libcamera: Unifying camera support on all Linux systems

LPC 2019
Lisbon, Portugal

Laurent Pinchart
laurent.pinchart@ideasonboard.com
libcamera

Cameras are complex devices that need heavy hardware image processing operations. Control of the processing is based on advanced algorithms that must run on a programmable processor. This has traditionally been implemented in a dedicated MCU in the camera, but in embedded devices algorithms have been moved to the main CPU to save cost. Blurring the boundary between camera devices and Linux often left the user with no other option than a vendor-specific closed-source solution.

To address this problem the Linux media community has very recently started collaboration with the industry to develop a camera stack that will be open-source-friendly while still protecting vendor core IP. Libcamera was born out of that collaboration and will offer modern camera support to Linux-based systems, including traditional Linux distributions, ChromeOS and Android.
In the beginning were simple pipelines...

Why?
... and they were simple to control, with a single API.
Then the world became complex ...
... and application developers were left suffering.
Solutions were proposed... but never implemented.
Then hope came back.
libcamera
libcamera provides a complete userspace camera stack.

The ‘mesa’ of the camera world.
Cameras Devices & Enumeration

**libcamera** enumerates cameras...

Diagram showing the relationship between Userspace and Kernel.
It supports multiple concurrent streams for the same camera...
Per-Frame Controls

\[ \text{\ldots and per-frame controls.} \]
Image Processing Algorithms (3A)

Image Processing Algorithm are loaded as external modules.
Adaptation layers offer backward compatibility with existing APIs...

Adaptation

libcamera
Adaptation

... and integrate libcamera with other operating systems.
Central to the stack is the Camera object, interfacing to device-specific pipeline handlers.
libcamera
- Provide camera support for all Linux systems
  Linux distros, Android, Chrome OS, ...

- Create an environment that fosters camera innovation
  vendor blobs and open-source

Goals
• Feedback from the Android community on the overall architecture
• Feedback from SoC vendors on the device-specific interfaces and device support in general
• Next development steps for libcamera to support the LEVEL 3 profile
• Contribution of libcamera to Project Treble and integration in AOSP
• Future of the Android Camera HAL API and feedback from libcamera team
• Future of the Linux kernel camera APIs

Discussions
The camera configuration is backed by device-specific validation from the pipeline handler.
The Camera Manager enumerates media devices and instantiates corresponding pipeline handlers.
The pipeline handlers create and register one or more cameras.
The pipeline handler interfaces with all kernel devices. It abstracts them and exposes video streams to upper layers.
Image Processing Algorithms (IPA) receive statistics from the hardware and compute optimal image parameters.
IPAs are separate modules that don’t access kernel devices directly. They only have access to their pipeline handler through the IPA API.
Closed-source IPAs are sandboxed in a separate process. They communicate with the pipeline handler through IPC.
The IPC is handled in core components, transparently for both the pipeline handler and the IPA.
Many helper classes ease the implementation of pipeline handlers for device vendors.
Native V4L2 applications are supported through a transparent compatibility layer.
A single Android camera HAL module implementation for all devices supported by libcamera.
libcamera
The camera implements a simple state machine.
The camera generates a configuration template from roles. The configuration can be modified, and shall be validated. The configuration is applied to the camera.

1. `configure()`
2. `generateConfiguration({
   Viewfinder,
   StillCapture
})`
3. `width=720`
4. `height=1280`
5. `validate()`
A request is created on the Camera, populated with a Buffer for each Stream, and queued for capture.
Buffer and request completion are notified separately.

Applications submit new requests to keep the streams going.

Request Completion
Contributing

libcamera is developed as a free software project and welcomes contributors. Whether you would like to help with coding, documentation, testing, proposing new features, or just discussing the project with the community, you can join our official public communication channels, or simply check out the code.

Mailing List

We use a public mailing list as our main means of communication. You can find subscription information and the messages archive on the libcamera-devel list information page.

IRC Channel

For informal and real time discussions, our IRC channel on Freenode is open to the public. Point your IRC client to #libcamera to say hello, or use the WebChat.

Source Code

libcamera is in early stages of development, and no releases are available yet. The source code is available from the project's git tree, hosted by LinuxTV.

$ git clone git://linuxtv.org/libcamera.git

Documentation

Project documentation is created using Sphinx. Source level documentation uses Doxygen. Please make sure to document all code during development.

Sphinx integration with Doxygen is planned, likely using Breathe and Exhale.

Submitting Patches
Obrigado