OpenPrinting®

cups-filters, CUPS Snap, Printer Applications, Driverless Scanning, ...

Till Kamppeter – OpenPrinting

5 May 2021

20 Years of OpenPrinting!

- In the **beginning of 2000** I discovered CUPS, as sys admin I installed it and improved printing a lot!
- Thanks, Michael Sweet!
- I wrote the first free software print dialog with easy option settings, got invited to LinuxTag, then to MandrakeSoft
- In **August 2000** I started at MandrakeSoft in Paris to move the distro from LPD to CUPS
- In **fall 2001** I was one of the founders of OpenPrinting, with Ira, Michael, and others on the VA Linux Printing Summit in San Francisco
- With my, Mike’s, and other’s upstream work the other distros also switched to CUPS.   R. I. P., LPD/LPRng.
- In **2006** I organized the first OpenPrinting Summit and we continued annually up to now
- From **2008** on annual participation in the Google Summer of Code (as the Linux Foumdation, but most projects are OpenPrinting)
- From **2016** on Aveek Basu is our program manager, especially active on building the community
- Now after **20 years**, next major architecture change: **Driverless printing and scanning, all “just works” out-of-the-(sand)box, all-Snap distros, ...**

cups-filters: Introduction

- **cups-filters takes up everything from CUPS which Mac OS X does not need** (CUPS 1.6.x)
  - Started end of 2011 by OpenPrinting, **overtaking most of CUPS' filters**
  - Switched filters over from PostScript-centric to **PDF-centric workflow**
  - **cups-browsed** introduced end of 2012, to introduce **browsing of DNS-SD-advertised remote CUPS queues**, as CUPS dropped its own broadcasting/browsing
  - **10** years of **further development** added things like driverless printing support, clustering, support for Printer Applications, IPP standards, ...

cups-filters Development: Filters

- **New features:**
  - **Filter functions:** Converted the CUPS filters into library functions (in libcupsfilters)
    - Called by **Printer Applications** without executable call overhead
    - Conserve years of coding work in filters
    - Control both via **IPP Job/printer attributes** and **PPDs**
    - All have **same call scheme**
    - **Auxiliary functions** to chain them, connect with file descriptors, no need to fork() manually
    - **CUPS filters are wrappers** around filter functions

cups-filters Development: Filters

- **New features:**
  - **pclmtoraster**: Convert **raster-only PDFs** (PCLm, also scanner output) into **raster image files**, for higher print output quality and converting scanned files. Vikrant Malik, GSoC 2020.
- **Planned**:
  - **Wrapper filter function** to call conventional CUPS filter/backend (e. g. for retro-fitting proprietary driver)
  - Let filter functions check **color space/depth of input** file and check available **color depths/spaces in printer IPP attributes/PPD**, **print-quality**, and **print-content-optimize** setting to select most suitable quality

cups-filters Development: libppd

- **New PostScript handling library, libppd:**
  - To be used for **retro-fitting classic PPD/filter drivers** into **Printer Applications**
  - Overtaken **all PPD-related functions from libcups**
    - Will get removed from libcups soon
    - Also added CUPS-private functions to API
    - **PPD collection handling** functions (based on the code of **cups-driverd** in CUPS) to list and select PPDs in Printer Applications
    - Minimum coding effort for this obsolete data format
  - Separate from libcupsfilters, as only for retro-fit
  - **Only for retro-fit, do not use for new drivers!**

cups-filters Development: driverless utility

- Mostly done by Nidhi Jain, LFMP 2020
- Added **IPP Fax Out** support
  - Shows Fax PPDs in printer setup tools. selecting them creates a fax queue
  - Support for Fax on most modern MF devices under Linux for the first time
  - Fax obsolete in most, but not all countries
- **IPPS support** (show only IPPS URI if available)
- Use of CUPS' **DNS-SD-service-name-based** URIs
  - IP- and port-independent (good for local services)
  - --std-ipp-uris option to show standard IPP URIs

cups-filters Development: cups-browsed

- **Main development goal: Speed, performance, scalability**
- **New features:**
  - **Multi-threading**: Handling discovered devices and creating local queues in separate threads, GSoC 2020 project of Mohit Moran
  - Longer timeout for cups-browsed to tell the CUPS backend which cluster member gets the job
  - **Do not remove created queues on shutdown**: Avoid GUI notification clutter when queues are created again on restart (configurable)
  - **Faster removal of left-over queues** from previous session

cups-filters Development: cups-browsed

- **Planned**
  - **Avahi browsing/resolving optimization**: Get rid of unneeded, time-consuming resolving
  - Separate cups-browsed from cups-filters, into **own OpenPrinting GitHub project**
  - Separate cups-browsed from CUPS Snap into **own Snap**
  - Turn cups-browsed into a **Printer Application**

cups-filters Development: 1.x → 2.x

- **License change** to **Apache 2.0 + (L)GPL2 exception (**approved by contributors)
- **All filters → Filter functions** (mostly done, GSOC)
- **Filter functions** should work **IPP-only, without PPD** (TODO GSoC 2021)
- **Filter functions** should safely run in parallel (TODO)
- Filter function to **chain filters** (DONE)
- Filter function to **call CUPS filters/backends** (TODO)
- Wrappers for **filter functions as CUPS filter** (DONE)
- **libppd** for retro-fitting classic drivers (DONE)
- Function to **clean/normalize make/model** from device IDs, for Printer Applications select driver (DONE)
- **pclmtoraster**: Convert raster-PDFs to raster (DONE)

cups-filters Development: 1.x → 2.x

- **cups-browsed multi-threading** (DONE)
- **cups-browsed** in **separate** project/Snap (TODO)
- Turn **cups-browsed** into **Printer Application** (TODO)
- Options for the **`./configure`** script for partial builds: No cups-browsed, no libppd/PPD support, no libqpdf, raster-only printing/scanning, … to **allow Snaps build only the part of cups-filters which they actually need** (TODO)

We have agreed on **not to rename cups-filters**. It will only get a **new generation number (versions 2.x.y)**. cups-browsed will perhaps get spun out into its own project.

CUPS in a Snap

- A **Snap containing CUPS**, cups-filters, cups-browsed, Ghostscript, QPDF → **Complete CUPS printing stack**
- **No support for classic drivers**, as filters and PPDs cannot get dropped into Snap’s file system → **Printer Applications**
- Sorting out all the problems with Canonical’s Snap gurus on the snapcraft.io forum (see all links in README.md)
- Components **always up-to-date**, independent of release cycles: CUPS 2.4.x, cups-filters 1.28.8, Ghostscript 9.54.0, QPDF 10.3.1
- Available in **Snap Store** “cups”: https://snapcraft.io/cups

CUPS in a Snap
Properties

- **Three run modes**:
  - **Stand-alone**: Snap’s CUPS is the only CUPS on the system, no classic CUPS present
  - **Proxy**: Classic CUPS present, Snap’s CUPS clones the queues, is firewall for the classic CUPS
  - **Parallel**: Classic CUPS present, Snap’s CUPS runs as second, indpendent CUPS (for development only)
- CUPS **always listens on Snap’s domain socket**, in stand-alone mode also on the standard domain socket and port 631 for unsnapped clients
- To not need to create system users and groups use snapd’s “snap_daemon” for “lp” user and “adm” for “lpadmin” group
- Adapted to Snap environment via cups-files.conf and file permissions, no patches

CUPS in a Snap Properties

- All **System-V-** and **Berkeley-style command line tools**, also **special tools** cupsfilter, driverless, ippfind, ipptool, ippeveprinter, ippproxy
- **cups-browsed** included, always attaching to the Snap’s CUPS

CUPS in a Snap
Security concept

- **Snaps are usually completely confined** and can communicate only through **defined interface connections**
- **Everyone can upload Snaps** to the Snap Store **but**
  - On the user's system only "**safe**" interfaces of downloaded Snaps **connect automatically**
  - "**Dangerous**" interfaces need to get **connected manually** after Snap install (if they do not have auto-connect permission from the Snap Store team)
  - **Unconfined** ("classic") need **permission** of the Snap Store team
- For **using CUPS from Snaps** there are two interfaces:
  - "**cups**": For user application Snaps which print (**safe**)
  - "**cups-control**": Admin access to cupsd (**dangerous**)

CUPS in a Snap
Security concept

- Most **Snap interfaces** are defined only by **AppArmor rules**, but
- “cups” vs. “cups-control” → **Snap Mediation**
  - If cupsd receives an **administrative request** it accepts it only if
    - The client is **no Snap** or a **classically confined Snap**
    - The client connects via “**cups-control**”
- User’s system **usually has classic CUPS**, not CUPS Snap → **No Snap Mediation**, therefore
  - “**cups**” interface only connects to **Snap’s domain socket**
  - Application Snap installation **force-installs CUPS Snap**
  - **CUPS Snap** in proxy (**firewall**) mode and mediates requests
  - User stays with **his queues** and (often proprietary) **drivers**

CUPS in a Snap as a distro’s CUPS

- **Complete the security concept** on the snapd side
- **Retro-fit all drivers** which usually come with distros into **Printer Applications**
- **GUI tools**, printer setup tool and print dialog adapted to the CUPS-Snap/Printer(Scanner)-Application architecture
- Feature request to Snap Store: **Search Snaps by hardware signature**, to easily install driver Snaps

Printer Applications

- **PAPPL** got standard framework
  - PAPPL provides **everything required in a library**
  - **Driver developer** only has to do the **printer-specific parts**
  - **Tutorial** for manufacturers/driver developers written in GsoD 2020
- Planned: **Printer driver retro-fit library**
  - Spin out from **PostScript Printer Application**
    - PPD handling: Listing, filtering, selecting, options, ...
    - Calling filter functions
  - Call **external CUPS filters/backends**

Printer Applications

- **Convert**:
  - Foomatic/Ghostscript
  - Gutenprint
  - HPLIP
  - SpliX
  - Foo2zjs
  - ...
- For **unmaintained drivers** wrap filters and PPDs into Printer Application
- **Native Printer Applications** for **maintained drivers**
- Wrapping **proprietary** drivers with **chroot**, **layout**, ...

Driverless Scanning

- **3 Standards**
  - **IPP Scan**, open PWG standard
  - **eSCL**, proprietary, from HP, specs published by Mopria
  - **WSD**, from Microsoft and W3C
- All are mainly intended for **multi-function printers**
- **eSCL** and **WSD** one **already available** in **AirScan** devices
- **2 SANE drivers** for eSCL: “escl” from Thierry Hucahrd and “airscan” from Alexander Pevzner, both in most distros
- Alexander has added **WSD support** and will add **IPP Scan** if needed/required
- **eSCL** also works via **IPP-over-USB** (ipp-usb)

Sandboxed Scanner Drivers

- **Current situation: SANE**
  - Scanner driver (**SANE backend**) is shared library
  - Scanning app (**SANE frontend**) links backends dynamically
  - **To add a driver it needs to be dropped in backend dir** => not good for sandboxed packaging
- **New scanning environment: eSCL/IPP Scan driverless**
  - Scanner drivers in **Scanner Applications, emulating driverless scanner**
  - Scanning app is **eSCL/IPP Scan client**
  - Legacy: App uses sane-airscan SANE backend, SANE drivers enclosed in legacy Scanner Application
  - Michael Sweet posted **scan support roadmap for PAPPL**

IPP-over-USB: ipp-usb

- **ippusbxd simple TCP ↔ USB packet relay**, leaves packets in USB buffer on client closing connection, packets re-appear in next connection messing it up → **Creation of alternative ipp-usb** by Alexander Pevzner
- Ipp-usb written in **Go**, as Go has **sophisticated HTTP library** to read out buffer on closed connection
- Ipp-usb **works perfectly**, esp. web admin interface
- **Chrome OS not accepting software in Go** due to high memory footprint → Own approach in Rust
- **ippusbxd** development **discontinued**
- **eSCL scanning** and **IPP Fax Out** work with ipp-usb

Printing GUIs
What do we need?

- **Print dialog**: We need to get **CPDB** into GTK and Qt
- Printer Setup Tool
  - **Main Window**
    - List all IPP services as reported by DNS-SD, list Printer Applications and their queues in a group, no duplicates for IPv4/IPv6, IPPS, interfaces
    - Buttons for web interface, add new queue, show jobs …
- **Add Printer Wizzard**
  - List of discovered non-driverless USB/network printers, click button to see list of Printer Applications supporting the printer, installed ones and available in Snap Store
  - Buttons to setup printer with selected Printer Application and to install Printer Application from Snap Store

Questions / Comments