Introducing Genode

Norman Feske
Genode Labs

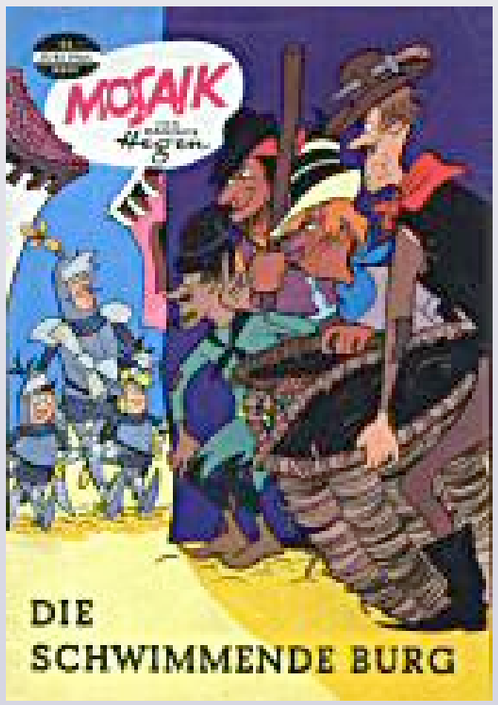
GENODE LABS

Overview

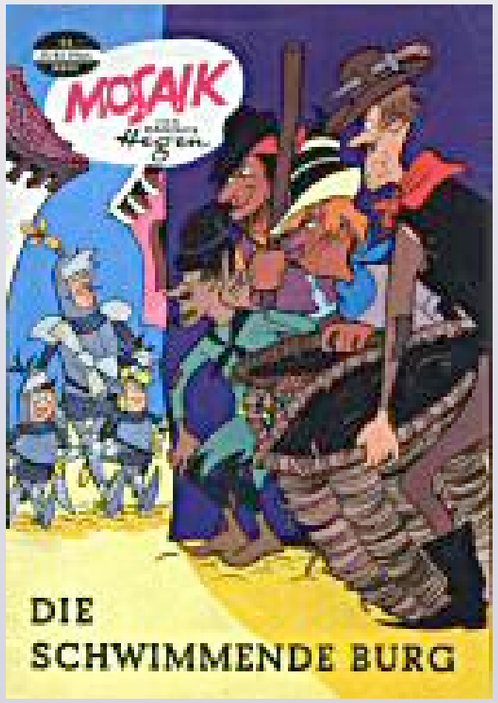
1. Why do we need another operating system?
2. Genode OS architecture at a glance
3. Features of the framework
4. Showcases
5. Plans for 2012

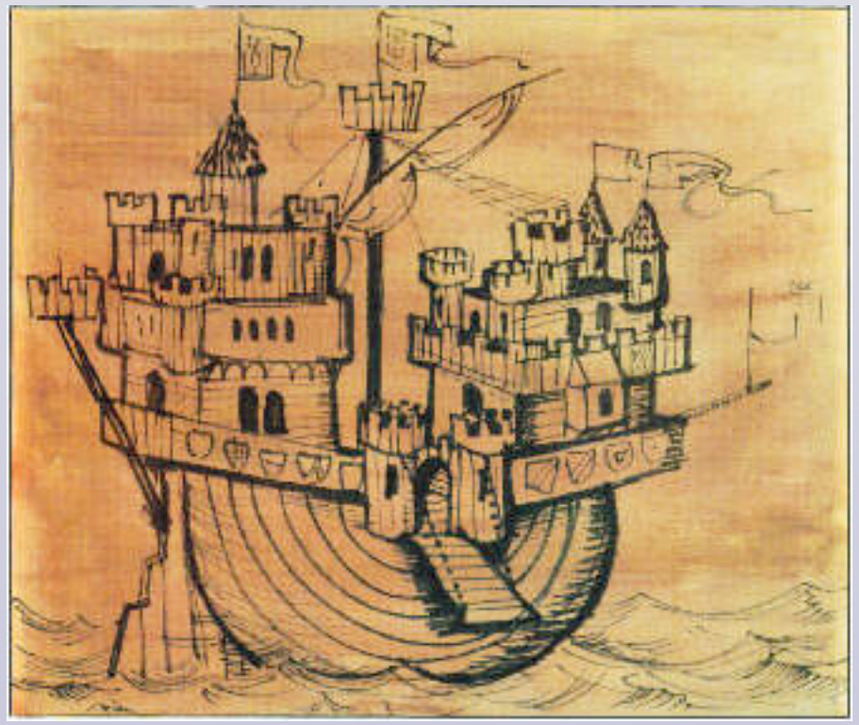
Why do we need another operating system?

Traditional technology, pimped up



Traditional technology, pimped up



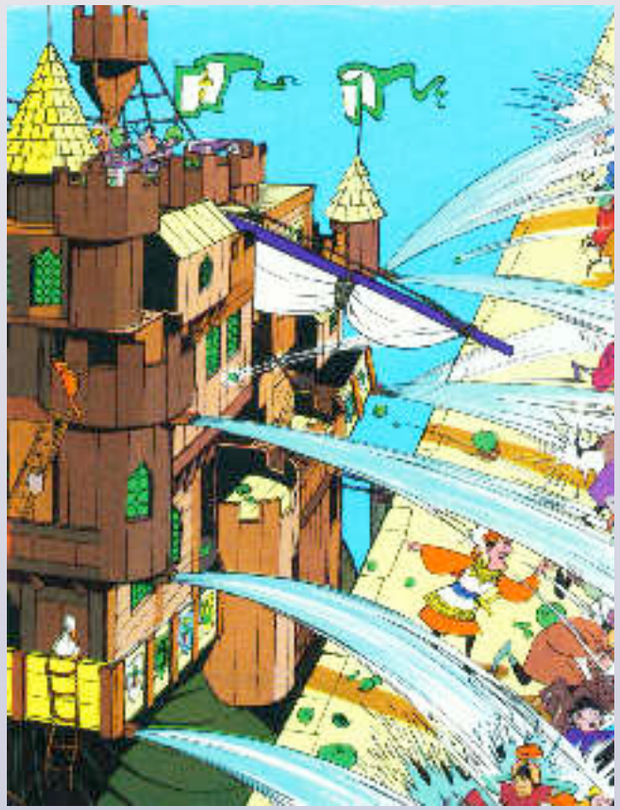


We are getting there...

Work in progress

We are getting there...

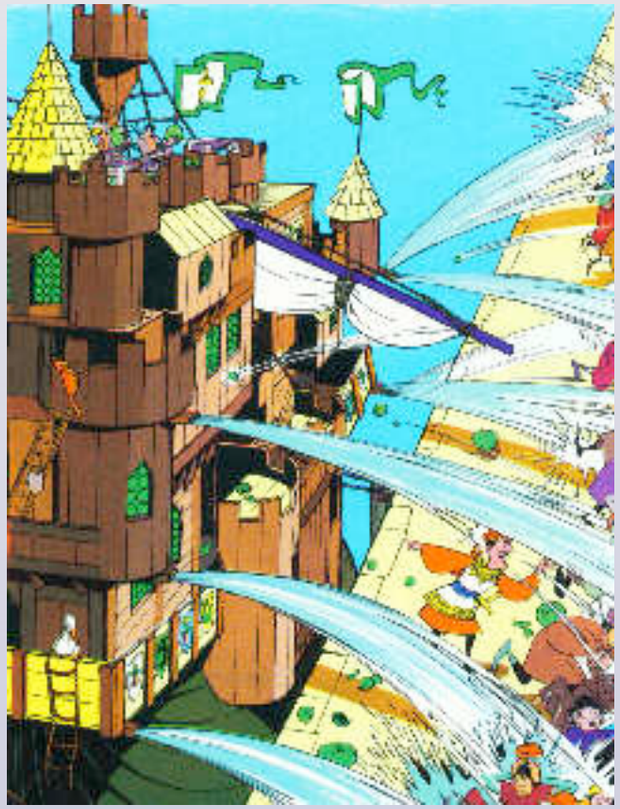
Work in progress



Security features

We are getting there...

Work in progress



Security features

Thriving community

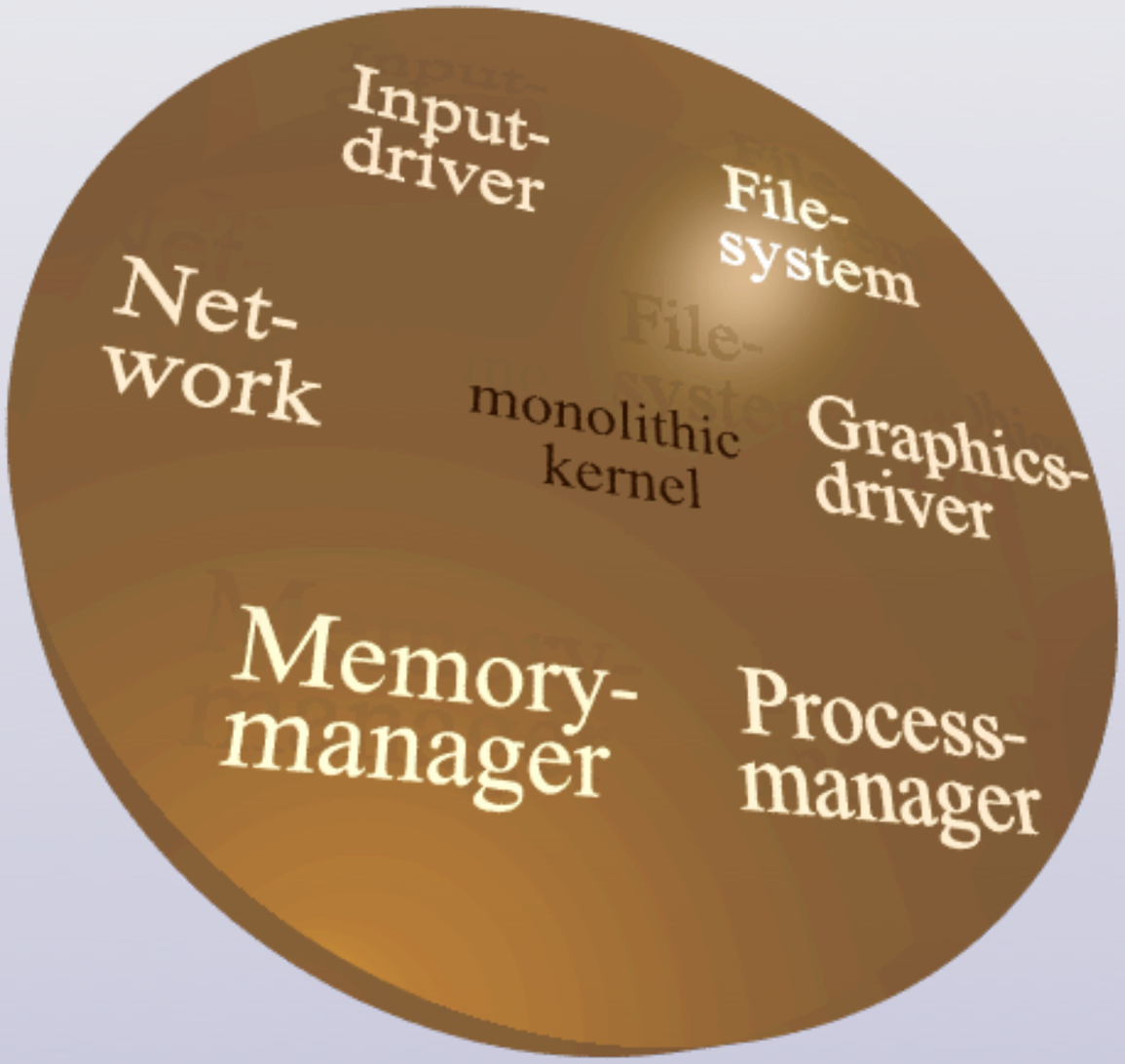
But...

What happens in the event of

- Storm
- Fire
- Leak
- Sabotage
- Directed remote attack

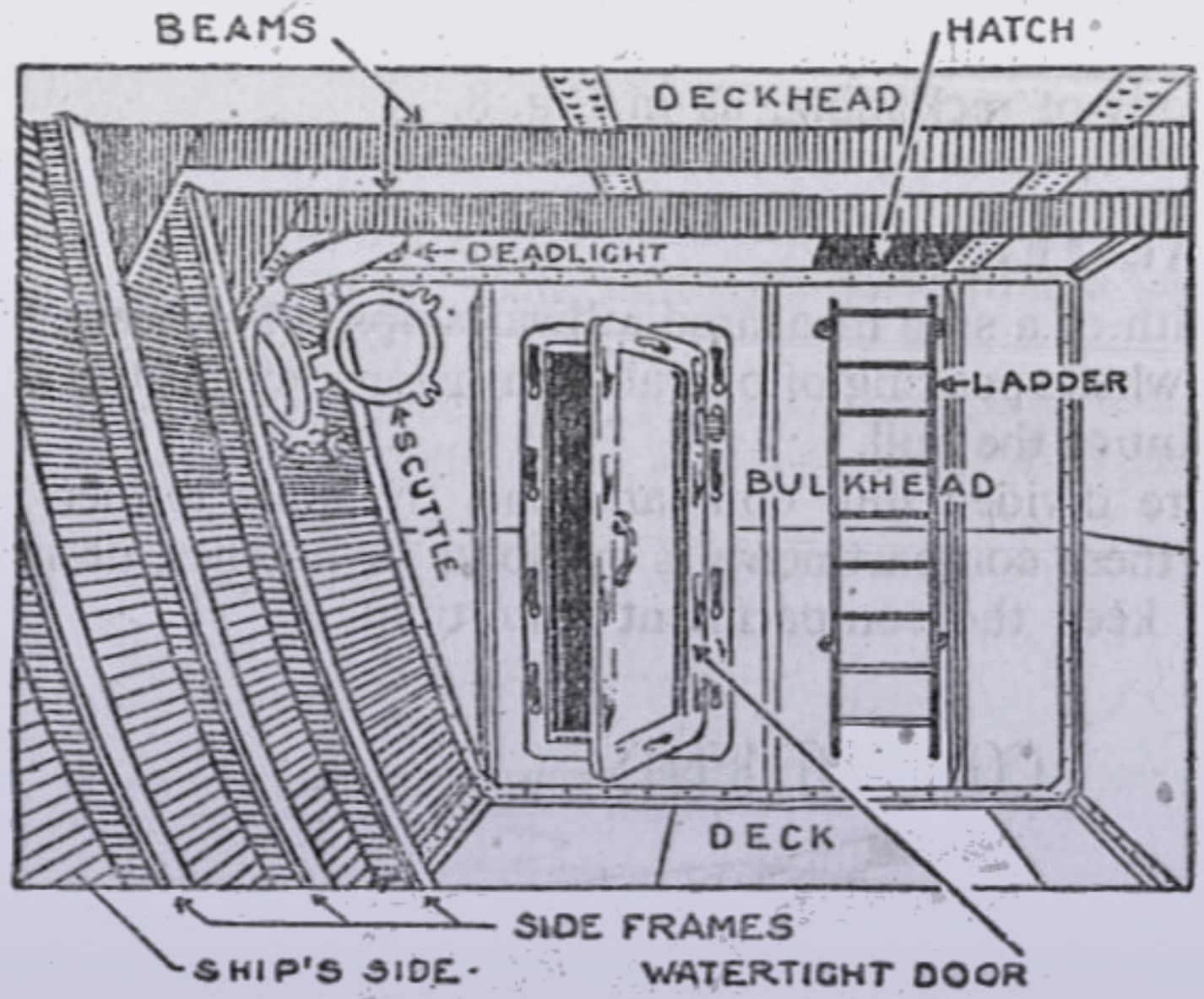
?

Genode OS architecture - Why?



Genode OS architecture - Why?

Bulkhead to the rescue



Genode OS architecture - Why?

Genode OS architecture - Why?

Compromises

Solution is

- Rather inflexible
- Costly (additional material)
- Adding weight (overhead)
- Bureaucratic (additional policy)

Central question:

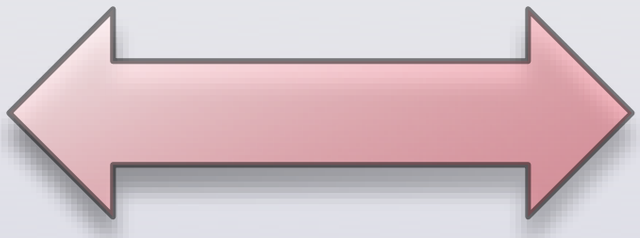
How to organize all those components in order to scale?

Genode OS architecture

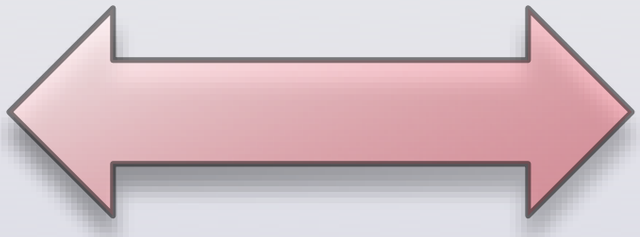
Leitmotif:

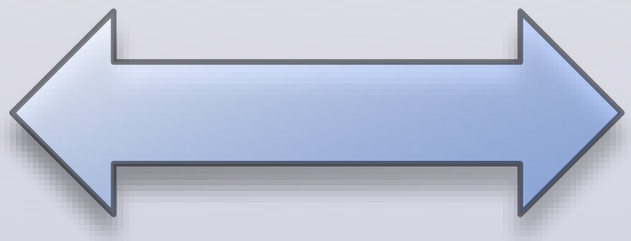
Minimize trusted computing base (TCB)
per application

Genode OS architecture - Universal truths (?)

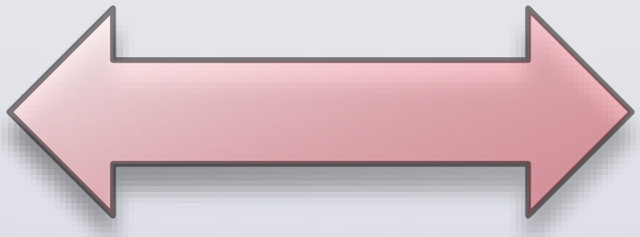
Ease of use  Security

Genode OS architecture - Universal truths (??)

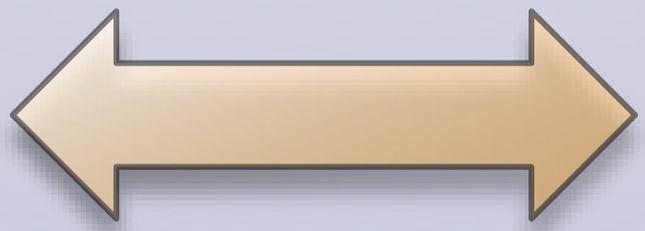
Ease of use  Security

Resource utilization  Resource accountability

Genode OS architecture - Universal truths (???)

Ease of use  Security

Resource utilization  Resource accountability

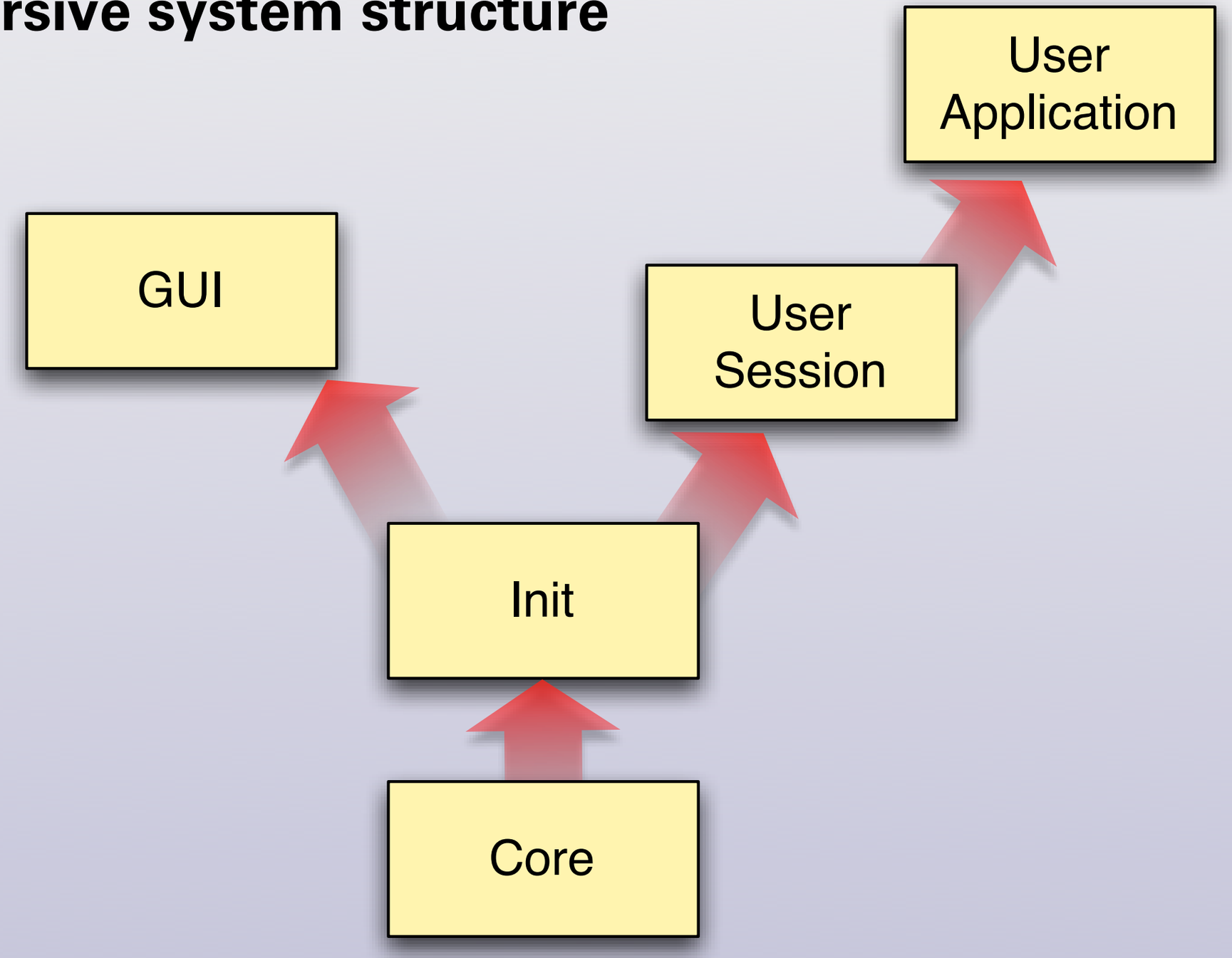
Simplicity  Scalability

Genode OS architecture

Genode sets out to solve these conflicts.

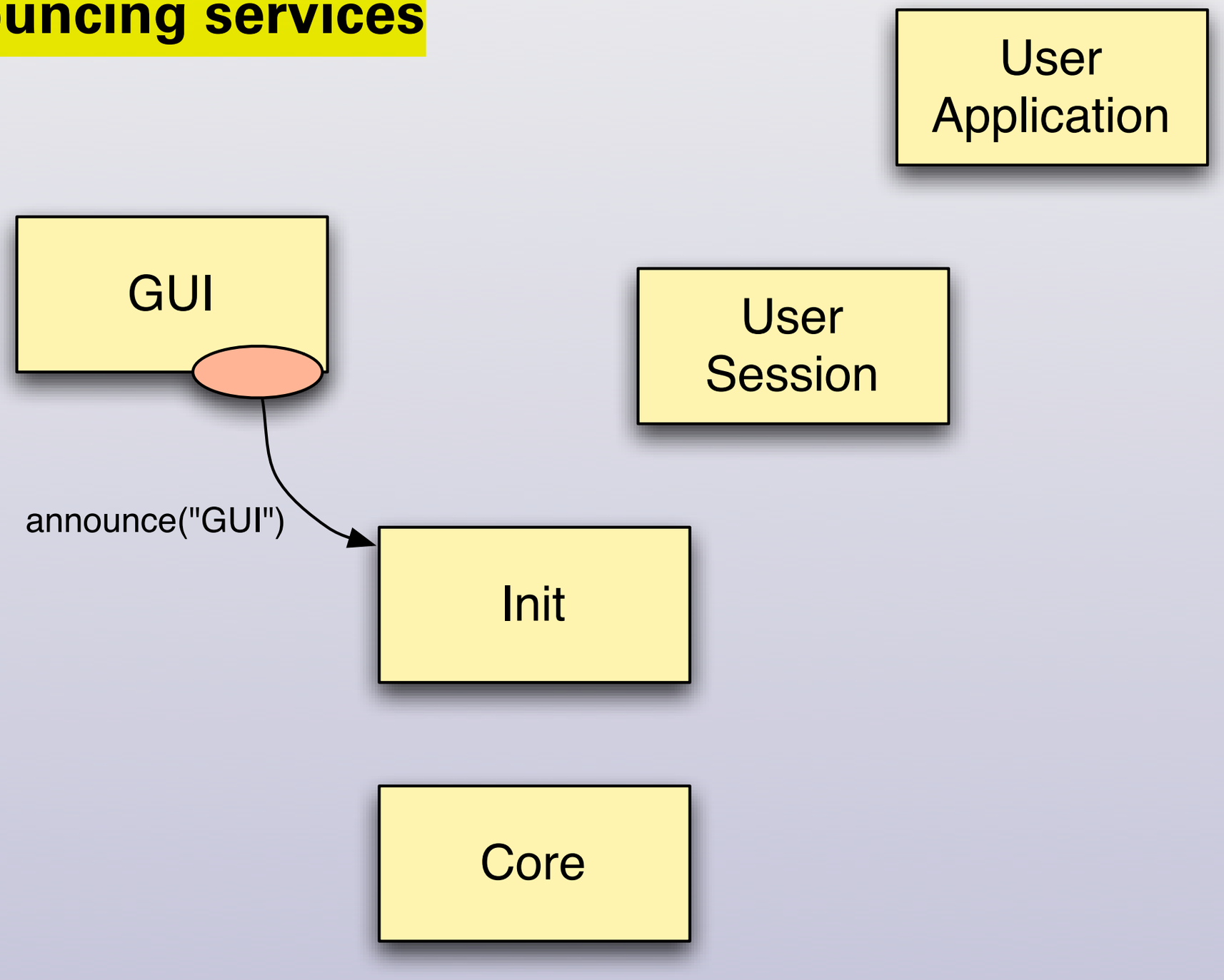
Principles of the architecture

Recursive system structure



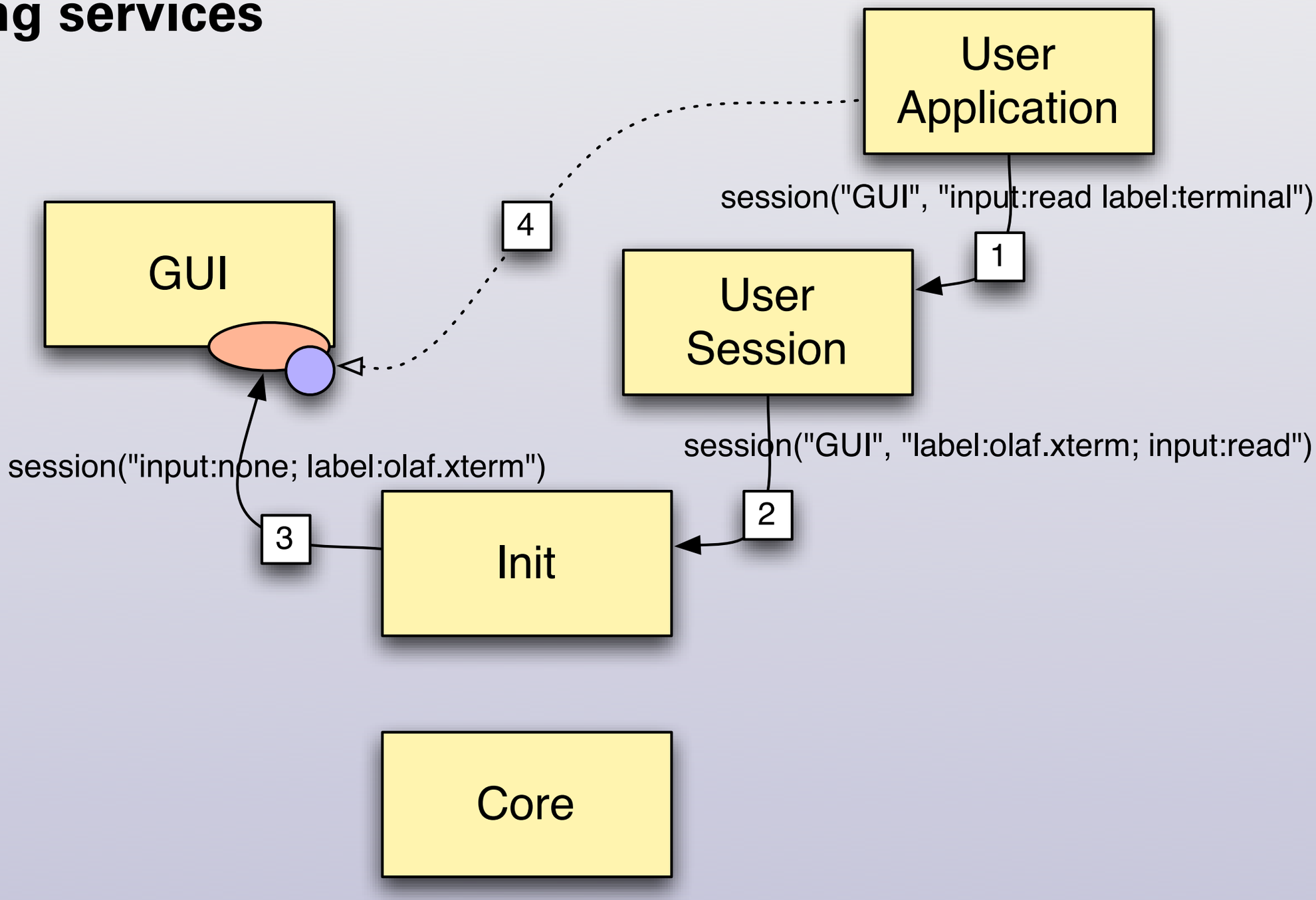
Principles of the architecture (II)

Announcing services



Principles of the architecture (III)

Using services



Principles of the architecture (IV)

Core - the root of the process tree

- Provides fundamental services:
  RAM, ROM, IRQ, I/O, RM, CPU, PD, CAP, LOG, SIGNAL
- Abstracts physical platform resources
- Policy-free
- Bootstraps the init process

Principles of the architecture (V)

Physical resources

- Physical resources are assigned to processes
- A client can lend its resources to services
- A server uses client resources by contract
- A client can regain resources

Principles of the architecture (VI)

Delegation of rights

- Each process lives in a virtual environment
- A process that possesses a right *(capability)* can
  - use it *(invoke)*
  - delegate it

Demonstration

One demo tells more than thousand slides.

Framework features

Pick one of 8 different kernels

FIASCO.OC

OKL4

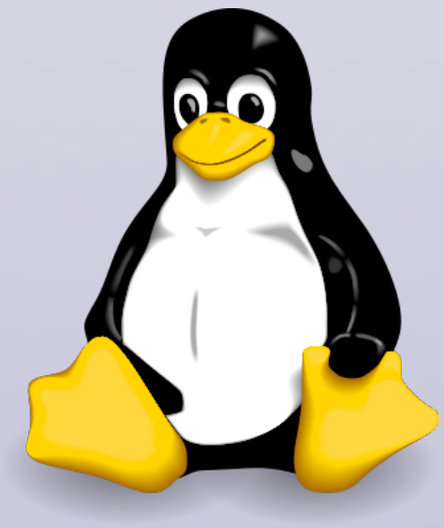
CODEZERO

FIASCO

NOVA

Microhypervisor

MicroBlaze



Ways for reusing existing software

1. **Support for existing APIs**
   POSIX (FreeBSD libc), libSDL, OpenGL, Qt4

   → enables Freetype, libpng, Python, MuPDF, ...

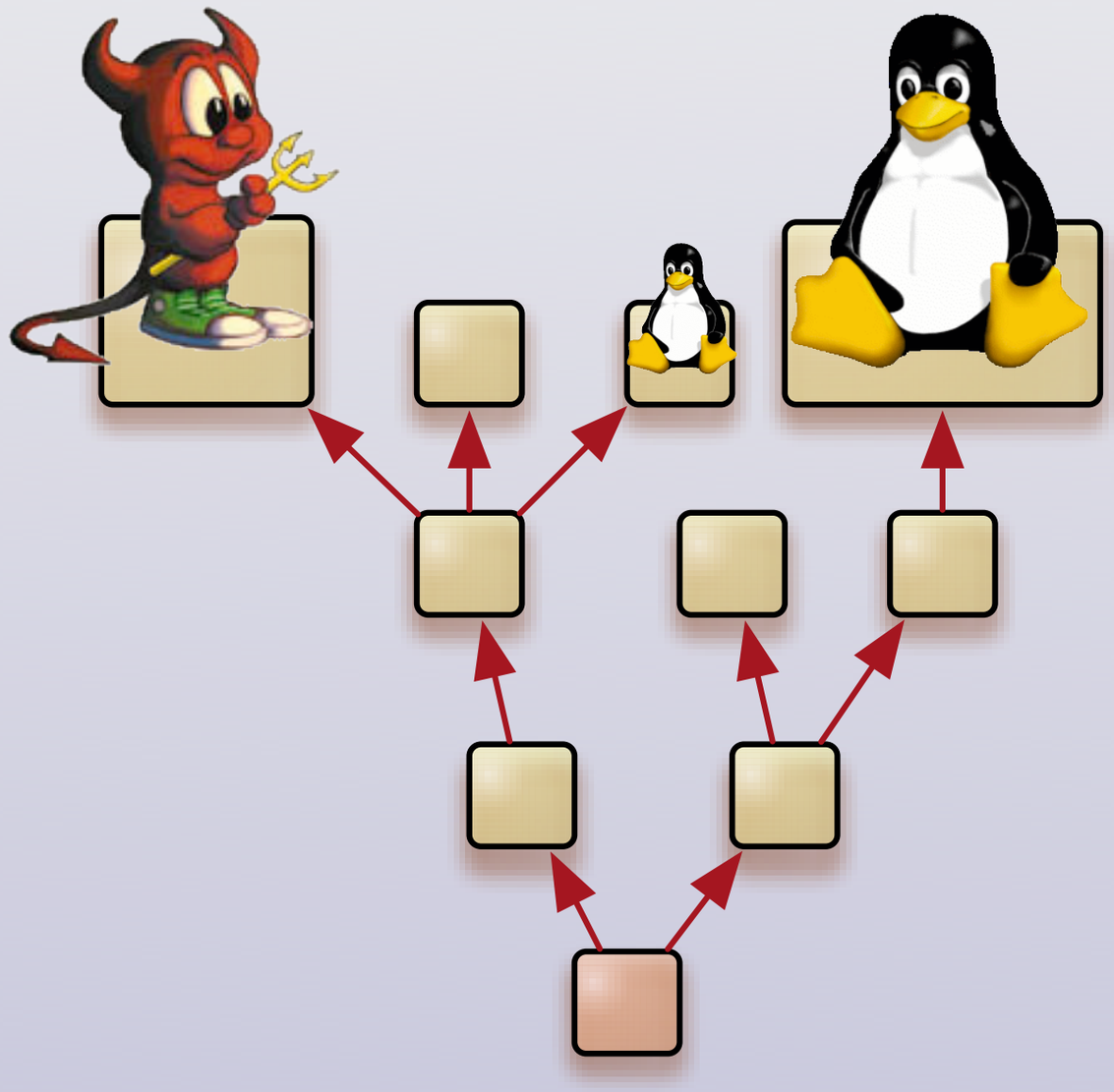
2. **Runtime environments**
   Linux / iPXE Device Driver Environment, Noux

3. **Virtualization**
   Paravirtualized Linux *(on OKL4, Fiasco.OC)*

   → runs unmodified Linux applications

   Faithful virtualization *(Vancouver on NOVA)*

Virtualization-enabled application compatibility



Expressing policy

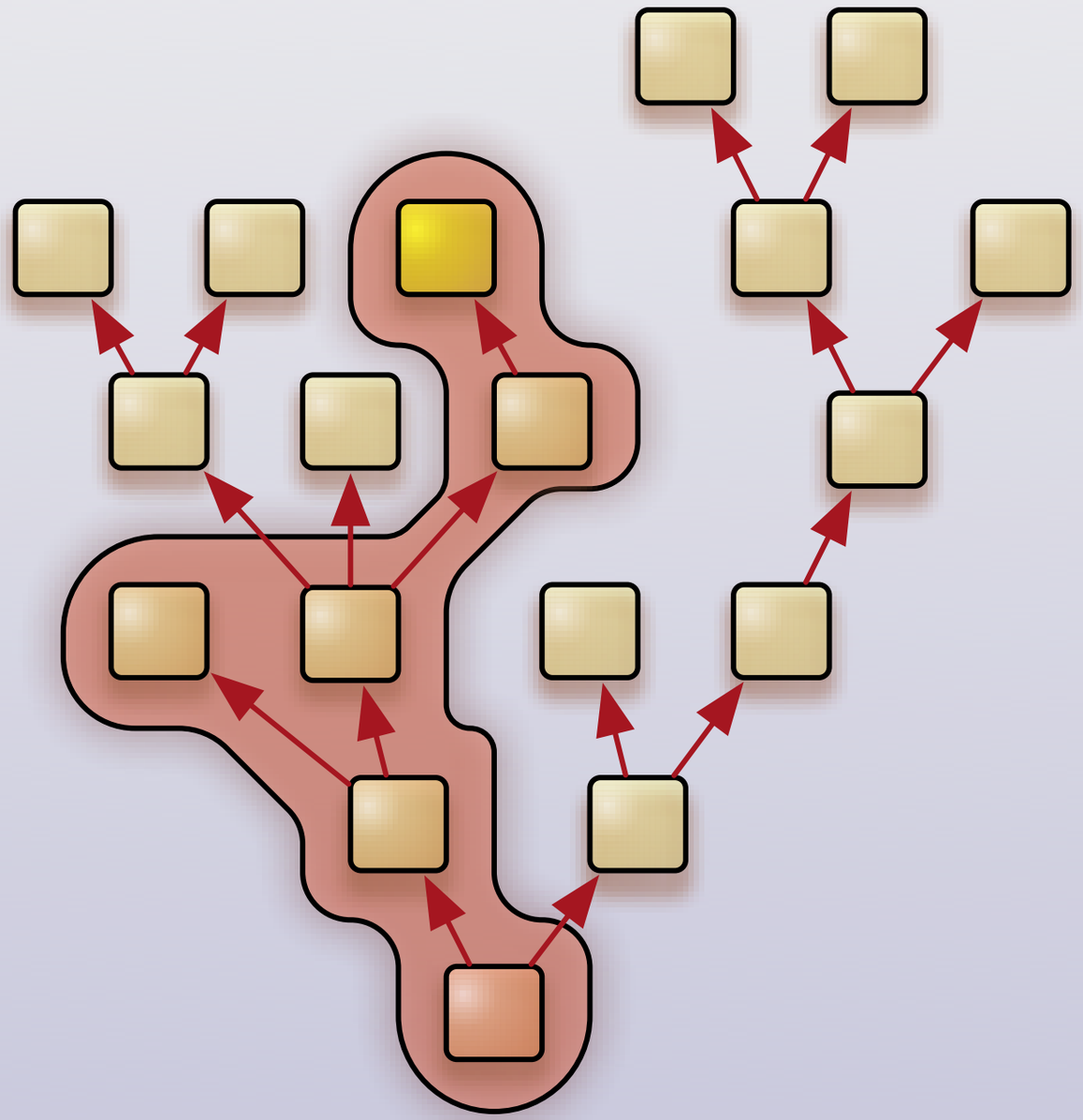
Security

- Uniform configuration concept
- Mandatory access control

Real-time

- Assign hard priorities to subsystems

Application-specific trusted computing base



Trusted computing base in numbers

Lines of code (OKL4 version)

Demo 3	34,200	Demo 2 + PNG support
Demo 2	20,600	Demo 1 + Liquid-FB, Nitlog, Scout
Demo 1	15,000	PS/2, Timer, Nitpicker, Test Application
Core + Init	10,800	
Core	9,400	

Trusted computing base in numbers

Lines of code (OKL4 version)

Demo 4	634,200	Demo 3 + simple Qt4 application
Demo 3	34,200	Demo 2 + PNG support
Demo 2	20,600	Demo 1 + Liquid-FB, Nitlog, Scout
Demo 1	15,000	PS/2, Timer, Nitpicker, Test Application
Core + Init	10,800	
Core	9,400	

Components

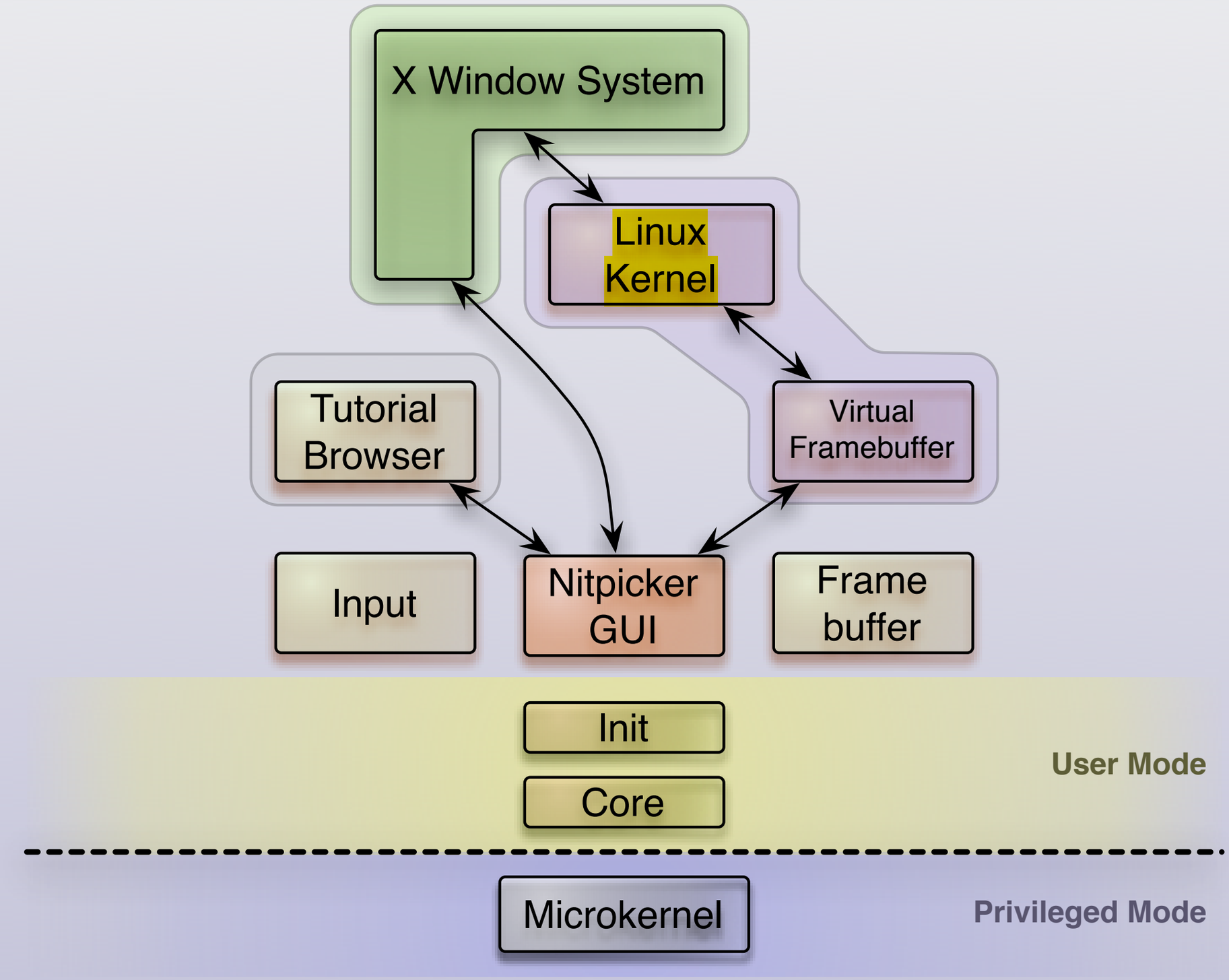
User-level device drivers

- Platform drivers for x86 and ARM
- USB, PCI, PS/2, timer, framebuffer
- 3D graphics (Intel GEM)
- Audio out (Linux drivers)
- Networking (iPXE drivers, Lan9118, MadWifi)
- Block devices (ATAPI, SATA, SD-card, USB)

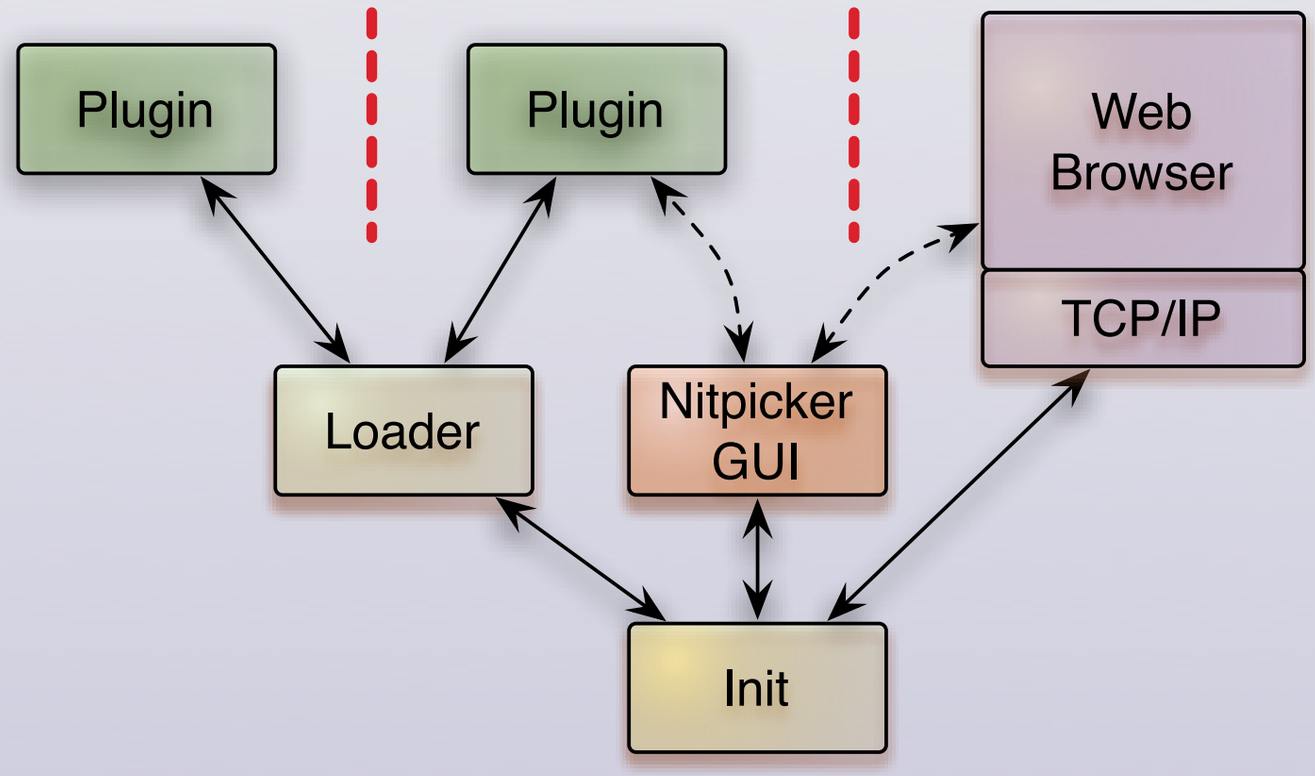
Protocol stacks

- GUI, Qt4
- DDE Kit (device driver API)
- TCP/IP (lwIP)
- Mesa/Gallium3D

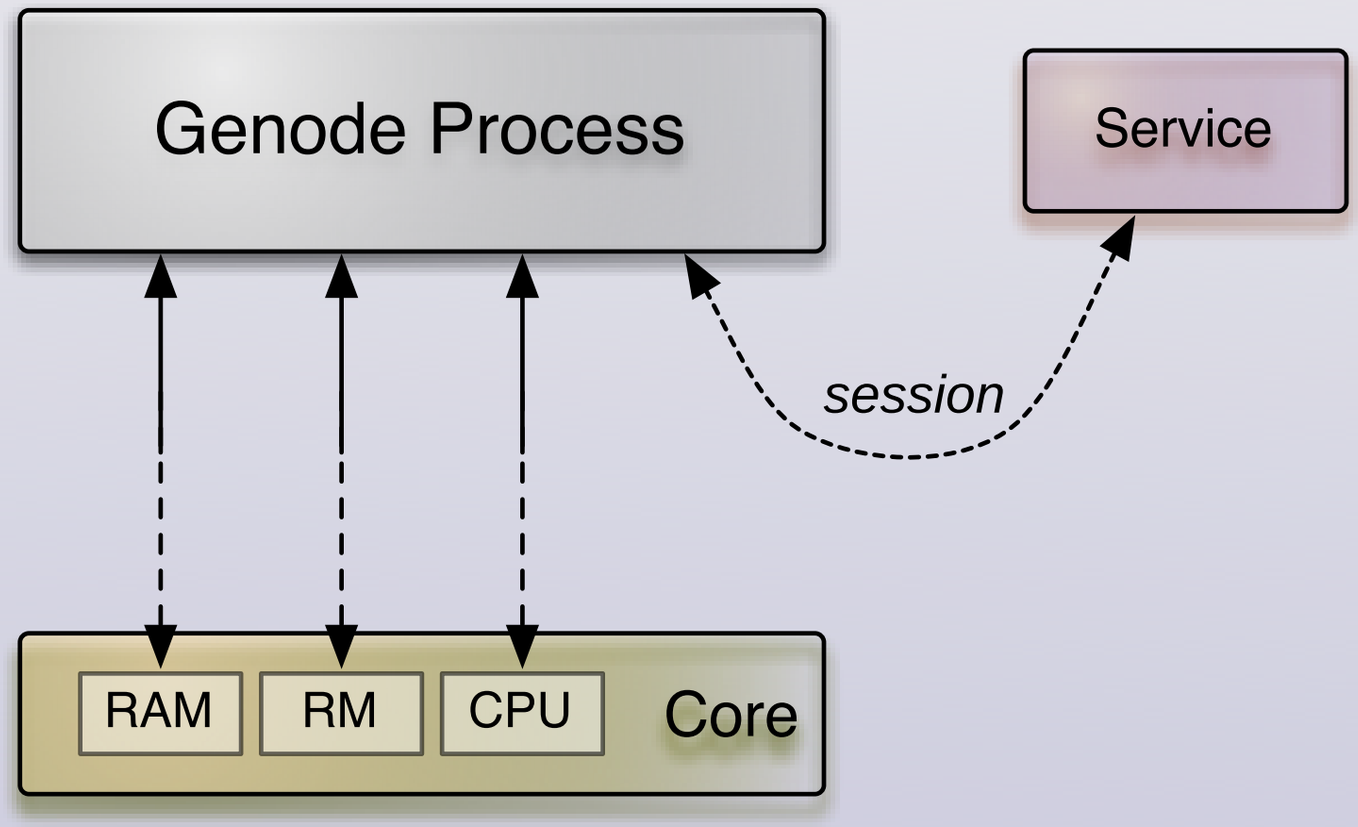
Showcase - Secure GUI



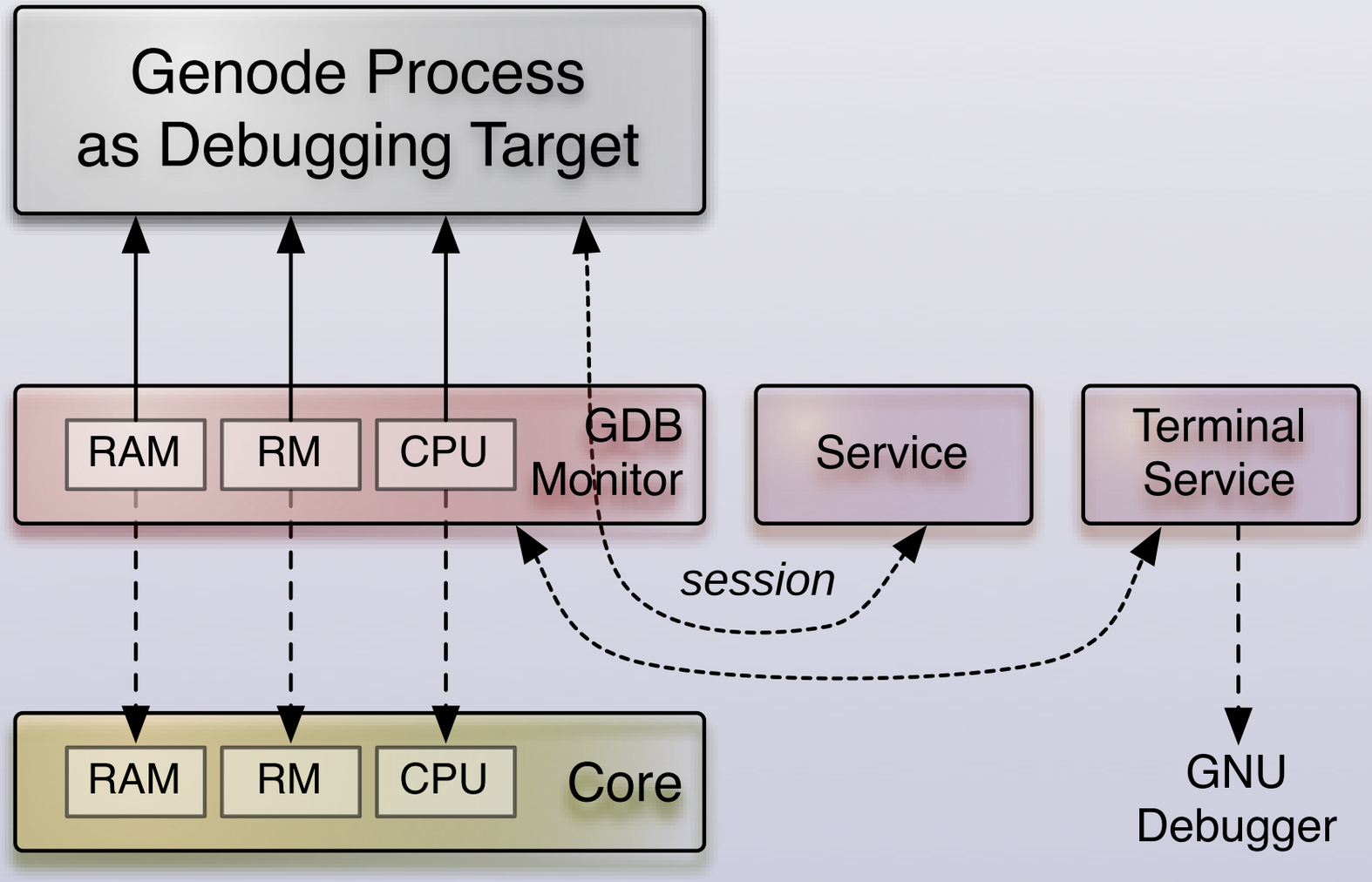
Showcase - Secure browser plugins



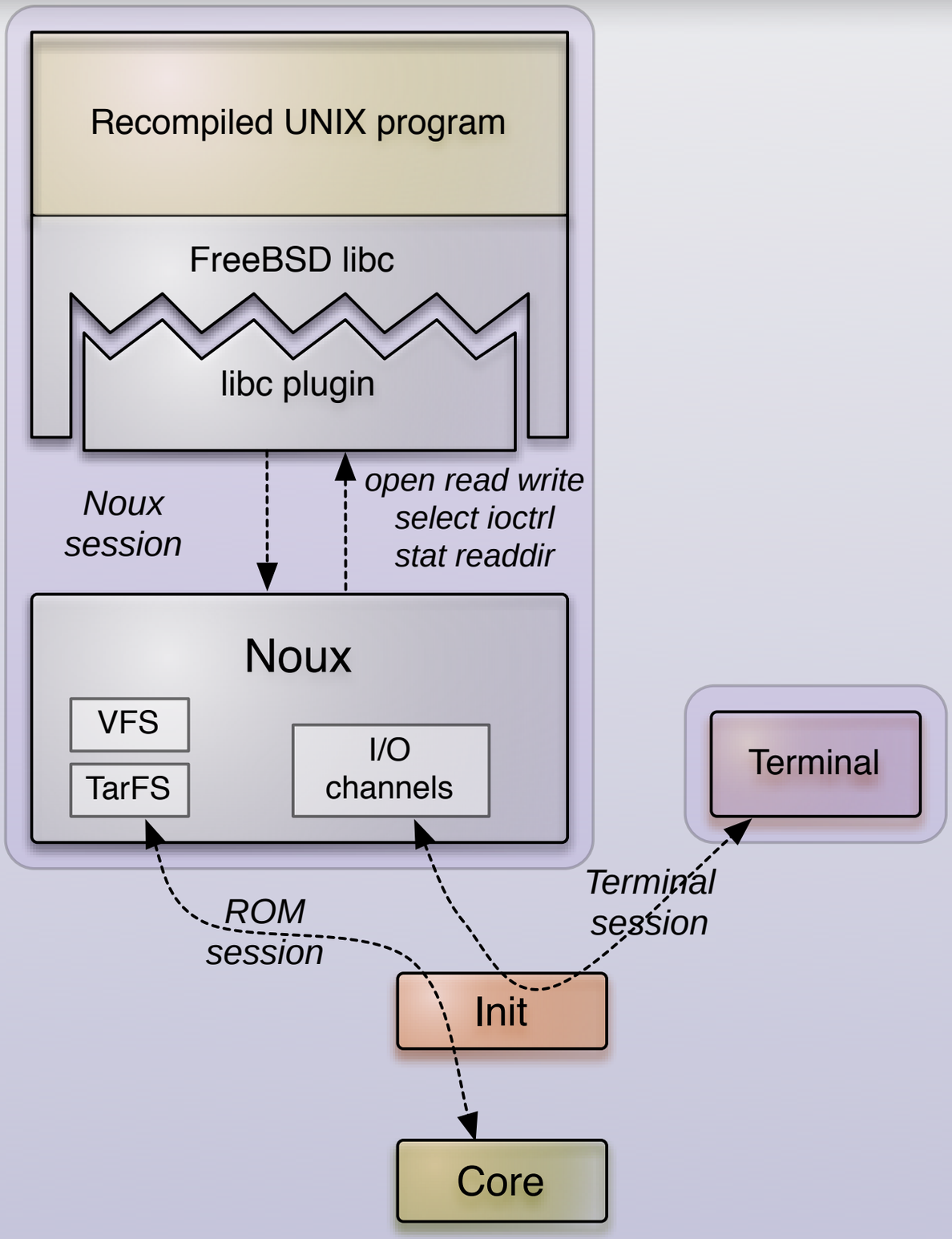
Showcase - Application-level virtualization



Showcase - Application-level virtualization

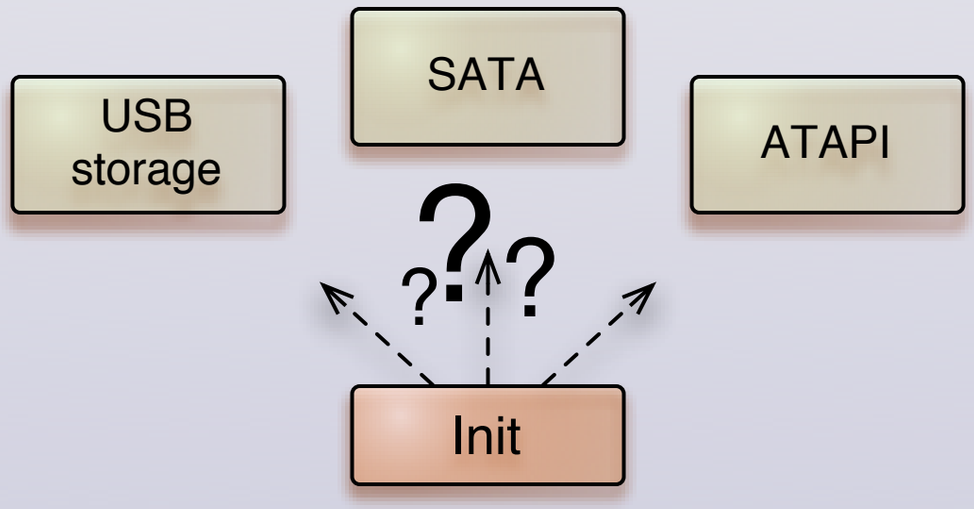


Showcase - OS-level virtualization



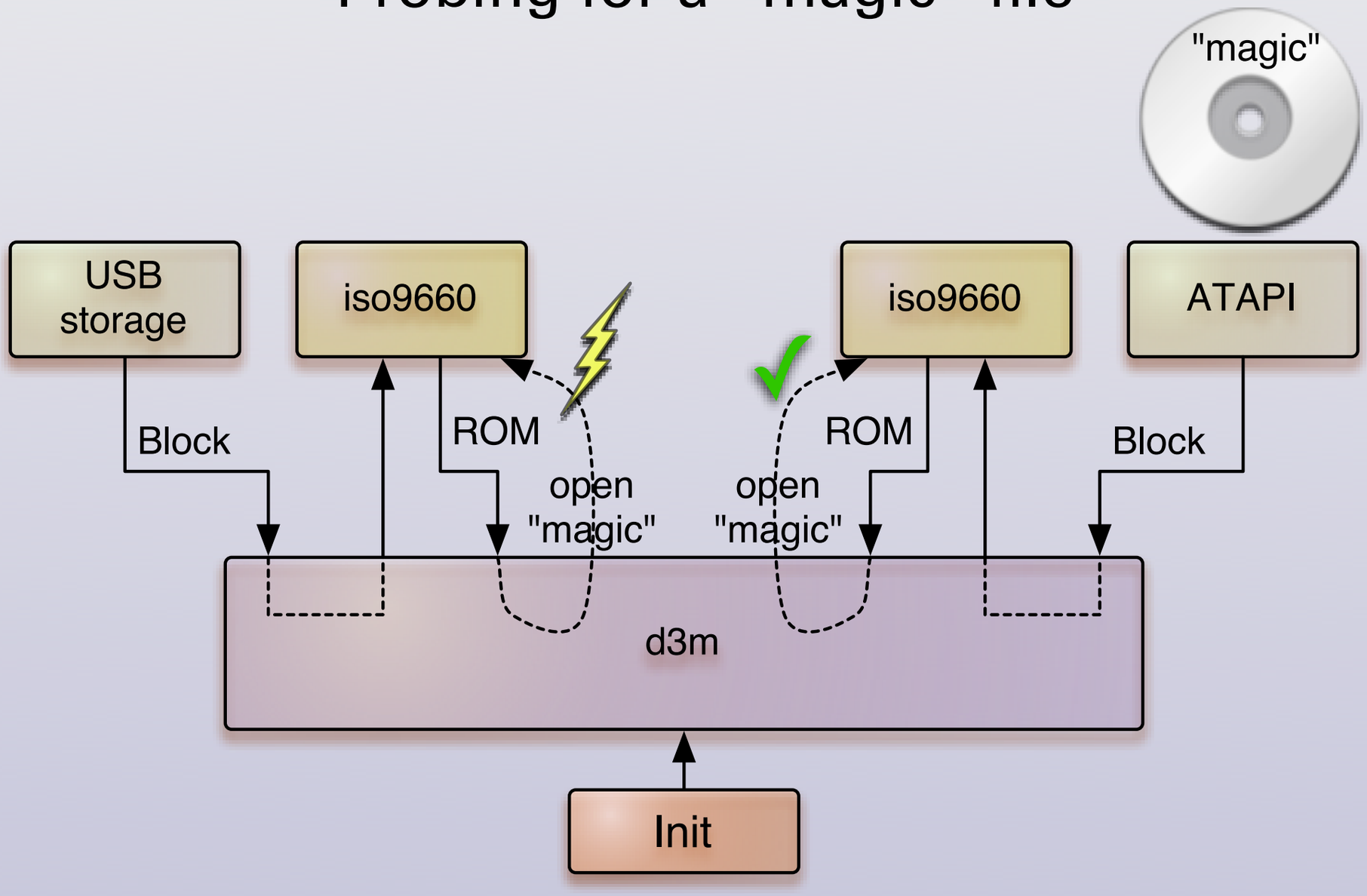
Showcase - Enslaving services (I)

Where to boot from?



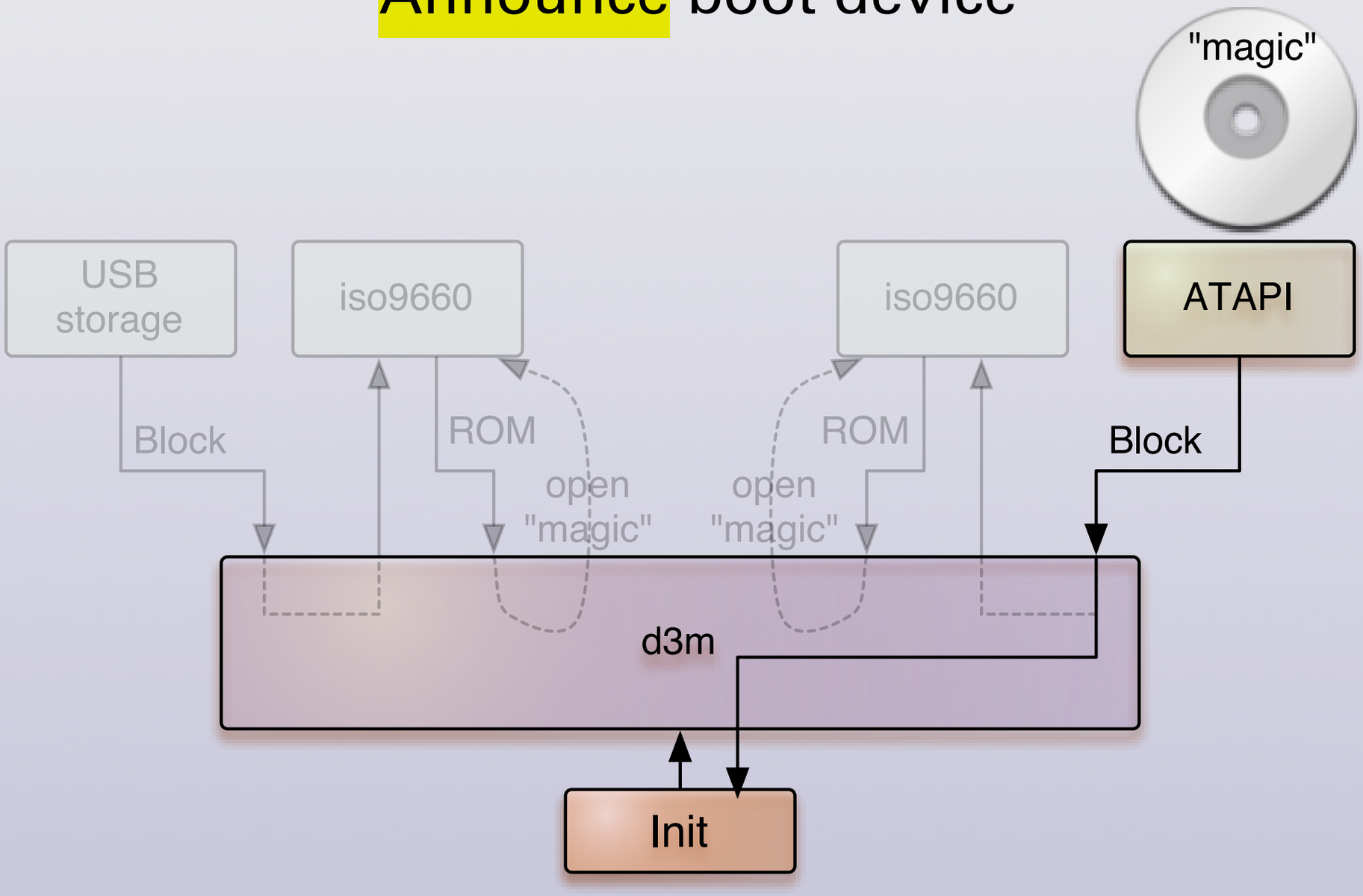
Showcase - Enslaving services (II)

Probing for a “magic” file



Showcase - Enslaving services (III)

Announce boot device

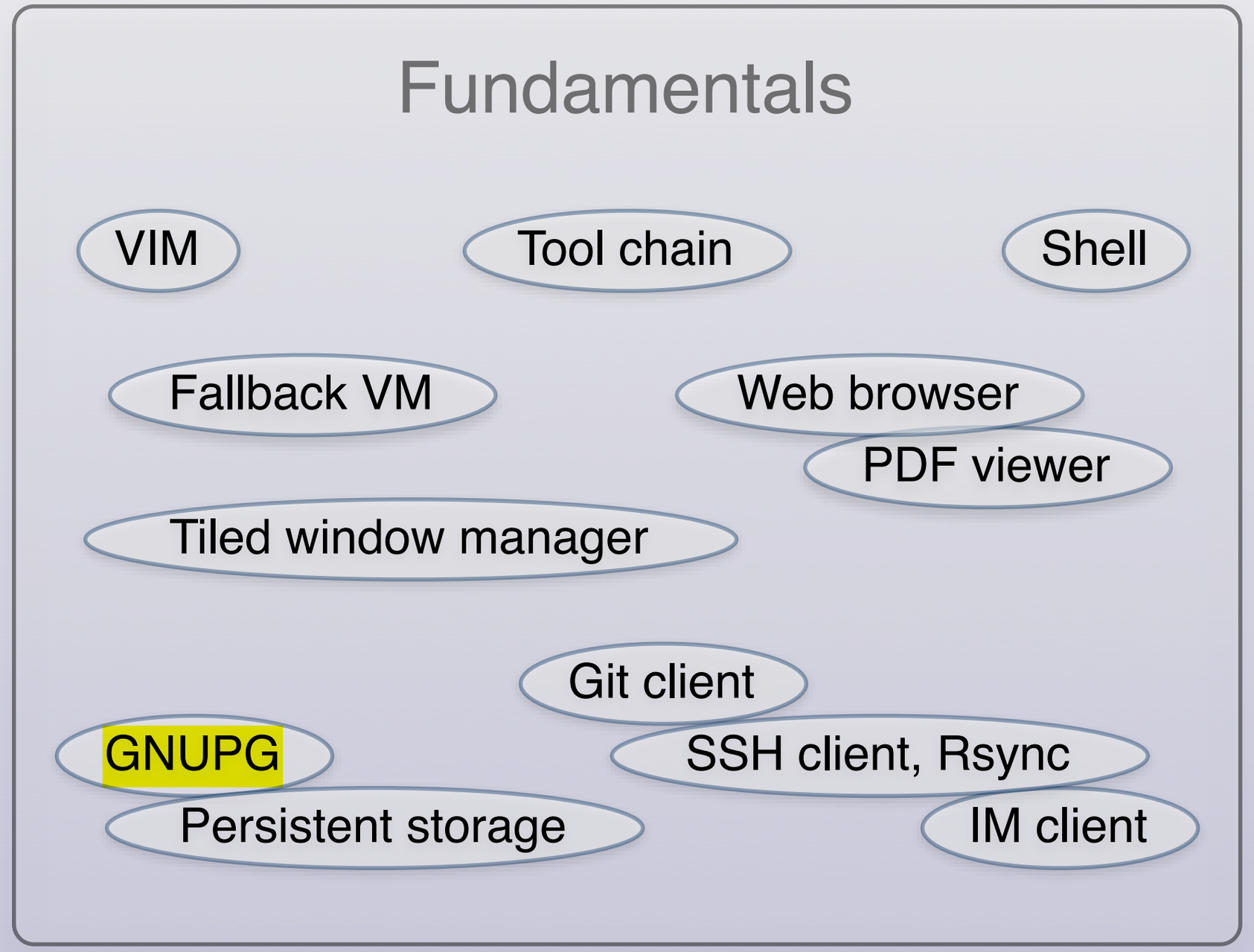


Plans for 2012

Eating our own dog food

→ Goal: Genode as our primary OS by end of year

Inventory of our computing needs



Nice to have

EMACS

Intel Wireless

Qemu

Thinkpad ACPI

Music player

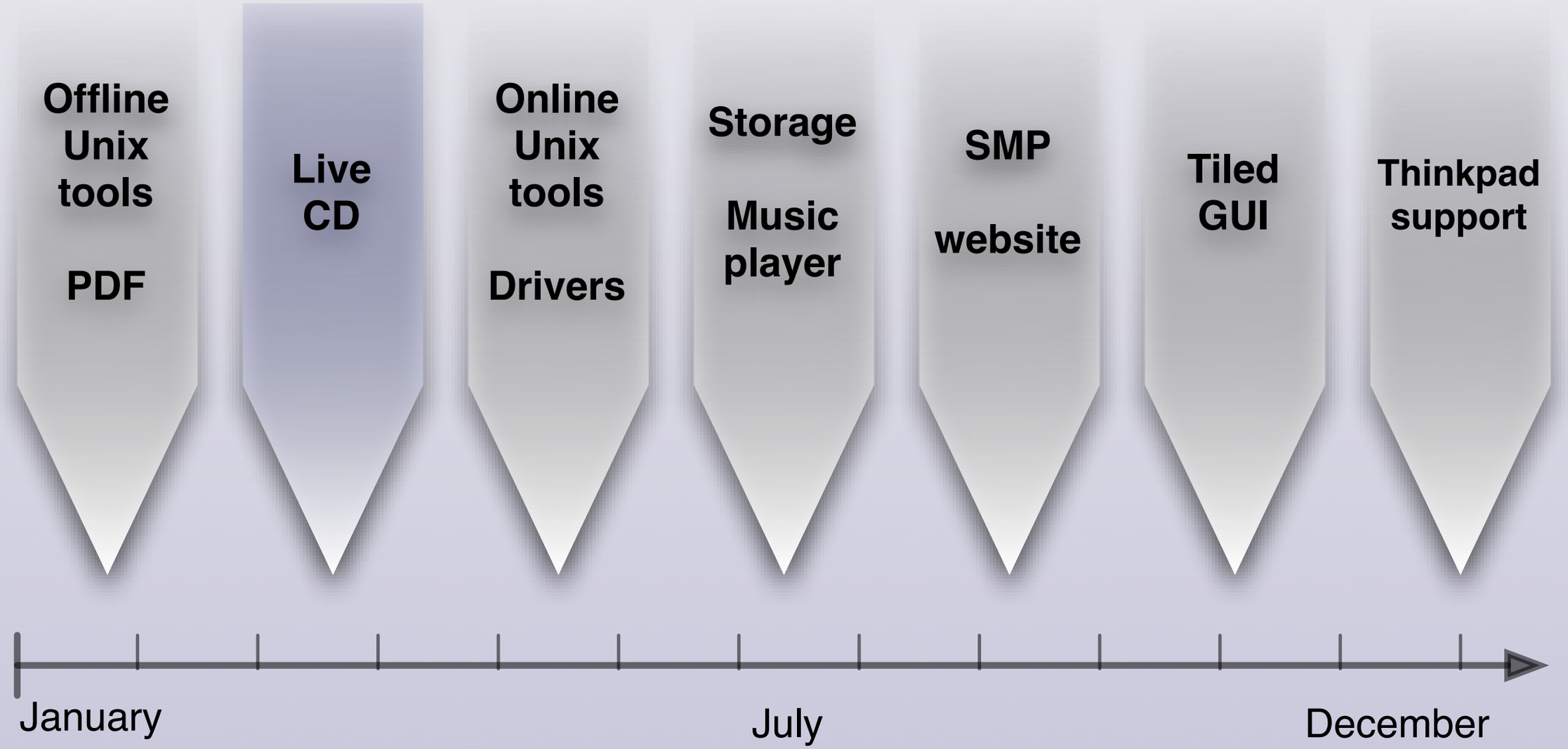
Mail-user agent

Tuxpaint

High-performance graphics

Additional command-line tools

Roadmap 2012



Questions?

Thank you.

http://genode.org

https://**github**.com/genodelabs/

norman.feske@genode-labs.com