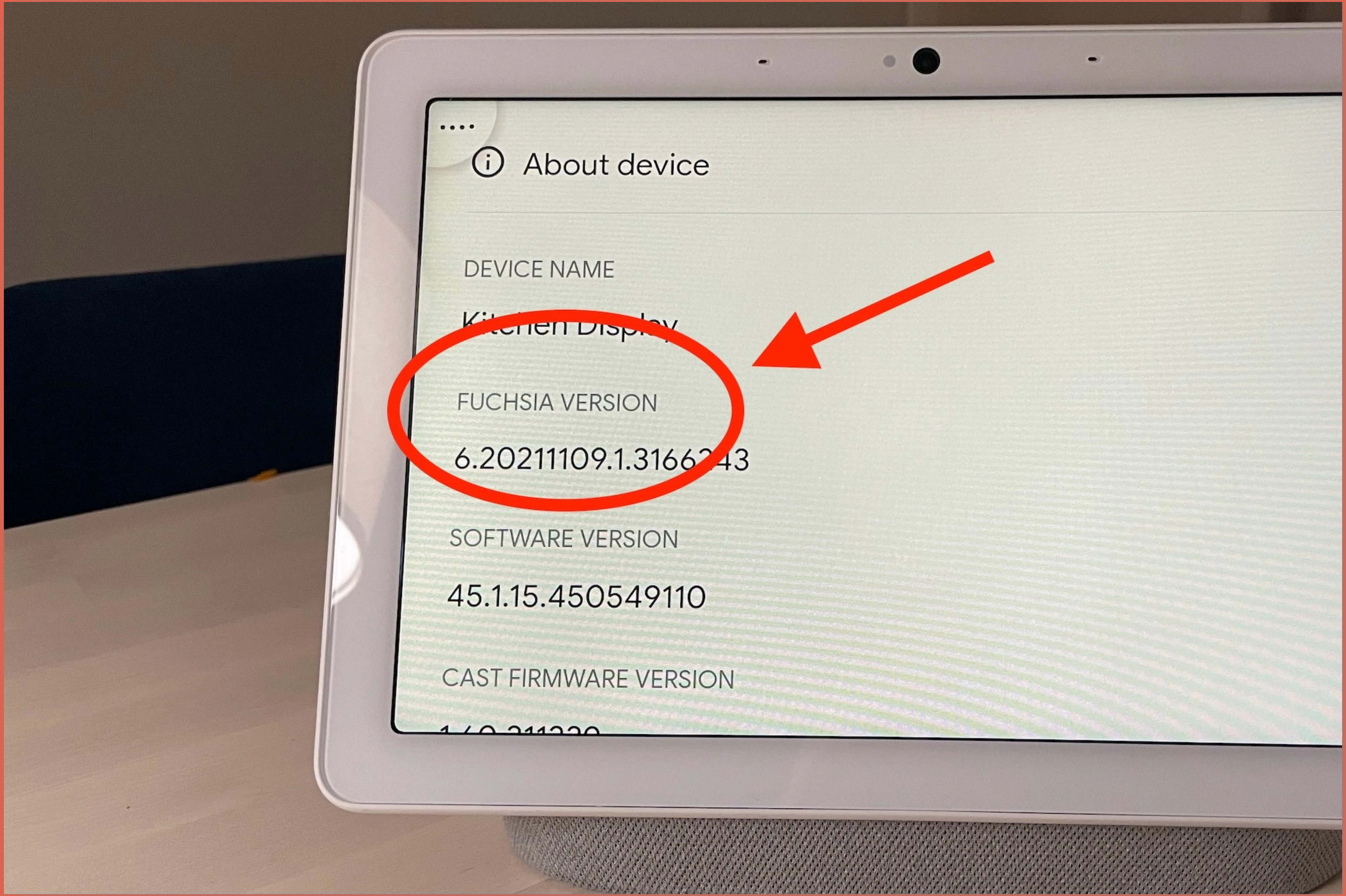
HelenOS

- **Currently purely community-driven effort**
  - Semi-regular releases, but overall development velocity below average
  - Support for newer hardware features missing (e.g. hardware virtualization)
- **http://www.helenos.org**

Fuchsia

Fuchsia by Google

- **Microkernel-based OS focusing on the Internet of Things**
  - Capability-based, message-passing Zircon microkernel
    - Authors deliberately understate the microkernel nature to avoid the “bad press” of the term
  - Targets seamless maintenance, remote management and upgrade of a fleet of devices
  - Agnostic to the implementation language of the core components
  - Currently shipping with Google Nest Hub



About device

DEVICE NAME

Kitchen Display

FUCHSIA VERSION

6.20211109.1.3166[illegible]43

SOFTWARE VERSION

45.1.15.450549110

CAST FIRMWARE VERSION

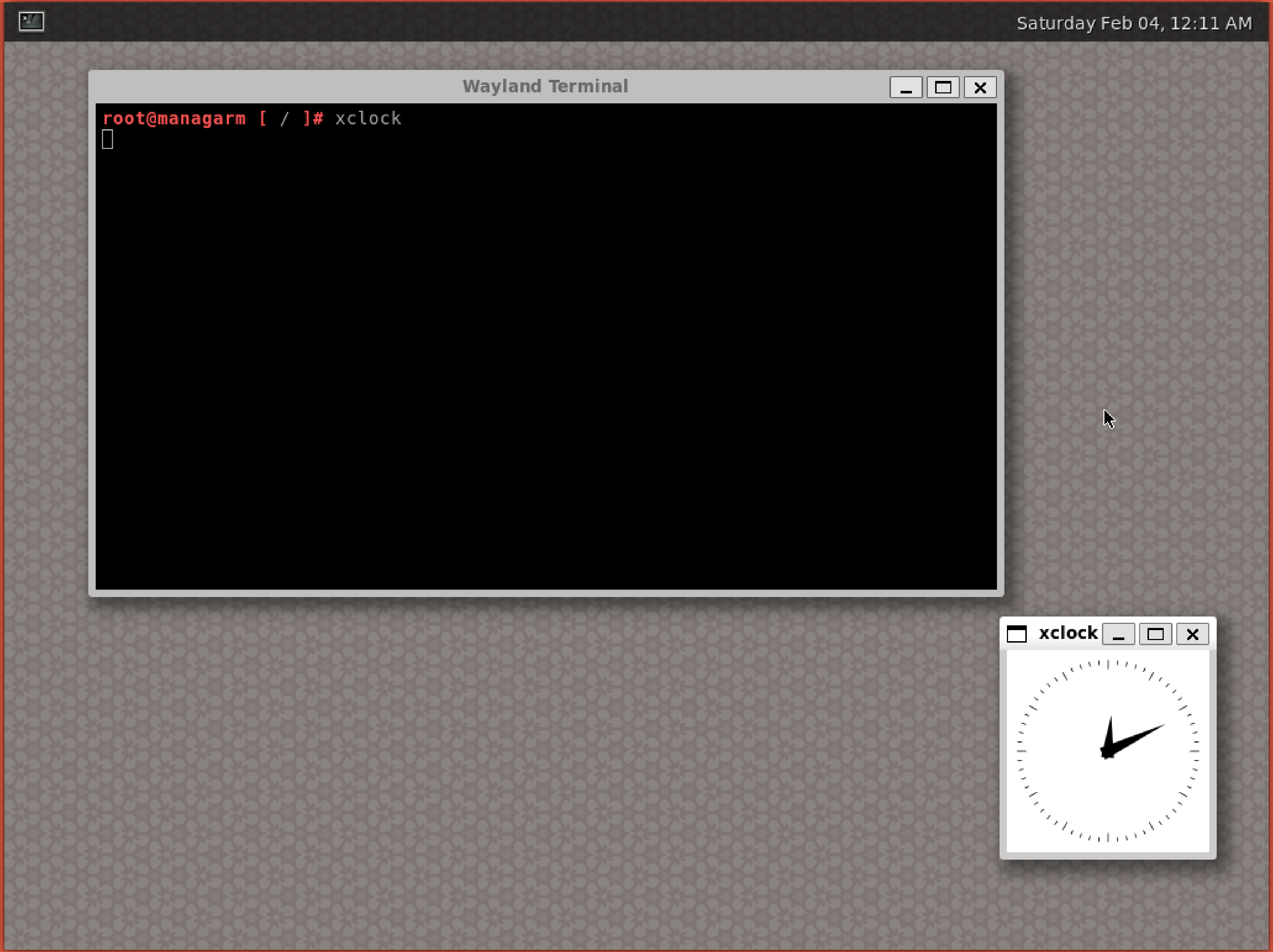
Photo source: Patterson B.: *How to check if your Google Nest Hub display is running Fuchsia*, 2022,
https://www.techhive.com/article/579622/how-to-check-if-your-google-nest-hub-is-running-fuchsia.html

Fuchsia by Google

- **Somewhat steep learning curve**
  - Non-trivial toolchain and build environment setup
  - Custom emulator
  - Several C/C++ bindings for the FIDL
- **Uses only native OS components (no ported “franken-components”)**
- **https://fuchsia.dev**

Managarm

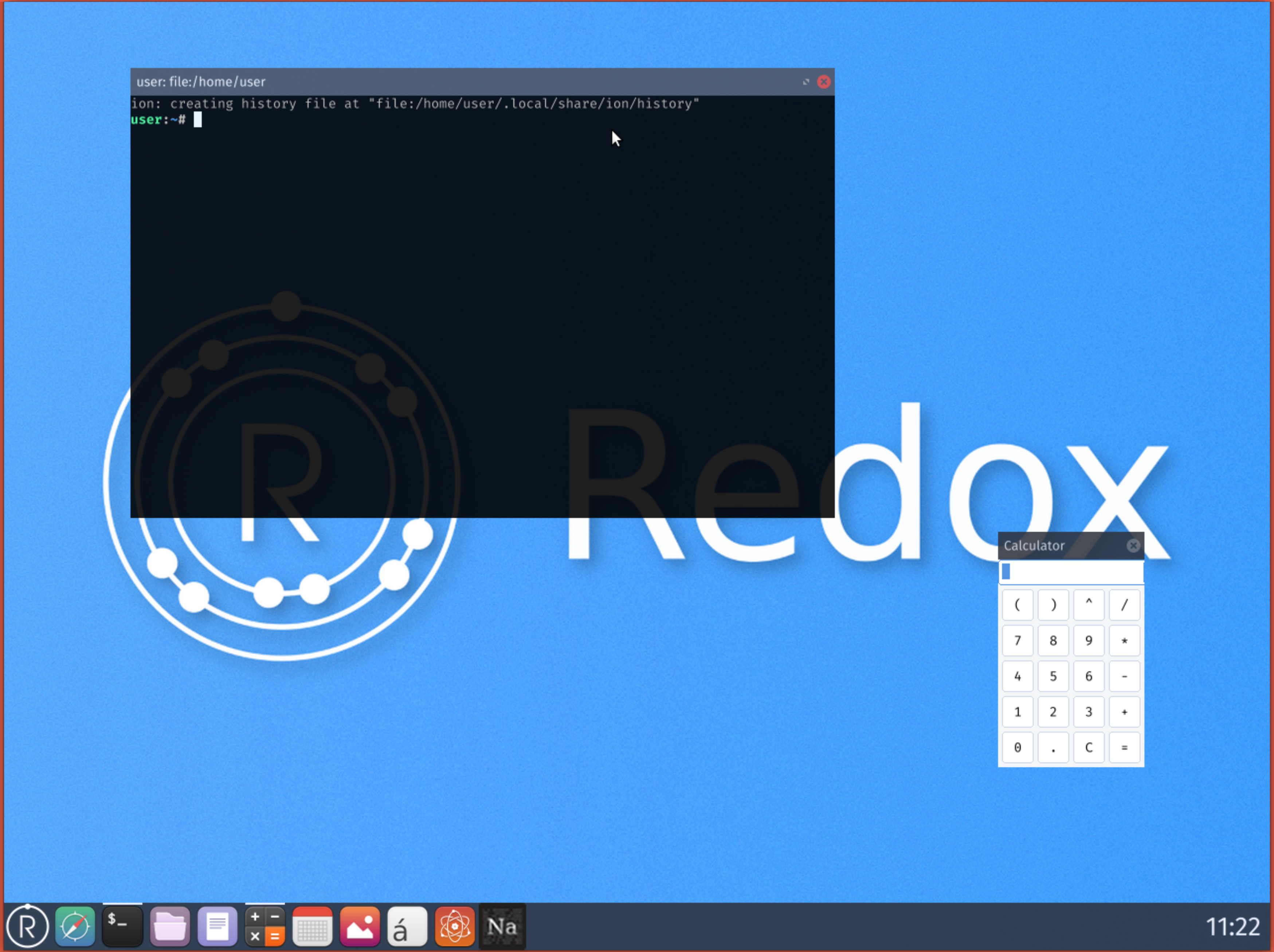
- **General-purpose, desktop-oriented microkernel-based OS**
  - Fully asynchronous kernel design
  - Various pragmatic kernel performance features (e.g. page cache)
  - Strong focus on the POSIX compatibility layer and Linux compatibility (supporting Weston, coreutils, Bash, GTK+, Qt, etc.)
  - Supports AMD64 (x86-64), ARMv8 and initially RISC-V
    - Some accelerated GPU drivers
- **https://managarm.org**



```
root@managarm [ / ]# xclock
```

Redox

- **Unix-like microkernel-based OS written in Rust**
  - Also core user space components in Rust (e.g. relibc)
  - Targets general-purpose and desktop deployment
  - Mostly focuses on AMD64 (x86-64), but there is also ARMv8 support
  - Strong focus on the POSIX compatibility layer (supporting coreutils, DOSBox, FFMPEG, SDL, etc.)
- **https://www.redox-os.org**



```
ion: creating history file at "file:/home/user/.local/share/ion/history"
user:~#
```

HarmonyOS

HongMeng OS by Huawei

- **Most "progressive" member of the HarmonyOS brand**
  - Overloaded marketing term that covers different OS architectures (including a Linux-based and a LiteOS-based)
- **Custom microkernel-based implementation**
  - Initial design inspired by the state-of-the-art, but there have been several redesigns
    - Fundamental capability-based memory management in user space
      - Inspired by seL4, but modified to be more practical
  - Targets safety (ISO 26262) and security (Common Criteria) certification
  - Shipped in millions of smartphones as the Trusted Execution Environment (TEE)

DUCK by Huawei

- **R&D effort primarily driven by the Dresden Research Center**
  - Clean-slate design and implementation
  - Capability model finer than in existing microkernels
  - State-of-the-art best practices in software engineering to achieve the highest code quality and maintainability
    - Targets full MISRA C compliance of the kernel
  - Targets high level of safety (ISO 26262 ASIL-D) and security (Common Criteria EAL5+) certification, potentially formal verification
  - Support for hard real time workloads
  - Core user space components in Rust

Other Notable Microkernel-based Projects

- **GNU/Hurd**
  - Intended microkernel replacement of Linux for GNU
    - Based on GNU Mach (derived from CMU Mach)
  - Still in active development, semi-regular Debian GNU/Hurd releases (supporting about 70 % of Debian packages)
  - Supports only IA-32
  - https://www.gnu.org/software/hurd
- **Ares**
  - Helios microkernel inspired by seL4, implemented in Hare
  - https://ares-os.org

Other Notable Microkernel-based Projects

- **Composite**
  - Focus on low latency, predictability, component composition
  - Lock-less kernel, user space scheduling, thread-migration IPC
  - https://composite.seas.gwu.edu
- **UX/RT**
  - QNX-inspired OS on top of the seL4 microkernel
  - Still in early stages of development
  - https://gitlab.com/uxrt

Other Notable Microkernel-based Projects

- **QNX by BlackBerry**
  - Still in active use, but little public information
  - https://blackberry.qnx.com
- **PikeOS by SYSGO**
  - Real-time hypervisor targeting automotive
  - Common Criteria EAL5+ certification
  - https://www.sysgo.com/pikeos
- **Many real-time, embedded and "retro" kernels could be technically described as microkernels**
  - Although the classification is somewhat blurry and questionable
  - Some examples: INTEGRITY-178B (Green Hills Software), Zephyr (up to 1.5), Exec & AROS (AmigaOS), MorphOS, Horizon (Nintendo)

Standalone Microkernels

- **NOVA Microhypervisor**
  - http://hypervisor.org
  - BedRock HyperVisor (BHV)
    - https://bedrocksystems.com
- **Hedron Hypervisor**
  - Fork of NOVA
  - Developed by Cyberus Technology as Secure Virtualization Platform
    - https://github.com/cyberus-technology/hedron
    - https://www.cyberus-technology.de/products/svp/

Standalone Microkernels

- **seL4**
  - https://sel4.systems
  - Google CantripOS (a.k.a. KataOS)
    - Extending the CAmkES framework for Rust
    - Targets verifiably secure embedded devices
    - https://github.com/AmbiML/sparrow-manifest
- **Muen Separation Kernel**
  - https://muen.sk

Microkernel-based Projects in Limbo

- Escape (https://github.com/Nils-TUD/Escape)
- M^3 (https://github.com/TUD-OS/M3)
- MINIX 3 (http://minix3.org)
- Robigalia (https://robigalia.org)
- RedLeaf (https://github.com/mars-research/redleaf)
- Barrelfish (https://barrelfish.org)

KERNKONZEPT

THANK YOU

martin.decky@kernkonzept.com
martin@decky.cz