Everything is a device!

The MH microkernel and the MRG runtime.

What happens when you build a kernel based on an idea you had in a pub

Gianluca Guida
glguida@gmail.com

Bruxelles, 3 Feb 2018
Engineer at ZEDEDA, Inc. (stealth)
Past: Apple, Mimecast, Bromium, VUAmsterdam, Citrix, XenSource.
OS affiliations: MINIX3, GNU HURD, Xen
Personal projects for fun, curiosity, learning
  - this is one of my projects
  - building random kernels is my hobby since late ‘90s
I. Introduction to MH/MRG
II. MH kernel architecture
III. NetBSD kernel components (rump)
IV. rumprun unikernels in MH/MRG
V. Conclusion + Q&A
Introduction to MH/MRG
 intro MH

- Microkernel
- Complete name is murgiahack
  - murgia, the hills where I grew up (Apulia, Italy)
  - hack, not a compliment to my code
- Background personal project
- Timeline:
  - 2015: started
  - 2016: presented at FOSDEM2016 (this devroom)
  - mid 2016: on hold
  - late 2017: development restarted
why MH

• Experiment with hardware
  – understanding the hardware I own
  – interacting with devices as directly as possible

• Experiment with software
  – create a modular system
  – use existing software in different ways

• Having fun
  – trying something relatively different
  – see faces of friends when you explain what it is
MH architecture

- Everything is a device(!):
  - An hardware device is exposed as a device
  - Kernel services (e.g., timers) are exposed as devices
  - A process may expose its services as devices

- The interface is low level:
  - Device are attached to a process bus
  - Memory is handled through user-level #PF
  - Syscall bus interface is hardware like: IRQs, IOMMUs, I/O ports

- The rest is UNIX-ish:
  - Fork to create new processes
  - UID/GID to handle resources
MH process interface
intro MRG

• Runtime library and basic services
• Basic primitives
  – Fibers
  – Event handling
  – Memory management
  – Device drivers libraries
• Basic system services
  – Bootstrap server
  – Console
• Native environment for MH processes
  – Exposes architecture of the kernel
  – Abstracts low level details of kernel interface
MH kernel architecture
MH kernel architecture

Machine Dependent

Machine Independent

Platform

Device

System

Device

System

Bus

User

Devices

user

space

devices

bootstrap

mm

cpu

sched

HW

Devices

Timer

pmap

pcpu

HW Platform

System

Device

Platform

Device

user

space

devices

Machine Dependent

Machine Independent
MH kernel architecture

- MD and MI code *relatively* traditional
- Different type of devices, same interface:
  - System Device
  - Platform and HW Devices
  - User Devices
- *bootstrap* creates HW Devices
- User space level devices are implemented in kernel by the *usrdev* device type.
NetBSD kernel components (rump)
rump kernels

- The *rump kernel* project has componentised the NetBSD kernel
  - When this happen, a lot of interesting thing can be done!
- MH interest is in reusing filesystem, network drivers and stack.

**Porting rump to your system is EASY**
- Interface with the external world has been condensed in a sane, easily portable component called *librumpuser*
- Writing librumpuser for mrg (called *librumpmrg*) and get it to an okay state took less than a day
  - Fear not, fighting the build system will entertain you for weeks!

- After linking a few librump modules, I can use MRG’s own *libahci* to read and write ext4fs (or anything else supported by NetBSD) from real hardware. I am running *stable* file system code in a young OS!
rump + MRG = <3

- MRG building blocks are libraries, as it is for rump components.
- In a world where everything is a device, device drivers are libraries in userspace. Rump gives us exactly that!
- I can create a single server that handles different devices based on linking.
  - Okay, it is a makefiles nightmare but cool!
- “Look ma’, network drivers!”
rumprun unikernels in MH/MRG (WIP)
rumprun unikernels

- `rumprun` is a set of tools and an build of a full rump kernel system.
- It allows to run *unmodified* POSIX programs as **unikernels** in a system that exposed a very simple and lowlevel interface.
- `rumprun-packages` is a repository of ready made unikernels (apache2, mysql, etc.)
- Xen and real hardware currently supported.
- `rumprun` implements its own `librumpuser`
- Porting interface is called `bmk`
**rumprunmrg**

- *rumprunmrg* is a port of *rumprun* for MRG.
- Implements its own *bmk* interface.
- **Porting the *bmk* interface to your system is easy!**
- The build system is a tad bit more complicated than *rump*.
- Took about two days to port rumprun to MRG.
- First result was MH booting into *mpg123*
  - “*I might not have a shell, but I can decode MP3s!*”
Conclusion and Q&A

- The goal of being fun has been reached
- Running *unikernels* as processes in a world where everything is abstracted into an hardware interface is strangely natural.
- Lots to do!
- Code all online (BSD license)
  - But it is a bit scattered, do ask me if you can’t find things.
Questions?

Thank you for listening!

Website:

mhsys.org

Github:

github.com/glguida/mh
Thank you!