This is not a step by step in the true sense of the word. That is to say, I am not going to take you by the hand and walk you through a print filter set up. This is an explanation of what print filters are, how they work, and how to use them to customize and trouble shoot your printing configuration. The knowledge gained here will help you solve a variety of printer set up problems.

This guide is based on LPRng. I have no idea how other printing software works. No GUI configuration tools are used in this discussion.

First, some definitions. Confucious is credited with saying that the beginning of knowledge is to call things by their proper names. Well, not being as wise as Confucious, I am going to call things by some name or another. The following definitions I have just made up. They only apply to this document.

Print filter: An optional shell script called by the print daemon lpd. This script is referenced by the "if" parameter in the printcap file. This script may or may not reformat the document being printed. It may or my not be used for a variety of other activites associated with printing a document. Print filters are the subject of this SxS.

Postscript: A computer programming language designed to be run on a printer. The printer receives the program (a postscript document) from the computer, and then draws the document, page by page, on the paper in the printer. Most if not all linux applications produce postscript files when they are called on to print. Most inkjet printers are not postscript printers, and so another program, ghostscript, is required to successfuly print on such printers. If you have a postscript printer, you don't need ghostscript, although you may still need a print filter.

Ghostscript: A program which converts postscript files into numerous other formats so different types of hardware can display them. This hardware is usually a printer but ghostscript can format documents to display on x11 terminals and fax machines, among other tricks. It is invoked as gs.

Output device: The parameter you supply to ghostscript to tell ghostscript what type of hardware is being used to display the document. Ghostscript reformats the document so the hardware, in this case your non-postscript printer, can properly display it. For example:

```
gs -sDEVICE=epson myfile.ps -sOutputFile=\|lpr
```

tells ghostscript to format the postscript file myfile.ps for display on an epson printer, and send it to the program lpr.

Enscript: A computer program which can convert plain text documents to postscript. Enscript can reformat your text document in many useful and interesting ways as it turns it into a postscript document. Enscript is needed on linux systems to print text documents. For example, this command:

```
enscript -o outfile.ps file.text
```

will convert file.text to postscript, and put it into outfile.ps. You can test this yourself. Create a text file, run enscript against it, and then read the file with gs (Ghostscript) or gv (Ghostview).

Now, let's play with filters. It would be best to work as root from here on out.

This article is writen assuming you know about printcap files. If you don't, the following may be confusing. First, do this:

```
 touch /tmp/printout
 chmod 777 /tmp/printout
```

Then, set up a print queue to play with in /etc/printcap. The following will do:

```
test:\
        :sd=/var/spool/lpd/test:\
        mx#0:\
        :lp=/tmp/printout:\
        :if=/var/spool/lpd/test/filter:
        :sh:
```

You can just cut and paste this into your printcap file.

Explanation of parameters:
sd = your print queue or spool directory
mx = No limit for file size. This is needed for any graphics. Just use it.
lp = Our printing device
if = Our print filter
sh = suppress header

This queue will send print jobs to the file printout instead of your printer. This makes experimentation much more pleasant.

Then, run checkpc -f and look for error messages. AND, be sure to stop and restart your lpd. I don't know where your startup scripts are, but in my Caldera 2.4 and Red Hat 7.2 these commands works.

```
 /etc/rc.d/init.d/lpd stop
 /etc/rc.d/init.d/lpd start
 cd /var/spool/lpd/test.
```

To write your first print filter, use any good text editor. I prefer vi. (I call my printer filters "filter".)

```
#!/bin/bash
cat
```

That' it. Make sure the filter is executable (chmod +x filter). And, it won't hurt to run checkpc -f again, to adjust the permissions on the filter file. You don't need to restart lpd each time you edit the print filter.

Now, before you start printing files and looking at the output, I suggest you make up some aliases to save typing. These aliases are very helpful for this exersize:

```
 alias cf='cat /tmp/printout'
 alias ef='cat < /dev/null > /tmp/printout'
 alias gf='gv /tmp/printout'
 alias pf='lpr -Ptest filter'
```

cf displays the printout file contents on the console.
ef erases the contents of the printout file.
gf displays the contents of the printout file when it is in postscript format.
pf prints the file filter to the file printout.

Print a file. I use the filter file as the file to print, but any text file would do.

```
 ef
 pf
 cf
```

should show you the print filter in unmodified text format. (If if doesn't,you'll have to find out why before you go any farther.)

What has happened? When the lpd daemon runs the filter, it does so in a new shell. In this shell, standard input for the filter comes from the file being printed, and standard output is directed to the lp parameter

in /etc/printcap. This is how it would look from the command line if you didn’t invoke lpd but sent the file to the filter directly.
cat filter (our FileToPrint) | filter > /tmp/printout
If you want to try this out, the command would look like this:
```
 cat filter  | ./filter > /tmp/printout
 cf
```

You should have the same result as if you had invoked lpr (pf).
The filter consists of the single command "cat". By itself, cat reads from the standard input and sends it to standard output. So, when called by lpd, the script does what the command line above does. Easy, huh?

Now, this isn’t a very useful filter. It just passes the file through unchanged. This would be fine if you had a printer that can handle straight text files from linux. Most don’t. Most will get the stair case effect, since linux doesn’t send a carriage return, just a line feed, at the end of each line of text. This is easy to solve. Just run the print job through a filter to add a carriage return at the end of each line. So, edit our filter to look like this:

```
#!/bin/bash
sed "s/.*/&\r/"
```

Then run:
```
ef
pf
vi /tmp/printout
```

With vi, /tmp/printout has ^M (carriage returns) at the end of each line.

Let’s suppose you have a printer that can handle postscript, but not plain text. Enscript is used in this situation.
Change your filter to read as follows:
```
#!/bin/sh
enscript  -p -
```

Enscript will by default send its output to the default printer, not standard output. So, you need the -p - option to specify standard output to enscript.
So, change your print filter to read as above and:
```
ef
pf
cf
gf
```

Now, you should see a postscript document in /tmp/printout.

If you read the file with gv, you will see that enscript has put a header into your document. Enscript can do a lot of things to your document. (I highly recommend reading man enscript.) A more usual enscript command would have a few more options, as follows:
```
#!/bin/sh
enscript -M Letter -Z -B  -p -
```

-M = Media type
-Z = Don’t try to convert postscript documents
-B = Don’t print the header

So, edit your filter to include the additional enscript options and:
```
ef
pf
gf
```
You should just see the filter without a header.

Now, if we had a postscript printer, we would be all done with our basic

print filter. But, likely, you don’t have one of those. So, this is where ghostscript comes in.
The following command takes a postscript file from standard input, formats it for an Epson Stylus 850 printer, and sends it to standard output.

```
 gs -q @stc800p.upp -sOutputFile=- -
```

So, if we are printing a text file to an Epson Stylus 850 printer, our print filter might look like this:

```
#!/bin/sh
enscript -M Letter -Z -B  -p - | gs -q @stc800p.upp -sOutputFile=- -
```

That’s all you need to print files, either text or postscript, to an Epson Stylus 850 printer.
Now, I can almost see the heads shaking. Why is this so simple when other print filters seem so complicated? The answer is that print filters get complicated when the designers have to anticipate diverse needs of many users. You are just one user with a specific need. Therefore, you don’t need a complicated print filter. In fact, this print filter is TOO complicated. The -M option isn’t needed because enscript uses Letter by default. I put it there in case someone needs to use A4 instead of Letter.

The only hard thing about print filters is finding the right ghostscript device for your printer. Some printer manufacturers now support linux, and so this problem is less difficult now then just a few years ago. I am not going to go into any more detail about ghostscript devices, since I don’t know much about them. Visit the ghostscript homepage for more detail.

You now know enough to set up a printer filter. The following information is crucial but it is very good to know when tackling a filter problem.

First, what it special about the shell that lpd starts to print files? We have already mentioned how input and output are redirected in this shell. There are also a variety of useful shell variables defined in this shell. To examine them, change your print filter to read as follows:

```
#!/bin/sh
enscript -Z -p -
set > shellvalues
```

Note, I have taken out the -B option. I like the header because it shows the print time, which can be handy for debugging.
Then,

```
ef
pf
cat shellvalues
```

You will see a bunch of stuff with the cat shellvalues command:

```
BASH=/bin/sh
BASH_VERSION=1.14.7(1)
CONTROL=
Hcc846558-a.twsn1.md.home.com
Proot
Jfilter
CA
Lroot
Nfilter
ldfA469cc846558-a.twsn1.md.home.com
UdfA469cc846558-a.twsn1.md.home.com
```

Most of these are obvious in their meaning. Howerver, the dfA... file is not so obvious. It contains the data (text, whatever) you are trying to print before it is filtered. If you look into your spool or queue directory, you will see files which have various prefixes but will contain the control number noted in the ldf entry. In this cases, this is print job 469 on this host. These other files are used to help control the printing process.

Now, you may be seeing a lot of these files in your spool directory. They should be removed when the print job is done, but, at least in my setup, these files aren't being removed when I print to this test queue.
So, if their presence bothers you, just rm dfA* hfA* and so on.

Now, there are more things that you can do in a print filter. You can print banner pages or send email when print jobs are finished.
For example, to send email to the user when his job is done, the following works. Note, I have gone back to postscript output just to save hassling with bigger comamnd lines.

```
#!/bin/sh
enscript -Z -p outfile -p -
echo Job done `date` | /usr/local/bin/mutt -s Printing \
    `set | grep -v = | sed -n "s/^P//p"`
```

Note: I never could get this to work right using mail instead of mutt.

Now, I suspect you may be thinking this stuff is OK for amateur work, but how to the pro's do it?
Let's find out. Below is a professional quality print filter. It comes with the linux software for the lexmark z53 printer. Lexmark supports linux. They have supplied a driver which takes raw ppm files and turns them into a format their printer understands.
Let's look at this filter and see how the professionals do it.

```
#!/bin/bash
DEBUG=0
PDIR=/usr/local/lexmark/z53
TMPFILE=`mktemp /tmp/lexmark.XXXXXX`

# lxgps is a little utility used to setup the
# Ghostscript parameters.  It sets two parameters
# in particular: resolution and paper size.  It
# determines what to set these to by reading the
# configuration file that is passed in.
GSPARMS=`$PDIR/lxacgsparm z53.conf`

runz53 ()
{
        cat > $TMPFILE
        file -b $TMPFILE | grep -i postscript > /dev/null
        if [ $? -eq 0 ] ; then
                gs -q -dNOPAUSE $GSPARMS -sDEVICE=ppmraw
                    -sOutputFile=- $TMPFILE | $PDIR/z53 --config
                     z53.conf --dotcounts
        else
                file -b $TMPFILE | grep -i PDF > /dev/null
                if [ $? -eq 0 ] ; then
                        gs -q -dNOPAUSE $GSPARMS -sDEVICE=ppmraw
                        -sOutputFile=- $TMPFILE | $PDIR/z53 --config
                         z53.conf --dotcounts
       else
         enscript -B -o - $TMPFILE | gs -q -dNOPAUSE
         $GSPARMS -sDEVICE=ppmraw -sOutputF ile=- -
       | $PDIR/z53 --output blackonly --config z53.conf --dotcounts
                fi
        fi
        /bin/rm -f $TMPFILE
}


runz53 2> /dev/null
```

     33

Some of the lines are just too long to fit so they wrap around. But this shouldn’t cause us much confusion.
Let’s walk through this program. First, the defined constants at the top, lines 3,4,5.
Line 3 DEBUG = 0, may have no use, since it is not referred to elsewhere in the script.
Line 4 PDIR simply points to where the lexmark software is installed.
Line 5 TMPFILE used by the mktemp command to generate a random name for a file to hold the print job before it is filtered.
Line 12 GSPARMS holds the ghostscript parameters based on what data the user has given the lexmark setup program regarding paper size, resolution, color vs black and white. At my current setting, echo $GSPARMS returns -r600 -g5100x6600

Line 32 The script begins here. Why they used a subroutine isn’t obvious to me, but maybe it helps to redirect errors to /dev/null.
So, go to line 14.
line 15: This simply puts the file sent to the print filter all in the $TMPFILE. Note that standard input is assumed but the file is redirected to $TMPFILE.
Lines 17 to 24 are repetitive, just checking to see if the file is either PDF or Postscript. If neither is the case, then enscript is invoked to produce a postscript file. Now, I have never had a problem with a linux application sending a PDF file to my printer. Acrobat prints in postscript. And, since enscript -Z doesn’t convert postscript files, these lines could all be just left out.
The gs command just converts a postscript file to a ppmraw format, which is then piped to the lexmark supplied driver for this printer with the appropriate parameters. Let’s rewrite this thing:

```
#!/bin/bash
PDIR=/usr/local/lexmark/z53
GSPARMS=`$PDIR/lxacgsparm z53.conf`
enscript -ZB -o - | gs -q -dNOPAUSE $GSPARMS -sDEVICE=ppmraw \
                  -sOutputFile=- - | $PDIR/z53 --config z53.conf --dotcounts
```

Believe it or not, this works fine, both with text files and postscript files, the only two types we expect to see coming to our printer. The only problem might be the --output blackonly parameter which I left off so I could get color printing. And, I am not sure where the errors are going to be sent, if any. The -q parameter suppresses messages from gs.

Well, that is enough fun with filters. I hope this SxS has given you enough confidence so you will tackle print filter problems yourself, instead of floundering around looking for a black box solution.

To learn more about LPRng, I strongly urge perusal of this document:
INSERT LPRng link.

Joel Hammer