Introducing Genode

Norman Feske
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

Copyright Tessloff-Verlag / MOSAIK Steinchen für Steinchen Verlag
http://www.mosapedia.de/wiki/index.php/Burgenschiff
We are getting there...

- Work in progress
- Security features
- Thriving community

Copyright Tessloff-Verlag / MOSAIK Steinchen für Steinchen Verlag
http://www.mosapedia.de/wiki/index.php/Burgenschiff
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?
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 sets out to solve these conflicts.
Principles of the architecture

Recursive system structure

- GUI
- Init
- Core
- User Session
- User Application
Announcing services

- GUI
- User Application
- User Session
- Core
- Init

announce("GUI")
Using services

```
session("GUI", "input:read label:terminal")
session("GUI", "label:olaf.xterm; input:read")
session("input:none; label:olaf.xterm")
```

```
User Application
```

```
User Session
```

```
Init
```

```
Core
```

Principles of the architecture (III)

Using services
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
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
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
- L4Ka
- MicroBlaze
- Microhypervisor

FIASCO OC
FIASCO
L4Ka

MicroBlaze

NOVA

Microhypervisor
Ways for reusing existing software

1. Support for existing APIs
   *POSIX (FreeBSD libc), libSDL, OpenGL, Qt4*
   \[\rightarrow\] enables Freetype, libpng, Python, MuPDF, ...

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

3. Virtualization
   Paravirtualized Linux *(on OKL4, Fiasco.OC)*
   \[\rightarrow\] 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)

<table>
<thead>
<tr>
<th>Component</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demo 3</td>
<td>34,200</td>
</tr>
<tr>
<td>Demo 2</td>
<td>20,600</td>
</tr>
<tr>
<td>Demo 1</td>
<td>15,000</td>
</tr>
<tr>
<td>Core + Init</td>
<td>10,800</td>
</tr>
<tr>
<td>Core</td>
<td>9,400</td>
</tr>
<tr>
<td>Demo 2 + PNG support</td>
<td></td>
</tr>
<tr>
<td>Demo 1 + Liquid-FB, Nitlog, Scout</td>
<td></td>
</tr>
<tr>
<td>PS/2, Timer, Nitpicker, Test Application</td>
<td></td>
</tr>
</tbody>
</table>
### Trusted computing base in numbers

#### Lines of code (OKL4 version)

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

- X Window System
- Linux Kernel
- Virtual Framebuffer
- Nitpicker
- GUI
- Tutorial Browser
- Input
- Frame buffer
- Init
- Core
- Microkernel
- User Mode
- Privileged Mode
Showcase - Secure browser plugins

- Plugin
- Plugin
- Loader
- Nitpicker GUI
- Init
- Web Browser
- TCP/IP
Showcase - Application-level virtualization

Genode Process

Service

session

RAM  RM  CPU
Core
Showcase - Application-level virtualization

Genode Process as Debugging Target

RAM  RM  CPU

GDB Monitor

Service

Terminal Service

GNU Debugger

Core

session
Showcase - OS-level virtualization

Recompiled UNIX program

FreeBSD libc

libc plugin

Noux session

open read write
select ioctl
stat readdir

Noux

VFS
TarFS
I/O channels

Terminal

ROM session

Terminal session

Init

Core
Where to boot from?

USB storage  SATA  ATAPI

Init
Showcase - Enslaving services (II)

Probing for a “magic” file

- USB storage
- ATAPI
- iso9660
- Block
- ROM
- open "magic"
- Init
- d3m
- iso9660
- Block
- ROM
- open "magic"
- "magic"
- ATAPI

"magic"
Showcase - Enslaving services (III)

Announce boot device

- d3m
- Init
- USB storage
- iso9660
- ATAPI iso9660
- Block ROM
- "magic"
Eating our own dog food

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

Fundamentals

- VIM
- Tool chain
- Shell
- Fallback VM
- Web browser
- PDF viewer
- Tiled window manager
- Git client
- GNUPG
- SSH client, Rsync
- Persistent storage
- IM client
Nice to have

- EMACS
- Intel Wireless
- Qemu
- Thinkpad ACPI
- Music player
- Mail-user agent
- Tuxpaint
- High-performance graphics
- Additional command-line tools
Roadmap 2012

- **January**
  - Offline Unix tools
  - PDF
  - Live CD
  - Online Unix tools
  - Drivers

- **July**
  - Storage Music player

- **December**
  - SMP website
  - Tiled GUI
  - Thinkpad support
  - Online Unix tools
  - Drivers
  - Offline Unix tools
  - PDF
  - Live CD
  - Online Unix tools
  - Drivers
  - Storage Music player
  - SMP website
  - Tiled GUI
  - Thinkpad support
Questions?

Thank you.

http://genode.org
https://github.com/genodelabs/
norman.feske@genode-labs.com