HelenOS in the Year of the Pig

http://www.helenos.org

Jakub Jermář
jakub@jermar.eu
HelenOS in a nutshell

- Portable microkernel-based multiserver OS
- Designed and implemented from scratch
- Not a clone of any existing OS / API
- Virtually no third-party code
- Fine-grained userspace components
Since last FOSDEM...

May
HelenOS 0.9.1 Armonia
L4Re::Treat<2019>

Jun / July
SUSE Hack Week 19: HelenOS of the <future>

August
HelenOS quasi-Camp 2019
Master and beyond

- C++ <future> header (master)
- Switch to the Meson build system (master)
- IPC v2 (WiP, branch)
- HelenOS as anykernel (WiP, branch)
- Display server (WiP, branch)
HelenOS IPC Recap

```
int ipc_answer_fast(int callid, int retval, void* arg1, void* arg2, void* arg3, void* arg4);
int ipc_forward_fast(int callid, int method, int arg1, int arg2, ...);
int ipc_wait_for_call_timeout(void);
int ipc_call_async_fast(int method, int arg1, int arg2, int arg3, ...);
Answerbox
Phone
Task

Answerbox
Phone
Task

Answerbox
Phone
Task
```

...
HelenOS IPC Recap

Call
  non-reusable, fixed-size, kernel allocated/managed

Phone
  (unnecessary?) level of indirection between client and server

Answerbox
  asynchronous, waitable
  (still) one-per task

Where should the answer go with >1 answerboxes?
How does a server wait for multiple requests?
How does a client wait for multiple answers?
 IPC v2 (WiP)

**IPC buffer**
preallocated, reusable, arbitrary size, waitable

**IPC endpoint**
asynchronous, waitable

**Caplists**
container for other kernel objects
similar to Mach port sets / epoll
waiting for multiple requests / answers
Simple Roundtrip

\[ b = \text{buf_alloc}(); \]

\[ \text{buf}_\text{send}(b, \text{ep}); \]

\[ \text{buf}_\text{wait}(b); \]

\[ \text{buf}_\text{free}(b); \]

\[ ep = \text{ep}_\text{create}(); \]

\[ \text{buf}_\text{receive}(ep, \text{tmp}_b); \]

\[ \text{buf}_\text{finish}(\text{tmp}_b); \]

\[ \text{ep}_\text{destroy}(ep); \]
Multiple buffers and endpoints

\[
\begin{align*}
\textit{bcl} &= \text{caplist\_create}(); \\
\textit{b1} &= \text{buf\_alloc}(); \\
\textit{b2} &= \text{buf\_alloc}(); \\
\textit{ecl} &= \text{caplist\_create}(); \\
\textit{ep1} &= \text{ep\_create}() \text{ caplist}\_create(); \\
\textit{ep2} &= \text{ep\_create}(); \\
\textit{bcl} &= \text{caplist\_destroy}(); \\
\textit{ep1} &= \text{ep\_destroy}(); \\
\textit{ep2} &= \text{ep\_destroy}(); \\
\textit{bcl} &= \text{caplist\_destroy}();
\end{align*}
\]
HelenOS as Anykernel (WiP)

Martin Děcký @mdecky · Aug 17, 2019
The first prototype of HelenOS anykernel runs on my machine. Compared to the NetBSD rumpkernel approach, our foundation is the fine-grained microkernel multiserver architecture. The goal is to have the maximum flexibility of the OS architecture at deployment time.

💬 2  🔄 4  💖 15  🔄
HelenOS as Anykernel

- client
- VFS server
- IPC
- kernel

- user space
- kernel space
- syscall
- direct call
- upcall
HelenOS as Anykernel

- client
- IPC
- VFS server

- user space
- kernel space
- syscall
- direct call
- upcall
Display server (WiP)

- Eventually to replace the current compositor
- More modular structure than compositor
- Compositor needs memory for all windows
- Better control of rendering and buffering policies
- HelenOS IPC is just a transport
- Different pixel formats, text mode
- And more, see ticket #816
Links

- http://www.helenos.org/wiki/ReleaseNotes/0.9.1
- https://blog.kernkonzept.com/retreat/social/2019/06/06/l4re_treat.html
- https://hackweek.suse.com/19/projects/helenos-of-the
- https://github.com/jxsvoboda/helenos/tree/gfx
- http://www.helenos.org/ticket/816
- https://github.com/jermar/helenos/tree/ipc
http://www.helenos.org
https://github.com/HelenOS/helenos
@HelenOSOrg
@jjermoar

Thank you!