Linux Printing Tutorial at Linux−Kongress 2002 Cologne, Germany:

(VI.) Connections:

How CUPS talks to Print Servers, Print Clients and Printers

Table of Contents

(VI.) Connections: How CUPS talks to Servers, Clients and Printers

(VI.) Connections: How CUPS talks to Print Servers, Print Clients and Printers

Presented by Till Kamppeter, maintainer of linuxprinting.org, leader of the Foomatic project, author of XPP, and responsible for the printing part of Mandrake Linux

What we will show: CUPS in heterogeneous networks – Receiving print data via IPP, LPR/LPD, SMB/CIFS, AppleTalk/NetATalk – sending print data via IPP, LPR/LPD, SMB/CIFS, AppleTalk/NetATalk, HP JetDirect/TCP/Socket, parallel, serial, USB, Firewire, SCSI – how to prepare Clients and Server for required protocols... – CUPS backends

CUPS in heterogeneous networks

With CUPS and some additional free software one can set up a universal print server, for the reception of jobs as well as the passing on.. Nearly every printer connection type and nearly every network printing protocol is supported.

As shown in Fig. 1 CUPS cannot only receive jobs from local applications. It offers its native protocol IPP also to the outside world, for print clients to send jobs over the ethernet cable.. With additional daemons, nearly every other printing protocol is available on the receiving end: LPD (Unix), SMB (Windows), and AppleTalk (MacOS).

CUPS prints on most kinds of locally connected printers: Printers on parallel, USB, serial, and IEEE 1394 ports, SCSI printers, and even on HP's multi–function devices using HPOJ (http://hpoj.sourceforge.net/). Printers directly connected to the network or hooked to remote servers are reachable with the TCP/Socket (HP JetDirect or similar), IPP, LPD, SMB, and AppleTalk protocols.

In the figure, all components represented by cyan boxes are part of the CUPS packages, all light red ones are from additional free software packages, which should be part of most standard distributions of GNU/Linux.

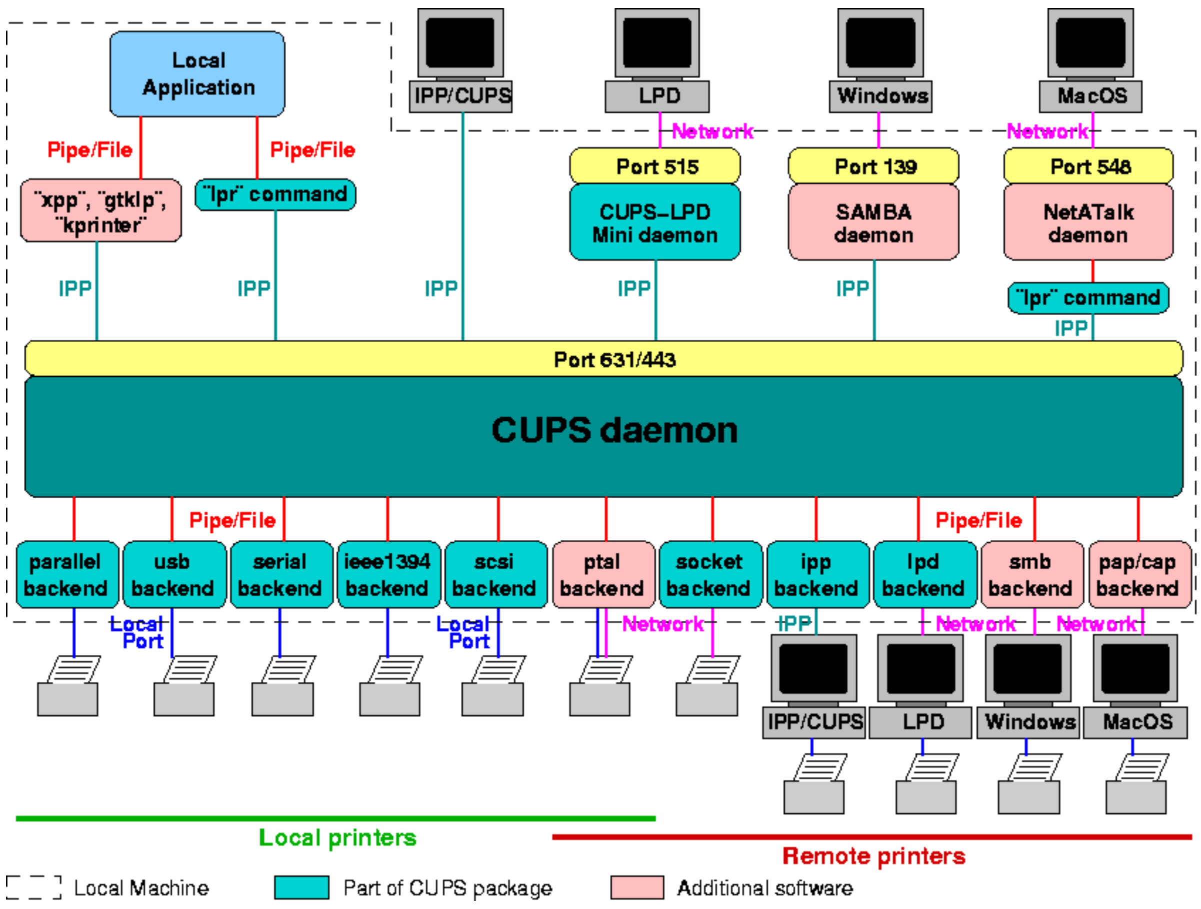

Fig. 1: Possible connections to and from CUPS

CUPS may be easily expanded to cope with other protocols. To support another destination type to where print jobs can be sent, one only needs to write a new CUPS backend. Backends may even be Shell– or Perl–scripts. Once you've written it, make it "world executable", place it in the `/usr/lib/cups/backend/` directory, restart CUPS, and set up a print queue using it. How the backend has to look like so that it works with CUPS is described in the "CUPS Software Programmers Manual" in the chapter "Writing Backends" (http://www.cups.org/spm.html#WRITING_BACKENDS). If you want to make your print server listening to a protocol not yet supported, you need to write a daemon listening to that protocol and either accessing CUPS via the usual command line tools ("`lpr`", "`lpq`", "`lpstat`", ...) or via the CUPS library. You find API info for the CUPS library in the "CUPS Software Programmers Manual" in the chapter "The CUPS API" (http://www.cups.org/spm.html#CUPS_API).

Receiving print data – CUPS as a print server

CUPS speaks many languages. It can be made to serve the needs of various clients, while it has a preference of its own too.

IPP – Internet printing protocol

The IPP is the native protocol of CUPS: Therefore no extra daemon is needed to receive IPP print jobs. The CUPS daemon listens for IPP requests on port 631. Port 631 is the default port for the IPP. Depending on its configuration it also accepts SSL–encrypted IPP, usually on the SSL–port 443. IPP clients can be other machines running GNU/Linux, Unix, or MacOS X (from version 10.2)with CUPS.Machines with other operating systems may have built–in IPP: like Windows 98SE, ME, 2000 and XP. For other Windows versions Microsoft provides free–of–charge add–ons. On the client one has to use

```
ipp://<host name>:631/printers/<printer queue name>
```

or

```
http://<host name>:631/printers/<printer queue name>
```

as URI (Unified Resource Identifier)/destination address. For SSL–encrypted connection one has to use

```
https://<host name>:443/printers/<printer queue name>
```

but both server and client must support it.

CUPS has even additional functionality: For once, IPP is built on the HTTP protocol. This means CUPS is also an HTTP server. When you enter the above URIs in a normal web browser you get into its web interface. Here you can see the printer status (jobs, errors, ...) and configure your printer. In addition, you can add and remove printers manage jobs, read the documentation and more via this web interface of CUPS. Simply start on

```
http://<host name>:631/
```

for unencrypted access or

```
https://<host name>:443/
```

when you have support for encrypted access.

If you have CUPS clients, then you can turn on the "browsing" facility of your CUPS server. This way the clients will "see" the printers on the server automatically. Administrators do not need to do any configuration on the clients when they enable "browsing".

If you have other clients and want to print in PostScript from them, often the printer driver can use a "PPD" (*PostScript Printer Description* file) for complete support of the printers' features. You can download the PPD files for any configured printer from the CUPS server:

```
http://<host name>:631/printers/<printer queue name>.ppd
```

or

```
   https://<host name>:443/printers/<printer queue name>.ppd
```

LPD − Unix clients

Many Unix and GNU/Linux systems use the classical LPD (sometimes also called LPR) printing system or one of the enhanced LPD variants LPRng or GNUlpr. Of course it is possible to convert such systems to CUPS. But in case you do not want this, you can setup CUPS to take LPR/LPD submitted jobs from these clients.

CUPS support for LPD−based clients supports all the basic functions for them to print successfully. With LPD you can print files to specific printers, list the queue status, and so forth. However, the automatic client configuration and printer options are not possible via the LPD protocol. So with LPD clients, you need to manually configure each client for the printers it needs to access.

The "`cups−lpd`" mini daemon provides support for LPD clients and can be used from either the "`inetd`" or "`xinetd`" daemons. Add the following line to the `/etc/inetd.conf` file. It enables LPD support on your server through the "`inetd`" daemon:

```
   printer stream tcp nowait lp /usr/lib/cups/daemon/cups-lpd cups-lpd \
      -o document-format=application/octet-stream
```

Type this is one line, without the backslash. Adapt the path to "`cups−lpd`" according to your installation.

Once you have added this line, give a kick to the "`inetd`" so that it re−reads its configuration:

```
   killall -HUP inetd
```

If you are using the "`xinetd`" daemon, create a file named `/etc/xinetd.d/cups−lpd`. I should contain the following lines:

```
service printer
{
    socket_type = stream
    protocol = tcp
    wait = no
    user = lp
    server = /usr/lib/cups/daemon/cups-lpd
    server_args = -o document-format=application/octet-stream
}
```

The "`xinetd`" daemon automatically reads the new configuration file and enables LPD printing support. Some operating system distributions already have an `/etc/xinetd.d/cups−lpd` file, but it may be de−activated by a line containing "`disable = yes`". Switch this line to "`disable = no`" to activate the file.

The "`−o document−format=application/octet−stream`" parameter shown in the examples is optional. It makes CUPS ignore the mime−type info which the client might send with the printjob. It forces CUPS to auto−detect the file type and pre−process the file accordingly. You can remove the option if it leads to trouble for you.

Security warning: "`cups-lpd`" currently does not perform any access control based on the settings in `/etc/cups/cupsd.conf` or in the `/etc/hosts.allow` or `/etc/hosts.deny` files used by TCP wrappers. Therefore, running "`cups-lpd`" on your server will allow any computer on your network (and perhaps the entire internet) to print on your server.

While "`xinetd`" has built–in access control support, you should use the TCP wrappers package with "`inetd`" to limit access to only those computers that should be able to print through your server.

See also the man pages for "`cups-lpd`", "`inetd`", and "`xinetd`" for more information.

SMB/CIFS – Windows clients

In many networks GNU/Linux or Unix is mainly used as server operating system. There, most workstations are still running Windows. For them the server must emulate the behavior of a Windows print server. Especially the network scanning facility of Windows must find the printers. Under Windows printer drivers often run on the client side, but are offered by the servers for download and semi–automatic installation. It is very convenient to have this "Point and Print" features for Windows clients from a CUPS server.

The CUPS package itself does not need to provide this functionality itself. It can rely here on the free Samba (http://www.samba.org/) package..

When installing Samba make sure that it is compiled with CUPS support. Check whether the command "`ldd `which smbd``" has "`libcups`" in its output.

To configure SAMBA for CUPS, edit the `smb.conf` (In the `/etc` or `/etc/samba` directory) file. In the "`[global]`" section replace the existing "`printing`" and "`printcap`" lines with:

```
printing = cups
printcap name = cups
```

You don't need to define any printing or job handling commands (like `lpq` or `lprm`). Samba calls CUPS library functions if it receives a printjob from Windows clients. Therefore it ignores any command definitions set otherwise in smb.conf..

That's all there is to it! Remote users will now be able to browse and print to printers on your system.

Samba is also able to make available Windows printer drivers so that clients can automatically download and install them. See more about that later in this tutorial (*"Playing Nice With Windows Clients"*).

AppleTalk/NetATalk – Mac clients

Another operating system which is rather popular as desktop operating system is MacOS. Especially the area of graphics and desktop publishing is a traditional Mac stronghold. CUPS itself does not provide MacOS support for old versions of Mac OS directly. However, there are several free and commercial software packages that do. And since the release of Mac OS X version 10.2, CUPS support is builtin there! It has replaced the old legacy LPD–based printing system. (You might not know, but Apple's Mac OS X is a full–blown Unix system, hidden underneath the "Aqua" GUI surface, based on a derivate from FreeBSD...)

Here we show how to configure CUPS for older versions of Mac OS clients to print to it. We use the free NetATalk (http://netatalk.sourceforge.net/) system. NetATalk comes with many GNU/Linux distributions.

Configure each printer in the `/etc/atalk/papd.conf` file. Specify the corresponding PPD file in the `/etc/cups/ppd/` directory for each printer. For a printer named "MyPrinter" the entry would look like:

```
Printer Description:MyPrinter@MyServer:\
        :pr=|/usr/bin/lp -d MyPrinter:\
        :op=daemon:\
        :pd=/etc/cups/ppd/MyPrinter.ppd:
```

NetATalk will use this entry to "publish" the availability to the printer to the clients. Clients will think it is a "normal" AppleTalk printer.

Sending print data – The CUPS backends

CUPS sends its printfiles away through special programs called "backends". There is a backend for each different printfile transfer protocol or connection type. If you install a printer with the `lpadmin` command, you need to specify the connection type with the `-v` parameter. An example (use in one line without backslashes):

```
lpadmin -d printername         \
        -v <device-URI>        \
        -E                     \
        -P </path/to/ppd-file>
```

Local printers: Parallel, USB, serial, FireWire, SCSI

The important part here is the "device–URI". This way you tell CUPS which backend it shall use with the printer *"printername"*. The backends for most types of local printers are already part of the CUPS package. CUPS 1.1.x contains backends for parallel, serial, and USB printers, CUPS 1.2.x will also support FireWire (IEEE 1394) and SCSI printers.

The backends do not only send data to the appropriate devices. They are also called when CUPS is started. They auto–detect which printer models are connected to which ports. So you should set up your BIOS for the parallel ports to allow bi–directional communication. Then your printer(s) can answer to the auto–detection requests.

To see which devices the CUPS backends auto–detect currently, execute them without command line options:

```
$ /usr/lib/cups/backend/usb
direct usb:/dev/usb/lp0 "HP PSC 950xi" "USB Printer #1"
direct usb:/dev/usb/lp1 "EPSON USB Printer" "USB Printer #2"
direct usb:/dev/usb/lp2 "EPSON USB Printer" "USB Printer #3"
direct usb:/dev/usb/lp3 "Unknown" "USB Printer #4"
$
```

The command

```
lpinfo -v
```

shows the devices currently known by CUPS (what CUPS found when it was started). It can look like this:

```
network socket
network http
network ipp
network lpd
direct parallel:/dev/lp0
direct parallel:/dev/lp1
direct parallel:/dev/lp2
direct usb:/dev/usb/lp0
direct usb:/dev/usb/lp1
direct usb:/dev/usb/lp2
direct usb:/dev/usb/lp3
network smb
```

Every line contains the information whether the backend is for local ("direct") or remote printers ("network"). The backend's name follows. If the backend is local, it is not simply the name, but the complete URI string. This "device file" name is used to access a device through this backend. This URI must be used when setting up a queue for a printer connected to this device. If a backend for local printers allows connecting to more than one device, there is one line per device. The example is from a machine with three parallel ports and four USB printers.

Note: CUPS only allows setting up printer queues for backends/devices which it found during its start. So turn on all printers before you start CUPS and check whether your printer's device file is listed in the output of "`lpinfo -v`". If not, restart the CUPS daemon.

Now you can set up a print queue. In graphical printer setup tools (as the web interface of CUPS or the KDE Printing Manager) the same devices as shown by "`lpinfo -v`" will also be listed. Here they appear in the menu where you choose the destination for your queue. If you use the command line tool "`lpadmin`" to set up your print queue, the device URI for the destination of the queue follows "`-v`" parameter (as outlined above). The device URI for local printers is usually composed out of the backend name, a colon and the absolute path of the device file

```
parallel:/dev/lp0
usb:/dev/usb/lp0
```

The serial backend may have additional parameters:

```
serial:/dev/ttyS0?baud=9600+size=8+parity=none+flow=soft
```

Here you specify the serial port (e.g. "`ttyS0`", "`ttyS1`", "`ttyd0`", "`ttyd1`"), baud rate (e.g. "`9600`", "`19200`", "`38400`", "`115200`", etc.), number of bits ("`7`", "`8`"), parity ("`none`", "`even`", "`odd`", and flow control. If you do not need flow control, delete the "`+flow=soft`" portion.

For the SCSI and FireWire/IEEE 1394 backends see the documentation of CUPS 1.2.x

HP's multi–function devices

HP and Epson are the only manufacturers which produce desktop multi–function devices (printer, scanner, copier, photo memory card drives). You can do more than just print with them –– even under GNU/Linux and Unix! For HP's devices the free HPOJ (http://hpoj.sourceforge.net/) software was developed with the support of HP.

This software provides daemons to which all printing, scanning, and photo card access requests are directed. These daemons do the communication with the device. For scanning there is a SANE driver, for photo card access there is an emulation of a DOS drive accessible by the "mtools" and for printing one either uses a special command–line tool ("`ptal-connect`") or one sends the data into a named pipe provided by HPOJ.

For printing via CUPS there is a special backend called "`ptal`" which is a wrapper around "`ptal-connect`". This backend is part of the HPOJ package since version 0.90 and it is automatically installed when a CUPS installation is found.

To use this backend you must at first configure HPOJ for your device. This is rather simple. After installing HPOJ you enter

```
   ptal-init setup
```

and follow the instructions on the screen. The device(s) will be auto–detected and you only need to confirm whether you want to have the device configured. For devices connected via JetDirect on your network you only need to enter the host name or IP of the device.

Restart CUPS now and "`lpinfo -v`" will show you the URI for your multi–function device. It consists of the backend name "`ptal`", a colon, a slash, and the HPOJ–internal (strange–looking) name of the device. This is shown in an example for an HP PSC 950 on the USB:

```
   ptal:/mlc:usb:PSC_900_Series
```

Use this URI for the "`-v`" option of the "`lpadmin`" command when setting up your print queue or choose the "PTAL ..." entry in the destination menu of your graphical printer administration tool.

IPP – Internet printing protocol

As already told, IPP is the native protocol of CUPS. No extra software is needed to use it. CUPS speaks IPP through its "`ipp`" backend. You can use this backend to send your print job to another CUPS server or to a printer with an IPP–capable network card or print server box.

As it would be a long time–consuming process to scan the network every time when one starts CUPS, "`lpinfo -v`" shows only the backend name for this and other network backends.

The URI of an IPP printer has the following form:

```
    ipp://<server name or IP>:<port>/<path>/<printer queue>
```

For a one–port HP JetDirect box or an Ethernet–connected HP printer (named "laserjet") you have:

```
ipp://laserjet:631/ipp
```

See the CUPS Software Administrators Manual (http://www.cups.org/sam.html#10_3) or the documentation of your printer/print server box for the URI you have to use.

On a CUPS (web) server all queues are in the "`printers`" location subdirectory. So the URI for the queue "epson" on the CUPS server "paperwaster" has the URI

```
ipp://paperwaster:631/printers/epson
```

Note: When you have turned on the "Browsing" facility of CUPS on both the clients and the server you don't need to manually set up print queues on the clients. CUPS will make your server's queues automatically available on the client.

On the command line you use as usual the "`-v`" option of "`lpadmin`" to specify the URI. In the KDE Printing Manager (the *"Add Printer Wizard"* part) you can scan your network for IPP printers, in other graphical tools you usually have to enter the URI manually.

TCP/Socket – Ethernet–connected printers

This protocol one usually finds in print server boxes (as HP JetDirect) or in printers directly connected to the Ethernet.

At first you must set up your printer/print server box to have an IP suitable for your network. How this is done is described in the documentation of the device. And when they only talk about setup tools for Windows and MacOS, don't hassle with WINE or VMware, simply follow the instructions in the CUPS Software Administrators Manual (http://www.cups.org/sam.html#COMMON_NETWORK).

Besides the IP which you have for your device now (and also a host name when you set up your name server appropriately) you need also the port number. Most devices listen for print jobs on port 9100 (especially HP). Consult the device's documentation or the CUPS Software Administrators Manual (http://www.cups.org/sam.html#10_3) to find the port number for your device. The device URI is simply

```
socket://<host name or IP>:<port>
```

For example a printer with the IP 192.168.100.20 listening on port 9100 has the URI:

```
socket://192.168.100.20:9100
```

The KDEPrint "Add Printer Wizard" also has a facility for scanning the network to find TCP/Socket printers.

LPD – Unix servers

The legacy LPD protocol is still used by many Unix and GNU/Linux machines (using the LPD, LPRng, or GNUlpr printing systems) and also by many print server boxes. What you need to know to access to a printer on an LPD server is the server's host name or IP and the name of the LPD print queue on the server.

If the server is a computer, you simply take a look in its `/etc/printcap` file. The queue name you find always as the first word of a printer entry. For print server boxes you have to consult the documentation of the box. The CUPS Software Administrators Manual (http://www.cups.org/sam.html#10_3) contains a list for some of the more commonly used types.

The structure of the LPD−URI is simple. It is not more than

```
lpd://<host name or IP>/<queue name>
```

so specifying

```
lpd://paperwaster/lp
```

lets your CUPS queue print to the "lp" queue on "paperwaster".

There are no auto−scanning tools for remote LPD printers.Here you have always to know the server name or IP plus the queue name. As LPD is listening on port 515 you could run a tool to scan the network for machines which have port 515 open. But this still does not give you the queue names...

SMB/CIFS − Windows servers

If you start to introduce GNU/Linux in a network where all machines run Windows, your only possibility to print is using a local printer or printing on a Windows print server. Thanks to Samba (http://www.samba.org/) also containing client software this is no problem. Samba enables the Server Message Block ("SMB") protocol, the native protocol of Windows, to be use in the Unix world.

To configure CUPS for SAMBA, run the following command:

```
ln -s `which smbspool` /usr/lib/cups/backend/smb.
```

Then restart the CUPS daemon. Now "`lpinfo -v`" must show a line:

```
network smb
```

The "`smbspool`" program is provided with SAMBA starting with SAMBA 2.0.6. Once you have made the link and restarted CUPS you can configure your printers with one of the following device URIs:

```
smb://<workgroup>/<server>/<sharename>
smb://<server>/<sharename>
smb://<user>:<password>@<workgroup>/<server>/<sharename>
smb://<user>:<password>@<server>/<sharename>
```

The workgroup name needs only to be specified if your system is using a different workgroup than the server. The "`<user>:<password>`" strings are required when printing to Windows NT/2000 servers or to shares with passwords enabled under Windows 95, 98, or XP.

Note: The Samba client transmits the password in clear text! This makes printing to Samba servers insecure! (One only needs to enter "`ps -auxwww | grep smb`" during the job transmission to the Windows server to see the whole URI−string including the password...) It is therefore strongly recommended to either use a separate account only for printing on the Windows server (best without writable home directory). Otherwise, connect all printers to servers running GNU/Linux or Unix.

The KDEPrint "Add Printer Wizard" has a comfortable facility for scanning the network for available printer shares on Windows (and Samba) servers.

AppleTalk/NetATalk – Mac servers

Also NetATalk (http://netatalk.sourceforge.net/) provides client software to access a MacOS server via AppleTalk. To integrate this software with CUPS, you need a "`pap`" backend. One such backend is available as a shell script from http://www.oeh.uni–linz.ac.at/~rupi/pap/.

On this web site you find a PDF file containing instructions and the "`pap`" backend script itself. Copy the script into the CUPS backend directory (`/usr/lib/cups/backend/`) and restart the CUPS daemon. Now "`lpinfo -v`" will show you the presence of the backend.

The device URIs are:

```
pap:/<AppleTalk printer name>
```

The web interface of CUPS and perhaps other frontends will list all available printers on AppleTalk servers.

Backends for other purposes than accessing a physical printer....

The flexible and modular architecture of CUPS makes it possible to write "backends" for many purposes. Many ideas for this are frequently exchanged on CUPS–related mailing lists. Here are some ideas (and implementations) from recent months:

- A "*multiply*" backend: send a job from a legacy mid–range host computer (generated by an ERP software) and multiply it to be printed on 5 different printers in 5 different department (implemented by a simple shell script).
- A "*network PDF distiller*": send a job to the "PDF" printer and get the file automatically converted into PDF, saved under a unique name and made available for HTTP download automatically (implemented by a simple shell script).
- A "*winpopup*" backend: let a Windows Popup message appear on the screen of the Windows client to tell him his job is printing (implemented by a simple shell script, using Samba's "smbclient" command).
- A "*devnull*" backend: send every job arriving here to "/dev/null" (useful for testing purposes)(implemented by a simple shell script)

If you've got your own ideas or questions about new possibilites, let us know...

(VI.) Connections: How CUPS talks to Servers, Clients and Printers