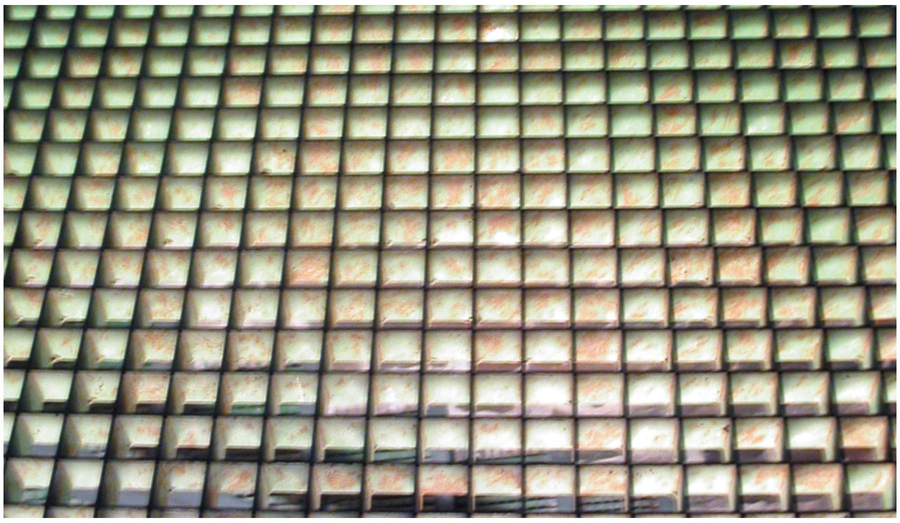
Linux Printing: Ghostscript, Foomatic, and Gimp-Print

Perfect Patterns



Under Linux, programs send PostScript-formatted documents to printers. For most printer models, Ghostscript provides a driver to convert the document into a format that the printer understands. Thanks to external helpers, this system is becoming more and more flexible. BY TILL KAMPPETER, KURT PFEIFLE

Printer drivers for Linux (and other Unix-style operating systems) are actually filters that convert an input format – typically Postscript – into a format that the printer can understand. Thus, in the Unix/Linux environment, the term *printer driver* always refers to the filter and not to the printer port driver, which resides within the kernel and supports communications with the printer via the parallel port, USB port, or network card. In the Linux world, Ghostscript is the #1 program for converting Postscript into dot matrix format. Ghostscript is a full-featured Postscript interpreter and Raster Image Processor (RIP). The Ghostscript interpreter is at the center of an ecosystem of components that have revolutionized the process of printing in Linux. In this article, you'll learn how Ghostscript works and how it interacts with other elements in the Linux printing infrastructure, such as CUPS, Gimp-Print, PPDs, and the great Perl wrapper, foomatic-rip. We'll end with a practical look at some little-known foomatic-rip options and functions.

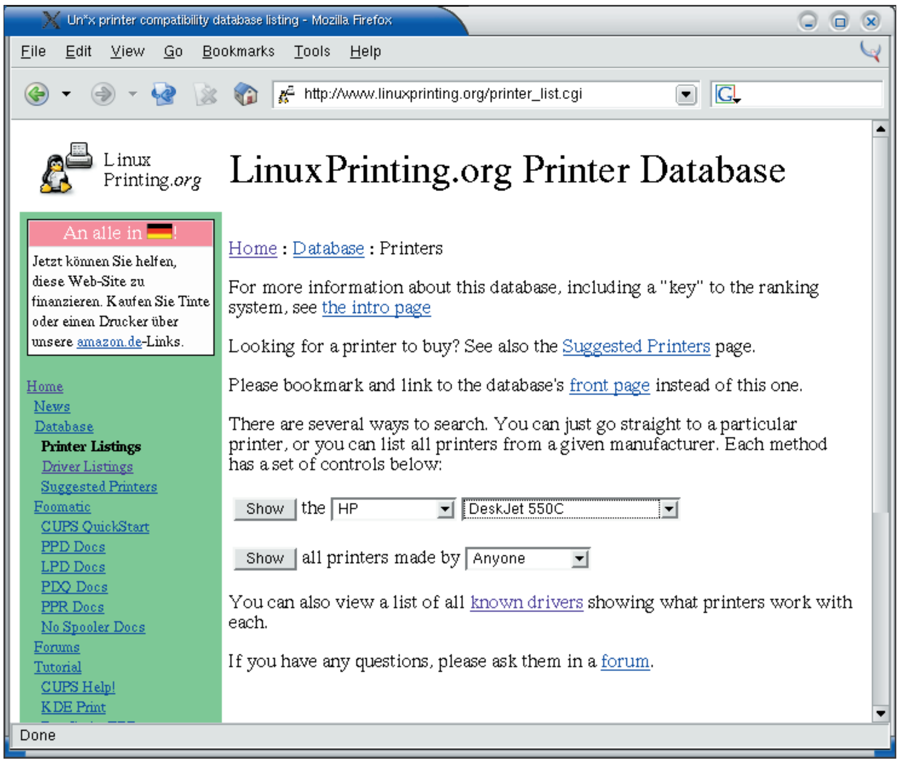
Postscript and Ghostscript

A Postscript program is a file that provides input for a Postscript interpreter such as Ghostscript. The output of the interpreter typically consists of a dot matrix, which the human eye will perceive as images, text, or lines. When Ghostscript runs the Postscript program that is passed to it, it first renders the pages to be printed, one after another, in the page memory. The page memory is a generic bitmap or raster image. The final resolution, size, and color depth depend on user input and the output format. After rendering a complete page, a driver converts the page into the final output format and sends it off to the output device. The resulting data can be the format required by the printer, a generic raster image (TIFF, PNG etc.), a vector format such as PDF, or even an X11 screen display. Postscript viewers such as GhostView do nothing but convert the Postscript file to a pixel image suitable for on-screen display.

After Ghostscript has processed the Postscript program and produced output in bitmap form, a printer-specific driver must transform this bitmap into a format compatible with the printer. Several methods exist for interfacing the driver with Ghostscript. The oldest Ghostscript driver design calls for statically linking printer drivers to a single monolithic block of executable code. Statically linked drivers have the benefit of installing the full range of Ghostscript drivers in a single step, without the need for modifying the printing environment later.

The disadvantage of statically linked drivers is that adding new drivers, bugfixes, or security updates to the existing driver package is quite complex. Changing the driver means that the user or package maintainer has to patch Ghostscript and re-compile the whole beast just to add a tiny piece of driver code. Some drivers will only install out of the box with a specific Ghostscript version. Additionally, there are a number of complicated conditions that you need to be aware of. For instance, the driver license has to be compatible with your Ghostscript version if you want to distribute Ghostscript with the driver. ESP Ghostscript includes the full range of free, statically linked drivers, as listed at Linuxprinting.org [1] (see Figure 1).

Modern driver architectures modularize these printing components. This modular design avoids some of the problems associated with statically linked drivers and gives users more freedom to add new drivers without having to modify Ghostscript. Several families of Ghostscript-compatible drivers have emerged, including the Omni drivers [2], authored by IBM; the HPIJS family [3], where the "HP" stands for Hewlett-Packard; and Gimp-Print, which comes from a pure open source and free software community background. All three families load dynamically at runtime, and all communicate with Ghostscript as stand-alone programs. The remainder of this article focuses on the Gimp-Print driver system, however, it is worth noting that the Omni and PHIJS families are also free software projects. IBM makes the Omni drivers available under the



Un*x printer compatibility database listing - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.linuxprinting.org/printer_list.cgi

Linux Printing.org

LinuxPrinting.org Printer Database

An alle in 🇩🇪!

Jetzt können Sie helfen, diese Web-Site zu finanzieren. Kaufen Sie Tinte oder einen Drucker über unsere amazon.de-Links.

Home
News
Database
Printer Listings
Driver Listings
Suggested Printers
Foomatic
CUPS QuickStart
PPD Docs
LPD Docs
PDQ Docs
PPR Docs
No Spooler Docs
Forums
Tutorial
CUPS Help!
KDE Print

Home : Database : Printers

For more information about this database, including a "key" to the ranking system, see the intro page

Looking for a printer to buy? See also the Suggested Printers page.

Please bookmark and link to the database's front page instead of this one.

There are several ways to search. You can just go straight to a particular printer, or you can list all printers from a given manufacturer. Each method has a set of controls below:

Show the HP DeskJet 550C

Show all printers made by Anyone

You can also view a list of all known drivers showing what printers work with each.

If you have any questions, please ask them in a forum.

Done

Figure 1: LinuxPrinting.org provides drivers and other resources.

Lesser General Public License (LGPL). Hewlett-Packard, the first manufacturer to place its drivers under a free software license, originally prohibited the use of the drivers with non-HP printers but eventually released the HPIJS code under the BSD license.

CUPS Raster Driver

The CUPS designers wanted to minimize the expense of creating a Linux print driver, so they devised a system that lets the maintainer add a new driver without patching and recompiling Ghostscript. The flexible CUPS architecture offers a two-level approach that separates Ghostscript from the raster image driver. Ghostscript outputs a bitmap, and the driver converts the bitmap to the target format (see Figure 2). This approach is similar to the traditional filter concept, however, CUPS defines a new bitmap format instead of using an existing format. CUPS-raster is capable of storing and transporting type-specific printer information in the metadata of a page bitmap. It has functions for RGG, CMYK, and color management and will be capable of supporting future extensions. Its data format is completely open. The developers registered CUPS-raster as a MIME type *application/vnd.cups-raster* with the (IANA) Internet Assigning Numbers Authority).

For driver developers, CUPS-Raster takes the headaches out of handling non-Postscript devices. It gives developers a standard format that is extremely easy to use. No need to worry about creating CUPS-raster images because CUPS will take care of that itself. Developers just need to write a simple raster image driver, which can even be proprietary, to turn a CUPS-raster to a manufacturer-specific and type-specific device raster.

The CUPS package comes with a few very generic CUPS raster drivers. Vendors such as ESP Print Pro [7] and Turboprint [8] have commercial raster drivers for CUPS. But the biggest package of free CUPS-raster drivers is the vendor-independent Gimp-Print system [9].

Gimp-Print

Gimp-Print [9] is the most important independent driver project. Launched by CUPS author Mike Sweet, and with support for six-ink printing added by amateur photographer Robert Krawitz, who wanted premium quality photo printing on his Epson Stylus Photo EX, this driver package has seen notable advances. Today, it supports more than 540 printers, mainly inkjets. Maximum printing quality is a matter of course for nearly all Epson inkjet printers, and Gimp-Print supports older Canon models, HP inkjets, some Lexmark and laser printers that speak PCL 4, PCL 5, and PCL 5e. There are historical reasons for the driver name. Its roots go back to the first printing plug-in for the GNU image manipulation program, Gimp. Later developments led to it becoming totally independent from Gimp (although it can still give Gimp direct printer access).

There are three different incarnations of the Gimp-Print driver. Depending on the configure setting, *make* will generate one, two, or all three of them:

- The print plug-in, which provides the image printing dialog for printing directly in Gimp.
- An IJS server, which processes printing data from arbitrary applications together with Ghostscript (*-sDEVICE=ijs*).
- A CUPS-raster driver, which converts the output from *pstoraster* (mimetype

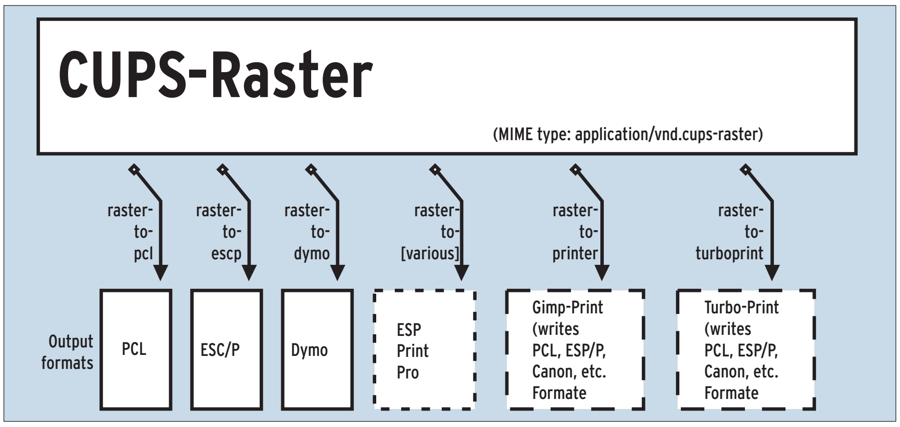


Figure 2: The CUPS-raster format is suitable for many open or proprietary printer-specific formats. Gimp-Print or Turboprint, among others, use it to create printer-specific raster image formats.

application/vnd.cups-raster) to the target format.

Until recently, there used to be an option for statically linking the driver in Ghostscript (*-sDEVICEstp*). But IJS technology has made this redundant and the feature is no longer supported. Modern Linux distributions know how to set up Gimp-Print, and the package is now included by all major distributions. Users can typically rely on the maintainers to build and pre-configure the package to reflect their system. Gimp-Print has an incredible number of settings for printing quality: paper type, gamma value, color density, color balance, brightness, contrast, sliders for the individual color channels (Cyan, Magenta, Yellow, Black). This allows the driver to achieve photo quality that rivals or exceeds the results that Windows drivers provide – at least with Epson printers. Gimp-Print became popular with many Mac OS X users at an early stage; Apple “Panther” (Mac OS X, Version 10.3) adopted Gimp-Print, and Apple includes Gimp-Print on the installation CDs.

PPDs Know Devices

To leverage the full functionality of a printer, you need to add printer-specific commands to the Postscript datastream (for example, to change paper for the second page, or staple a set of pages). Unfortunately, neither the applications, nor Postscript, which is device-independent by design, know these printer-specific settings. Postscript Printer Description (PPD) files provide the answer to this dilemma by specifying user-definable options. The PPD file has Postscript code for each job parameter. Additionally, PPDs are aware of a device’s capabilities (for example monochrome or color printing, or autodetect ID information). CUPS, *foomatic-rip* (which you’ll learn about in the next section, and a few applications such as OpenOffice.org parse PPDs, using the information stored in the files to add device-specific code to the Postscript job stream. PPD-capabilities let CUPS and *foomatic-rip* provide full Linux support to any Postscript printer.

Foomatic-rip

foomatic-rip is an enormous Perl script (more than 5000 lines) that provides a wrapper for Ghostscript (see Figure 3). Its job is to generate the best possible input syntax for Ghostscript. Ghostscript uses this input to create the raster image file. Of course, the best achievable quality depends on your printer.

foomatic-rip parses a PPD file to discover what capabilities the printer provides. *foomatic-rip* uses PPDs for any printer type, not just for Postscript printers, although the PPD design originally envisaged Postscript only. After creating the raster output, *foomatic-rip* passes the output to the printing system back-end. Considering how quickly manufacturers put new printers on the market, it would be too much trouble to create a new PPD manually for each device. To remove the need to do so, the foomatic system provides an XML-based database called *foomatic-db* to generate PPDs. Anyone can install the system, but there is no need to do so, as LinuxPrinting.org offers PPD downloads. Distributors, such as SuSE for example, also install PPDs for supported printers by default.

Using CUPS Drivers with Other Spoolers

CUPS has its own two-level printer driver design. It first converts the print job to a special CUPS-raster format. Then, the CUPS-raster driver converts the data to the target format for the printer (see Figure 4). *foomatic-rip* allows you to use these drivers with other spoolers, and this can be very useful if a manufacturer only provides a CUPS driver for Linux or Unix. You will need to install ESP Ghostscript and *foomatic-rip* along with the driver. To do so, you may need to install your distro’s *libcups* package. This library is just a minor CUPS component that, among other things, provides functions for generating and reading CUPS-raster formats. If you are building the driver and/or ESP Ghostscript yourself, you will additionally need your distribution’s *libcups-devel*, or *libcups-dev* pacages. Type *gs -h | grep cups* to check if Ghostscript really has a CUPS-raster format driver. The output should contain the word *cups*. Now set up your printer, as described at LinuxPrinting.org, to support LPD/ LPRng, PDQ, PPR, or without a spooler. But do not use a LinuxPrinting.org PPD (and of course, avoid PPDs for genuine Postscript printers!) Instead use a PPD that was designed as a native CUPS driver. *foomatic-rip* will handle the two level CUPS rendering process for you, not your CUPS spooler, and this means typing *gs -sDEVICE= cups...* to create CUPS-raster and then adding it to your *cupsFilter:* PPD.

If things don’t work out: *foomatic-rip* writes a log file with detailed information. The file is typically located in */tmp/foomatic-rip.log*. Debug mode saves the last Postscript file to be processed as */tmp/foomatic-rip.ps* to

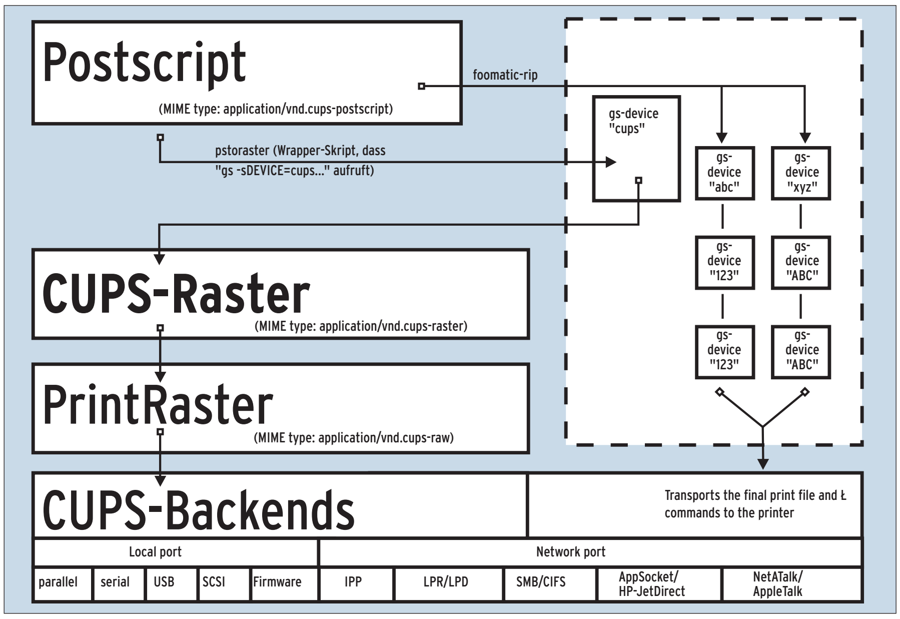


Figure 3: ***foomatic-rip*** **is a wrapper script that controls the generation of a raster image file from Postscript. It can use a variety of Ghostscript devices to do this.**

support troubleshooting. You need to enable debugging by changing a line in the first section of the Perl code for *foomatic-rip*. The line should read *debug = 1;* (instead of *0*).

The *foomatic-rip* source code includes detailed commentary. Even if you have been wary of touching Perl so far, just check it out! The comments give you a lot of information on printing with Postscript and non-Postscript files.

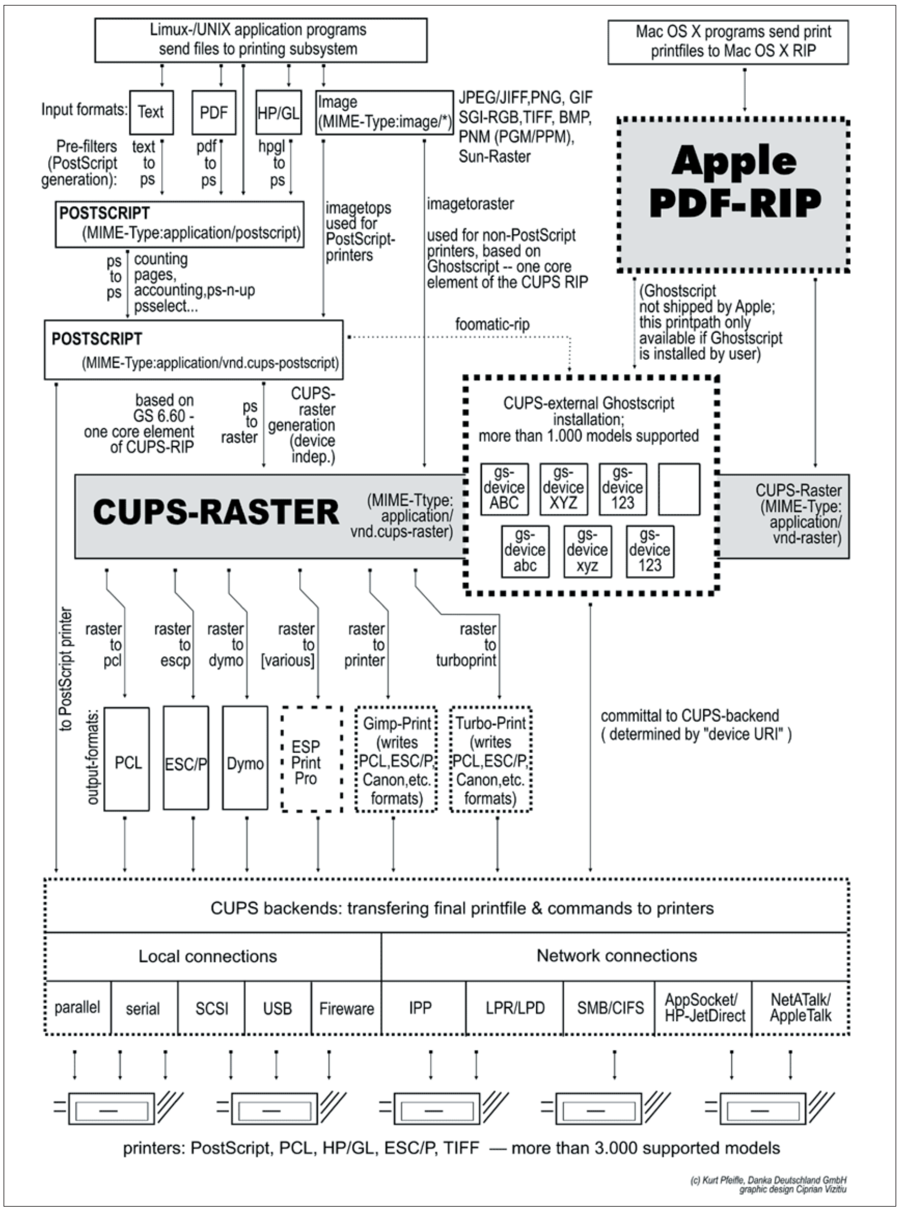
Tricks with *foomatic-rip*

foomatic-rip has a number of special functions that are very useful, although many people are not aware of them. If you do not know whether your printer uses foomatic-rip, you can try the following to find out:

```
$ grep foomatic-rip ➥
   /etc/cups/ppd/*
/etc/cups/ppd/digimaster.ppd:➥
   *cupsFilter: U
"application/➥
   vnd.cups-postscript 0 U
"foomatic-rip"
```

The grep result shows the **cupsFilter: "application/vnd.cups-postscript 0 foomatic-rip"* line in *digimaster.ppd*, which tells us that the *digimaster* really does use *foomatic-rip*.

Most users are unaware of the capabilities their printers provide and that foomatic-rip supports. Graphical tools such as *kprinter*, *xpp*, or *gtklp* will display the options, but the console does not have this kind of support. Help is at



Linux-/UNIX application programs send files to printing subsystem
Mac OS X programs send print printfiles to Mac OS X RIP
Input formats:
Text
PDF
HP/GL
Image (MIME-Type:image/*)
JPEG/JIFF,PNG, GIF SGI-RGB,TIFF, BMP, PNM (PGM/PPM), Sun-Raster
Apple PDF-RIP
Pre-filters (PostScript generation):
text to ps
pdf to ps
hpgl to ps
POSTSCRIPT (MIME-Type:application/postscript)
imagetops used for PostScript-printers
imagetoraster
used for non-PostScript printers, based on Ghostscript -- one core element of the CUPS RIP
ps to ps
counting pages, accounting,ps-n-up psselect...
(Ghostscript not shipped by Apple; this printpath only available if Ghostscript is installed by user)
foomatic-rip
POSTSCRIPT (MIME-Type:application/vnd.cups-postscript)
based on GS 6.60 - one core element of CUPS-RIP
ps to raster
CUPS-raster generation (device indep.)
CUPS-external Ghostscript installation; more than 1.000 models supported
gs-device ABC
gs-device XYZ
gs-device 123
gs-device abc
gs-device xyz
gs-device 123
CUPS-RASTER
(MIME-Ttype: application/ vnd.cups-raster)
CUPS-Raster (MIME-Type: application/ vnd-raster)
to PostScript printer
raster to pcl
raster to escp
raster to dymo
raster to [various]
raster to printer
raster to turboprint
output-formats:
PCL
ESC/P
Dymo
ESP Print Pro
Gimp-Print (writes PCL,ESC/P, Canon,etc. formats)
Turbo-Print (writes PCL,ESC/P, Canon,etc. formats)
committal to CUPS-backend (determined by "device URI")
CUPS backends: transfering final printfile & commands to printers
Local connections
Network connections
parallel
serial
SCSI
USB
Fireware
IPP
LPR/LPD
SMB/CIFS
AppSocket/ HP-JetDirect
NetATalk/ AppleTalk
printers: PostScript, PCL, HP/GL, ESC/P, TIFF — more than 3.000 supported models
(c) Kurt Pfeifle, Danka Deutschland GmbH graphic design Ciprian Vizitiu

Figure 4: The CUPS printing environment.

Three versions of Ghostscript

Three different different versions of Ghostscript are currently available: AFPL Ghostscript [4], GNU/GPL Ghostscript [5], and ESP Ghostscript [6]. The AFPL Ghostscript variant is the most advanced. The current 8.13 version is the first to have critical elements such as color management and PDF version 1.4 support. AFPL Ghostscript is used by many commercial products, although it is free for personal use if you download the software yourself. AFPL Ghostscript is not free software, as the license (Aladdin Ghostscript Free Public License) prohibits commercial distribution of AFPL. Commercial Linux distributions are not allowed to include AFPL Ghostscript.
The manufacturers of AFPL Ghostscript always release older versions as GNU Ghostscript under the GPL (GNU General Public License) 12 months after the original release. The current version is 7.07. As bugs and security holes are inevitable in the production cycle of any software, packagers need to put in a lot of work after the GPL version has been released, fixing the GPL xyz.0 version with their own collection of hotfixes, re-compiling, and placing the results on their own distribution media.
ESP Postscript was launched about three years ago, when the CUPS (Common Unix Printing System) developers invited anyone who was interested to join in the common effort to create an up-to-date Ghostscript variant. The ESP version is based on GNU Postscript and the current version number is 7.07.1. Today, the project is hosted at Sourceforge, and the maintainers include developers from Mandrake, Debian, Suse, and Red Hat, but also people from software projects like Gimp-Print, Foomatic, Linuxprinting.org, or Omni. ESP Ghostscript has the biggest collection of printer, and other raster image drivers. The latest GNU Ghostscript version reports about 180 devices when you enter *gs -h*, in contrast to the 300 devices that ESP Ghostscript has. ESP Ghostscript also includes the CUPS raster image driver.
Insiders predict that ESP and GNU/GPL Ghostscript will merge sometime in the near future. This merger will free up more programming capacity and eliminate the inefficiency of maintaining separate and parallel open source Ghostscript projects.

Foomatic-rip, CUPS and Postscript Printers

If you set up a Postscript printer with the original manufacturer PPD file, it will not use *foomatic-rip*, and the options described earlier will not be available. Despite this, you can use the options without ditching the original PPD file. To do so, simply add the following lines to the PPD file for your printer (in »/etc/cups/ppd/printername.ppd«):

```
*cupsFilter:"application/vnd.➢
cups- U postscript 0 foomatic-rip"
```

This line should be added to the header area of the PPD file (but not before the first line, **PPD-Adobe*!). Then *killall -HUP cupsd* to tell CUPS to apply the changes. Now type *which foomatic-rip* to check whether your system recognizes *foomatic-rip*. If not, or if your version is too old, download the version from LinuxPrinting.org, make it executable, and store it in your *$PATH*. Then create a symbolic link:

```
ln -s `which foomatic-rip`U
/usr/lib/cups/filter/«.
```

This allows CUPS to find *foomatic-rip* in its own filter directory and use the driver.

hand. To print a list of options, stipulate the *docs* option with the *lpr* command (of course, this assumes that you use *foomatic-rip*). If you launch a print job with this option, the printer will output a list of options rather than the job itself. Assuming CUPS as the spooler, the command is as follows:

```
$ echo x | lpr -P ➢
  digimaster -o docs
```

And with LPRng as the spooler:

```
$ echo x | lpr -P ➢
    digimaster -Z docs
```

lpr expects to be passed a file as the print job, so you will either need to point to an existing, and readable, file on your disk, or to standard input. As the *docs* option tells *lpr* not to print the file, it does not really matter how you do this. Our example simply uses the *echo* command to send an *x* to the printer.

Read the page(s) that *-o docs* gives you, and keep them for future reference, or at least for the following sections. The options should be printed in cleartext, with an example for each option.

Most inkjets, and many laser printers, can use arbitrary paper sizes. However, this does not mean that the printer will automatically detect the paper size, so you need to let your printer driver have this information when you submit a job. The following CUPS command

```
$ lpr -p photoprinter -o U
"PageSize=➢
   Custom.20.7x23cm file.ps
```

tells the driver what size the user requires. *mm*, *pt* (Postscript units: 1 pt = 1/72 inch) and *in*, that is inches, are all valid units. LPRng uses the following command:

```
$ lpr -p photoprinter -Z U
"PageSize=➢
   Custom.20.7x23cm file.ps
```

One page of a document may require different settings than the remaining pages – this could be a long letter that starts off on letterhead paper and then uses normal sheets for the following pages, for example. If the printer uses *foomatic-rip*, you can restrict the optional settings to specific pages. To continue the example with the letterhead:

```
$ lpr -P laser -o 1:InputSlot=U
"Letterhead -"o InputSlot=➢
    Standard file
```

The first option says, use the letterhead paper tray for page 1. The second option says, use the standard tray for the remaining pages. The first option takes priority over the second, as it is more specific (it only applies to page 1). This puts the first page on the letterhead, and any other pages on normal paper. The following example prints a watermark on even pages and grayscales on the odd pages:

```
$ lpr -P color -o even:➢
   Watermark=U
"on -o odd:ColorMode=➢
   Gray file
```

The following line

```
$ lpr -P digimaster -o ➢
   1,6-10,15,U
"20-:MediaType=➢
   YellowPaper file
```

prints pages 1, 6, 7, 8, 9, 10, 15, 20 and any following pages on yellow paper, and it puts any other pages on the default paper for this printer. If your printer driver is not reacting as you expect and you do not have yellow paper in any of your paper trays, just experiment. Make sure that the option you want to try is in the list that you printed earlier. (Remember the *-o docs* trick?) ■

THE AUTHORS

Till Kamppeter is the maintainer of the LinuxPrinting.org project and works as a printing and digital imaging developer for Mandrakesoft in Paris, France. You can meet Till at many Open Source events, where he gives talks on his work and is available to discuss it at various booths.

Kurt Pfeifle writes documentation for various free software projects, (CUPS, LinuxPrinting.org, Samba, KDE/KDE Print) and works as a systems specialist and consultant for network printing at Danka Deutschland Holding GmbH, Germany. Last year, he supervised various migration projects (moving NT-based print server to Linux/CUPS/Samba).

INFO

[1] Linuxprinting.org: *http://www.linuxprinting.org/*
[2] Omni drivers by IBM: *http://www.ibm.com/linux/ltc/projects/omni/*
[3] Ink Jet Server, IJS: *http://www.linuxprinting.org/ijs/*
[4] Aladdin Ghostscript: *http://www.ghostscript.com/*
[5] GNU Ghostscript: *http://www.gnu.org/software/ghostscript/ghostscript.html*
[6] ESP Ghostscript: *http://www.cups.org/ghostscript.php*
[7] ESP Print Pro: *http://www.easysw.com/printpro/*
[8] Turboprint for Linux: *http://www.turboprint.de/english.html*
[9] Gimp-Print: *http://gimp-print.sourceforge.net/*
[10] KDE Printmanager: *http://printing-kde.org*
[11] Foomatic: *http://www.linuxprinting.org/foomatic.html*
[12] Spooler LPRng: *http://www.lprng.org/*
[12] Spooler PDQ: *http://pdq.sourceforge.net/*