Leveraging Free and Open Source Software in a Production Environment

Matt Porter

EA Solutions, Inc.

Embedded Alley

Introduction

- The value in leveraging Free and Open Source Software (FOSS) is obvious
  - Save time
    - Saves money and we get to market faster
- If I use the GNU toolchain, Linux kernel, and a standard rootfs…what else is there?
  - Much..much..more
  - Toughest part of leveraging FOSS is knowing that useful code already exists for your project
  - There is no substitute for good Google research skills

Case Study

- The best way to see how to maximize FOSS usage is to use an example
- We will use a real product case study
  - Product steps
    - Define application requirements
    - Break down requirements to software components
    - Identify software components fully or partially available as FOSS
    - Integrate/extend FOSS components with value add software to meet application requirements

Digital Photo Frame (DPF)

- Digital Photo Frame (DPF)
  - Typical current embedded Linux application
  - Illustrates use of a varied set of FOSS components
  - Requirements are clear and concise
  - Many people are familiar with DPF device functionality

Embedded Alley

DPF platform

- Hardware assumptions
  - ARM SoC
    - DSP
    - PCM audio playback
    - LCD controller w/ 16-bit color support
    - MMC/SD controller
    - NAND controller
  - 800x600 LCD
  - Small number of navigation buttons
  - MMC/SD slot
  - NAND flash
  - Speakers

DPF Requirements

- DPF shall support display to the LCD
- DPF shall detect SD card insertion
  - Notify DPF application of SD card presence
  - DPF application will catalog photo files on SD card
- DPF shall provide a modern 3D GUI and transitions
  - Menu navigation via buttons
  - Configuration for slideshows and types of transitions to use via menus

DPF Requirements

- DPF shall support audio playback from speakers
  - MP3 audio playback
  - Playlist handling
  - ID3 tag display
- DPF shall support JPEG resize and rotation
  - Shall handle arbitrary size JPEGs up to 1600x1200
  - Dithering support for 16-bit color display
  - Display on 800x600 LCD

DPF software components

- Based on the previous requirements we have the following component breakdown
  - Firmware
  - OS kernel
  - I/O drivers
  - Base userspace framework/applications
  - Media event handler
  - Jpeg library (run on ARM or DSP)
  - MP3 and supporting audio libraries
  - OpenGL ES library for 3D interface
  - Main DPF application

DPF FOSS components

- First, we cover the obvious FOSS components
  - Firmware
    - U-Boot, and others
  - OS kernel
    - Linux, of course!
  - I/O drivers
    - Leverage SD/MMC, FB, Input, ALSA subsystems
  - Base userspace support/applications
    - Busybox, OE build system

Media event handler

- Udev
  - Receives events from kernel
    - SD card insertion/removal
  - Creates device nodes
  - Uses standard udev rule set
    - Optionally use prepopulate option for performance
    - Optionally use custom rules for local unique naming
  - Sends the SD card event over a socket to the HAL daemon
    - Custom rule

Media event handler

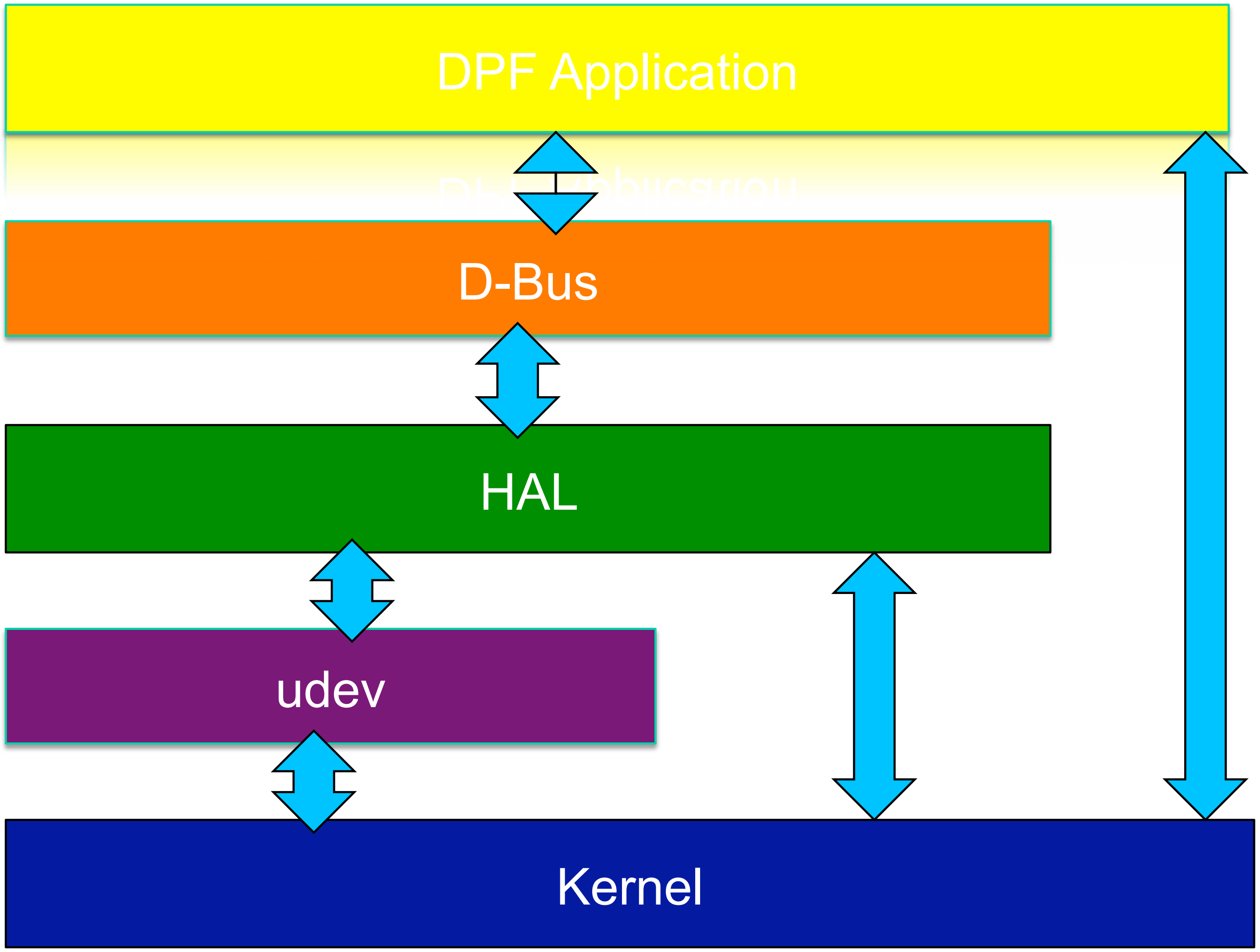
- HAL
  - Hardware Abstraction Layer
  - Daemon to handle hardware interaction
    - Maintains a database of known device objects
    - Received uevents are processed according to device information files.
    - Add-Ons provide specific functionality for devices
    - Storage Add-On polls for SD changes
    - SD insert/removal messages are sent to the DPF application
  - D-Bus is the API by which messages are delivered to the DPF application

Embedded Alley

Media event handler

- D-Bus
  - IPC framework
    - Implements a system-wide message bus
  - Applications can communicate with each other over the message bus.
    - Communication is asynchronous
  - HAL<->DPF communication takes place over D-Bus
  - DPF application subscribes to HAL SD
    - SD change events are delivered asynchronously from the HAL daemon to the DPF application on the message bus
    - Mount/umount can also be controlled via HAL

Media event handler



Embedded Alley

JPEG library

- Libjpeg
  - Handles JPEG decode
- Jpegtran
  - Resize and rotation support
- FIM (Fbi IMproved)
  - Dithering support

MP3 and supporting libraries

- Libmad
  - Run on ARM
  - Decode MP3 audio for playback
- Libid3
  - Handle id3 tags for display
- Libm3u
  - Handle media playlists

DSP acceleration

- What can be leveraged to accelerate JPEG and MP3 processing on the DSP?
- Need a DSP bridge
  - Openomap.org
  - In some cases, requirements might dictate a different approach
    - Use libelf to process ELF DSP binaries
    - Allows for pre-runtime patching of symbols
    - Allows for cross calls from DSP to ARM
- Leverage general purpose libraries
  - Libjpeg, jpegtran, FIM, and libmad can be ported to run portions on a DSP

OpenGL ES library

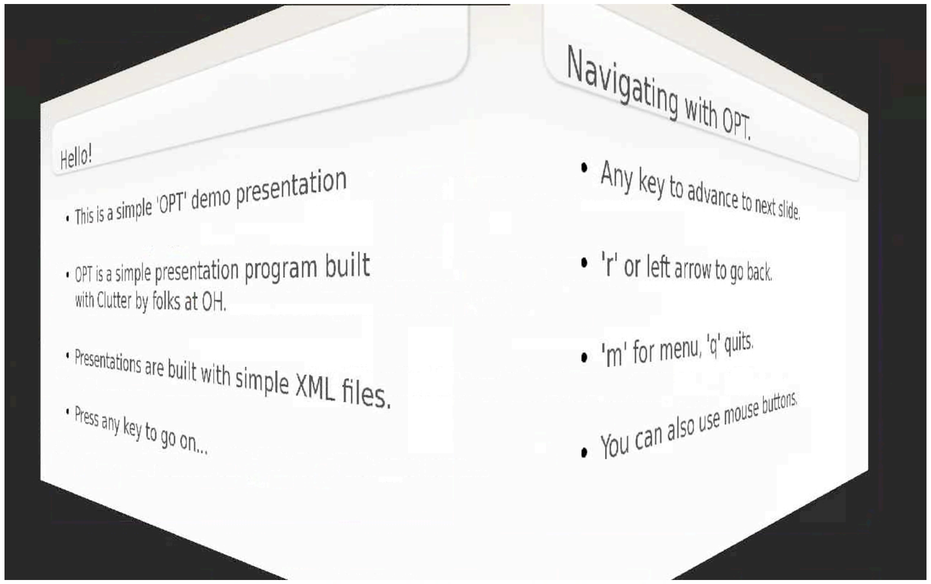
- Vincent
  - OpenGL ES 1.1 compliant implementation
  - Compatible with GLU|ES GLUT|ES supporting libraries
  - Nokia branch ported to Linux/X11, easily modified for FB operation
  - Can be extended for hardware accelerated color and floating/fixed pointed conversions
  - Can be extended for GPU acceleration

OpenGL ES library

- A complete GUI can be implemented in low-level OpenGL ES
  - Shaded/textured widgets
  - Font rendering to textures using the freetype library
  - Enables 3D “desktop look” for interface
- 3D photo transitions are possible
  - Photos are loaded to textures
  - Transitions managed as polygon animation and camera view management

OpenGL ES library

- Higher level libraries can be leveraged
  - Clutter
    - OpenGL ES backend due to “COGL” abstraction
    - Provides high level interface building tools
      - Actors (Widgets)
      - Stages (Windows)
      - Eases creation of more complex interfaces over raw OpenGL ES



Hello!

- This is a simple 'OPT' demo presentation
- OPT is a simple presentation program built with Clutter by folks at OH.
- Presentations are built with simple XML files.
- Press any key to go on...

Navigating with OPT.

- Any key to advance to next slide.
- 'r' or left arrow to go back.
- 'm' for menu, 'q' quits.
- You can also use mouse buttons.

Embedded Alley

DPF application

- The main DPF application integrates all of the FOSS components
  - Manages media events
  - Uses the JPEG library to decode and render photos
  - Handles Linux input events and drives OpenGL ES based GUI
  - Manages user-selected configuration
  - Displays photo slideshow using selected transitions

Conclusions

- Good research is the key to maximizing FOSS use
- Many components will require extensions and/or optimization
- Smart use of FOSS where possible will save time, money, and speed product to market

Q&A

- Questions?

Embedded Alley