




Understanding the Linux Graphics Stack training

Course duration _____

 2 days – 16 hours

Language _____

Materials English

Oral Lecture English
French

Trainer _____

This training is no longer offered by Bootlin.

Contact _____

 training@bootlin.com

 +33 484 258 097

Audience

People developing multimedia devices using the Linux kernel

Training objectives

- Be able to understand the basics of graphics display: image and color representation, pixel drawing, pixel operations.
- Be able to understand graphics hardware: display pipeline components, display and rendering hardware.
- Have a solid understanding of the Linux kernel graphics stack components and role: TTY, framebuffer and DRM subsystems.
- Have a solid understanding of the Linux user-space graphics stack components and role: DRM from user-space, X.org, Wayland, OpenGL.

Prerequisites

- **Solid experience with the C programming language:** participants must be familiar with the usage of complex data types and structures, pointers, function pointers, and the C pre-processor.
- **Experience with low-level development in Linux and hardware interfaces:** participants should have a minimal understanding of memory management, interaction with common hardware interfaces (registers, interrupts), the interaction between Linux user-space applications and the Linux kernel (system calls). Following Bootlin's [Linux kernel driver development course](#) allows to fulfill this pre-requisite.
- **Minimal English language level: B1**, according to the *Common European Framework of References for Languages*, for our sessions in English. See the [CEFR grid](#) for self-evaluation.

Pedagogics

- Lectures delivered by the trainer: 75% of the duration
- Practical demonstrations done by the trainer: 25% of the duration
- Electronic copies of presentations, lab instructions and data files. They are freely available [here](#).

Certificate

Only the participants who have attended all training sessions, and who have scored over 50% of correct answers at the final evaluation will receive a training certificate from Bootlin.

Disabilities

Participants with disabilities who have special needs are invited to contact us at training@bootlin.com to discuss adaptations to the training course.



Onsite
training

Required equipment

For on-site session delivered at our customer location, our customer must provide:

- Video projector
- One PC computer on each desk (for one or two persons) with at least 16 GB of RAM, and Ubuntu Linux 24.04 installed in a free partition of at least 30 GB
- Distributions other than Ubuntu Linux 24.04 are not supported, and using Linux in a virtual machine is not supported.
- Unfiltered and fast connection to Internet: at least 50 Mbit/s of download bandwidth, and no filtering of web sites or protocols.
- PC computers with valuable data must be backed up before being used in our sessions.

For on-site sessions organized at Bootlin premises, Bootlin provides all the necessary equipment.

Training Schedule

Day 1 - Morning

Lecture	Image and Color Representation	<ul style="list-style-type: none">▪ Light, pixels and pictures▪ Sampling, frequency domain, aliasing▪ Color quantization and representation▪ Colorspaces and channels, alpha▪ YUV and chroma sub-sampling▪ Pixel data planes, scan order▪ Pixel formats, FourCC codes, modifiers <p><i>Introducing the basic notions used for representing color images in graphics.</i></p>
Lecture	Pixel Drawing	<ul style="list-style-type: none">▪ Accessing and iterating over pixel data▪ Concepts about rasterization▪ Rectangle drawing▪ Linear gradient drawing▪ Disk drawing▪ Circular gradient drawing▪ Line drawing▪ Line and shape aliasing, sub-pixel drawing▪ Circles and polar coordinates▪ Parametric curves <p><i>Presenting how to access pixel data in memory and draw basic shapes.</i></p>
Lecture	Pixel Operations	<ul style="list-style-type: none">▪ Region copy▪ Alpha blending▪ Color-keying▪ Scaling and interpolation▪ Linear filtering and convolution▪ Blur filters▪ Dithering <p><i>Providing basic notions about filtering, with very common examples of how it's used.</i></p>
Lab	Drawing and operations	<ul style="list-style-type: none">▪ Examples of various shapes and region drawing▪ Examples of basic pixel operations <p><i>Illustrating the concepts presented along the way.</i></p>

Day 1 - Afternoon

Lecture	Pipeline Components and Generalities	Overview <ul style="list-style-type: none">▪ Types of graphics hardware implementations▪ Graphics memory and buffers▪ Graphics pipelines▪ Display, render and video hardware overview <p><i>Presenting the hardware involved in graphics pipelines.</i></p>
---------	--------------------------------------	--

Lecture	Display hardware	<ul style="list-style-type: none"> Visual display technologies: CRT, plasma, LCD, OLED, EPD Display timings, modes and EDID Display interfaces: VGA, DVI, HDMI, DP, LVDS, DSI, DP Bridges and transcoders <p><i>Presenting the inner workings of display hardware.</i></p>
Lecture	Rendering Hardware Specifics	<ul style="list-style-type: none"> Digital Signal Processors (DSPs) Dedicated hardware accelerators Graphics Processing Unit (GPUs) <p><i>Describing the architecture of processing and rendering hardware.</i></p>
Lecture	System Integration, Memory and Performance	<ul style="list-style-type: none"> Graphics integration and memory Shared graphics memory access Graphics memory constraints and performance Offloading graphics to hardware Graphics performance tips <p><i>Topics related to graphics integration, memory management and performance aspects.</i></p>

Day 2 - Morning

Lecture	Display Stack Overview	<ul style="list-style-type: none"> System-agnostic overview: kernel, userspace display and rendering Linux kernel overview Linux-compatible low-level userspace overview X Window and Wayland overview High-level graphics libraries and desktop environments overview <p><i>Presenting what software components are required for modern computer graphics and how they are divided between kernel and userspace.</i></p>
Lecture	TTY Kernel Aspects, Framebuffer Device Kernel Aspects	<ul style="list-style-type: none"> Linux TTY subsystem introduction Virtual terminals and graphics Virtual terminals switching and graphics Fbdev overview Fbdev basic operations Fbdev limitations <p><i>How TTYs interact with graphics in Linux along with a short presentation of fbdev and why it's deprecated.</i></p>
Lecture	DRM Kernel Aspects	<ul style="list-style-type: none"> DRM devices DRM driver identification and capabilities DRM master, magic and authentication DRM memory management DRM KMS dumb buffer API DRM FourCCs and modifiers DRM KMS resources probing DRM KMS modes DRM KMS framebuffer management DRM KMS legacy configuration and page flipping DRM event notification DRM KMS object properties DRM KMS atomic DRM render DRM Prime zero-copy memory sharing (dma-buf) DRM sync object fencing DRM debug and documentation <p><i>An exhaustive presentation of the DRM interface.</i></p>
Lab	Kernel Aspects	<ul style="list-style-type: none"> Linux TTY and virtual terminals DRM KMS mode-setting DRM KMS driver walkthrough DRM render driver walkthrough <p><i>Illustrating how kernel aspects work.</i></p>

Day 2 - Afternoon

Lecture	X Window Userspace Aspects	<ul style="list-style-type: none"> ▪ X11 protocol and architecture ▪ X11 protocol extensions ▪ Xorg architecture and acceleration ▪ Xorg drivers overview ▪ X11 and OpenGL acceleration: GLX and DRI2 ▪ Xorg usage, integration and configuration ▪ Major issues with X11 ▪ Xorg debug and documentation <p><i>Presenting all things related to X11 and Xorg.</i></p>
Lecture	Wayland Userspace Aspects	<ul style="list-style-type: none"> ▪ Wayland overview and paradigm ▪ Wayland protocol and architecture ▪ Wayland core protocol details ▪ Wayland extra protocols ▪ Wayland asynchronous interface ▪ Wayland OpenGL integration ▪ Wayland status and adoption ▪ Wayland debug and documentation <p><i>An in-depth presentation of Wayland.</i></p>
Lecture	Mesa 3D Userspace Aspects	<ul style="list-style-type: none"> ▪ Standardized 3D rendering APIs: OpenGL, OpenGL ES, EGL and Vulkan ▪ Mesa 3D overview ▪ Mesa 3D implementation highlights ▪ Mesa 3D internals: Gallium 3D ▪ Mesa 3D internals: intermediate representations ▪ Mesa 3D Generic Buffer Management (GBM) ▪ Mesa 3D hardware support status ▪ Mesa 3D versus proprietary implementations ▪ Mesa 3D hardware support: debug and documentation <p><i>Presenting 3D APIs and the Mesa 3D implementation.</i></p>
Lab	Userspace Aspects	<ul style="list-style-type: none"> ▪ Xorg code walkthrough ▪ Wayland compositor core walkthrough ▪ Wayland client examples ▪ Mesa code walk-through ▪ OpenGL and EGL examples <p><i>Illustrating userspace aspects, client and server implementations.</i></p>