To manage printers and print directly from a computer or across a networked environment, you need to know how to configure and install a printer. Printing itself requires software that converts information from the application you are using to a language your printer can understand. The Linux standard for printing software is the Common UNIX Printing System (CUPS).

Modern Linux desktop systems make installing and administering printers pretty simple and intuitive, and not unlike how it is done on other operating systems. Nevertheless, it is instructive to understand the underpinnings of how it is done in Linux

CUPS is the underlying software almost all Linux systems use to print from applications like a web browser or LibreOffice. It converts page descriptions produced by your application (put a paragraph here, draw a line there, and so forth) and then sends the information to the printer. It acts as a print server for both local and network printers.

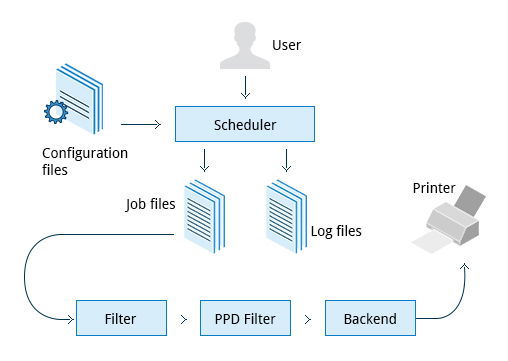
Printers manufactured by different companies may use their own particular print languages and formats. CUPS uses a modular printing system which accommodates a wide variety of printers and also processes various data formats. This makes the printing process simpler; you can concentrate more on printing and less on how to print.

Generally, the only time you should need to configure your printer is when you use it for the first time. In fact, CUPS often figures things out on its own by detecting and configuring any printers it locates.

UPS carries out the printing process with the help of its various components:

* Configuration Files
* Scheduler
* Job Files
* Log Files
* Filter
* Printer Drivers
* Backend.

You will learn about each of these components on the next few pages.



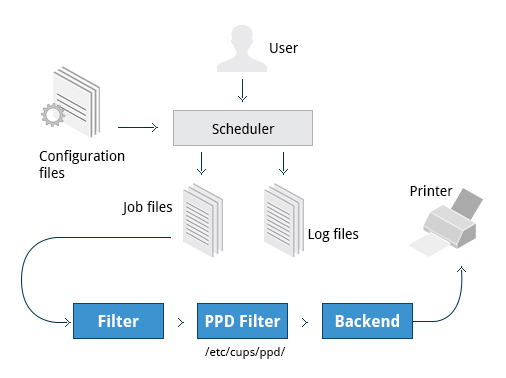
Scheduler

CUPS is designed around a print scheduler that manages print jobs, handles administrative commands, allows users to query the printer status, and manages the flow of data through all CUPS components.

We will look at the browser-based interface that can be used with CUPS,  which allows you to view and manipulate the order and status of pending print jobs.

CUPS uses**filters** to convert job file formats to printable formats.**Printer drivers** contain descriptions for currently connected and configured printers, and are usually stored under **/etc/cups/ppd/**. The print data is then sent to the printer through a filter, and via a **backend** that helps to locate devices connected to the system.

So, in short, when you execute a print command, the scheduler validates the command and processes the print job, creating job files according to the settings specified in the configuration files. Simultaneously, the scheduler records activities in the log files. Job files are processed with the help of the filter, printer driver, and backend, and then sent to the printer.



**Filters, Printer Drivers, and Backends**

Assuming CUPS has been installed you'll need to start and manage the CUPS daemon so that CUPS is ready for configuring a printer. Managing the CUPS daemon is simple; all management features can be done with the **systemctl**utility:

**$ systemctl status cups**

**$ sudo systemctl [enable|disable] cups**

**$ sudo systemctl [start|stop|restart] cups**

***Note*:**The next screen demonstrates this on Ubuntu, but is the same for all major current Linux distributions.

## Adding Printers from the CUPS Web Interface

A fact that few people know is that CUPS also comes with its own web server, which makes a configuration interface available via a set of CGI scripts.

This web interface allows you to:

* + - Add and remove local/remote printers
    - Configure printers:

– Local/remote printers

– Share a printer as a CUPS server

* + Control print jobs:

– Monitor jobs

– Show completed or pending jobs

– Cancel or move jobs.

The CUPS web interface is available on your browser at: [http://localhost:631](http://localhost:631/).

Some pages require a username and password to perform certain actions, for example to add a printer. For most Linux distributions, you must use the root password to add, modify, or delete printers or classes.

**lp**and **lpr** acceptcommand line options that help you perform all operations that the GUI can accomplish. **lp**is typically used with a file name as an argument.

Some **lp** commands and other printing utilities you can use are listed in the table:

|  |  |
| --- | --- |
| **Command** | **Usage** |
| **lp <filename>** | To print the file to default printer |
| **lp -d printer <filename>** | To print to a specific printer (useful if multiple printers are available) |
| **program | lp echo string | lp** | To print the output of a program |
| **lp -n number <filename>** | To print multiple copies |
| **lpoptions -d printer** | To set the default printer |
| **lpq -a** | To show the queue status |
| **lpadmin** | To configure printer queues |

**lpoptions** can be used to set printer options and defaults. Each printer has a set of tags associated with it, such as the default number of copies and authentication requirements. You can type **lpoptions help** to obtain a list of supported options. **lpoptions**can also be used to set system-wide values, such as the default printer.

You send a file to the shared printer. But when you go there to collect the printout, you discover another user has just started a 200 page job that is not time sensitive. Your file cannot be printed until this print job is complete. What do you do now?

## Managing Print Jobs

In Linux, command line print job management commands allow you to monitor the job state as well as managing the listing of all printers and checking their status, and canceling or moving print jobs to another printer.

Some of these commands are listed in the table.

|  |  |
| --- | --- |
| **Command** | **Usage** |
| **lpstat -p -d** | To get a list of available printers, along with their status |
| **lpstat -a** | To check the status of all connected printers, including job numbers |
| **cancel job-id** OR **lprm job-id** | To cancel a print job |
| **lpmove job-id newprinter** | To move a print job to new printer |

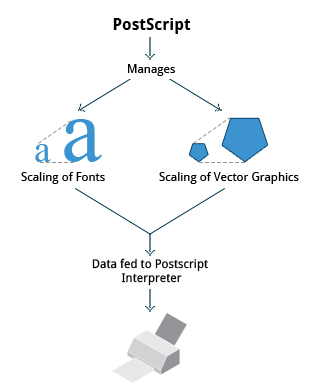
## Working with PostScript and PDF

PostScript is a standard  page description language. It effectively manages scaling of fonts and vector graphics to provide quality printouts. It is purely a text format that contains the data fed to a PostScript interpreter. The format itself is a language that was developed by Adobe in the early 1980s to enable the transfer of data to printers.

Features of PostScript are:

* + - It can be used on any printer that is PostScript-compatible; i.e. any modern printer
    - Any program that understands the PostScript specification can print to it
    - Information about page appearance, etc. is embedded in the page.

Postscript has been for the most part superseded by the PDF format (Portable Document Format) which produces far smaller files in a compressed format for which support has been integrated into many applications. However, one still has to deal with postscript documents, often as an intermediate format on the way to producing final documents.



**Working with PostScript and PDF**

## Working with enscript

**enscript** is a tool that is used to convert a text file to PostScript and other formats. It also supports Rich Text Format (RTF) and HyperText Markup Language (HTML). For example, you can convert a text file to two columns (**-2**) formatted PostScript using the command:

**$ enscript -2 -r -p psfile.ps textfile.txt**

This command will also rotate (**-r**) the output to print so the width of the paper is greater than the height (aka landscape mode) thereby reducing the number of pages required for printing.

The commands that can be used with **enscript** are listed in the table below (for a file called **textfile.txt**).

|  |  |
| --- | --- |
| **Command** | **Usage** |
| **enscript -p psfile.ps textfile.txt** | Convert a text file to PostScript (saved to **psfile.ps**) |
| **enscript -n -p psfile.ps textfile.txt** | Convert a text file to n columns where n=1-9 (saved in **psfile.ps**) |
| **enscript textfile.txt** | Print a text file directly to the default printer |

## Manipulating PDFs

At times, you may want to merge, split, or rotate PDF files; not all of these operations can be achieved while using a PDF viewer. Some of these operations include:

* Merging/splitting/rotating PDF documents
* Repairing corrupted PDF pages
* Pulling single pages from a file
* Encrypting and decrypting PDF files
* Adding, updating, and exporting a PDF’s metadata
* Exporting bookmarks to a text file
* Filling out PDF forms.

In order to accomplish these tasks there are several programs available:

* qpdf
* pdftk
* ghostscript.

**qpdf** is widely available on Linux distributions and is very full-featured. **pdftk** was once very popular but depends on an obsolete unmaintained package (**libgcj**) and a number of distributions have dropped it; thus we recommend avoiding it. **Ghostscript** (often invoked using **gs**) is widely available and well-maintained. However, its usage is a little complex.

## Encrypting PDF Files with pdftk

If you’re working with PDF files that contain confidential information and you want to ensure that only certain people can view the PDF file, you can apply a password to it using the **user\_pw** option. One can do this by issuing a command such as:

**$ pdftk public.pdf output private.pdf user\_pw PROMPT**

When you run this command, you will receive a prompt to set the required password, which can have a maximum of 32 characters. A new file, **private.pdf**, will be created with the identical content as **public.pdf**, but anyone will need to type the password to be able to view it.