Complex Lab – Operating Systems
Graphical Console

Martin Küttler
Last assignment

► Any questions?
► Any bug reports, wishes, etc.?
We are here

Pong Server

Paddle Client 1

Paddle Client 2

Memory Management

Moe

Sigma0

Fiasco Kernel
Today’s goal

Pong Server

Paddle Client 1

Paddle Client 2

Keyboard Driver

Console

Memory Management

Moe

Sigma0

Fiasco Kernel
Graphics (VES)

- Using VBE/XGA BIOS extension
- Put computer into XGA mode:
  - Requires evil real-mode code
  - GRUB: vbeset <mode>
  - L4 FBDRV: command line option -m <mode>
  - mode: 0x100 - 0x11F, see e.g. Wikipedia on VBE
- Get access to hardware frame buffer
- Render graphics into frame buffer
VESAT\textsuperscript{a} on L4Re

- IO server manages all I/O resources
- fb-drv server provides a frame buffer interface.
IO Configuration files:

- Hardware description file (src/l4/pkg/io/io/config/x86-legacy.devfs)
- vbus configuration file (x86-fb.io)
Lua example

```
local L4 = require("L4");
local ld = L4.default_loader;
local vbus = ld:new_channel();
local fbdrv = ld:new_channel();

ld:start({
cap = {fbdrv = vbus:svr(), icu = L4.Env.icu,
       sigma0 = L4.cast(L4.Proto.Factory, L4.Env.sigma0)
       :create(L4.Proto.Sigma0)},
    log = {"I0", "yellow" },
    "rom/io/rom/x86-legacy.devs/rom/x86-fb.io"});

ld:start({
caps = {vbus = vbus, fb=fbdrv:svr()} ,
    log = {"fbdrv", "red"},
    "rom/fb-drv-m-0x117"});

ld:start({
caps = {fb = fbdrv},
    "rom/your_fb_client"})
```
L4Re Framebuffer Interface

Headers are at
- `src/l4/pkg/l4re-core/l4re/include/video/goos`, and
- `src/l4/pkg/l4re-code/l4re/util/include/video/goos_fb`

Interface to `Goos_fb`
- `Goos_fb(char const *name)` — Create FB using capability name
  (channel to fb-driv)
- `Goos_fb::view_info()` — FB information
- `Goos_fb::attach_buffer()` — Get FB data space
- `Goos_fb::refresh()` — refresh, not necessary for physical FB.
Example: Drawing Pixels

```cpp
auto base = fb.attach_buffer();

L4Re::Util::Video::View::Info info;
int r = fb.view_info(&info);
if (r != 0) error(...);

auto addr = base + y * (info.pixel_info.bytes_per_pixel() * info.width) + x * info.pixel_info.bytes_per_pixel();
```

// details about color encoding in info.pixel_info
*static_cast<unsigned*>(addr) = value;

Rendering Text

Use C library: `libgfxbitmap`

- Initialize: `gfxbitmap_font_init();`
- Render text:

```c
gfxbitmap_font_text
(void *fb_base, l4re_video_view_info_t *fbinfo,
gfxbitmap_font_t font, char const *text,
unsigned len, unsigned x, unsigned y,
gfxbitmap_color_pix_t foreground,
gfxbitmap_color_pix_t background);
```

- `fb_base` – base address of FB
- `fbinfo` – L4Re::Framebuffer::Info struct, cast
- Colors are `unsigned int`
- Useful constants: `GFXBITMAP_DEFAULT_FONT`, `GFXBITMAP_USE_STRLEN`
Drawing graphics

- There is a libpng, contact me if you need/want it.
- Consult your favorite Computer Graphics reference for drawing algorithms.
- None of these is necessary for this assignment, as Pong can already draw itself.
Assignment: Graphical text console

- Make your echo server render text into the physical framebuffer (direct access for now)
- Scroll down when the screen is full, as in a terminal.
- When we are going to have input, you might want to scroll up, so keep history.
Next meeting

- Next up: Keyboard driver, graphics multiplexing, and integration.
- We could meet on 17.01. or 31.01. for our last meeting.