LPU - An Intro to John R. Schrader's Label Printing Utility

Jim Jackson <jj@franjam.org.uk>

January 2, 2010

This document can be found at `http://www.comp.leeds.ac.uk/jj/linux/lpu.html`

Contents

This Introduction assumes using **LPU version 0.9-26**

1 The Software

1.1 What it is......

LPU (the Label Printing Utility) is a tool for creating labels and any kind of forms, from a textual description.

It uses a line oriented command language to describe the desired objects. Some of the features of the language are: variables, expressions, assignments, procedures, control elements for repeated or conditional execution of statements, import of data from a textual database, and even a small debugger.

To achieve a maximum of flexibility, LPU uses Adobe PostScript(TM) as output format.

The author is John R. Schrader <**jrs@jrssoft.de**>

The Source is at `http://www.jrssoft.de/john/lpu/`

It is best to have ghostscript installed. LPU needs ghostscript style font definitions available, and most of us use **gs** for viewing and printing postscript anyway.

2 Installation

The usual - untar, **cd** into directory, **make** (you may need to change the **BINDIR** setting in the **Makefile**), then **make install** as root.

```
~> tar xfz lpu-0.9-26.tar.gz

~> cd lpu-0.9-26

~/lpu-0.9-26> cd src

~/lpu-0.9-26/src> grep BINDIR Makefile
BINDIR  = /usr/local/bin
#BINDIR  = $(HOME)/bin
cp lpu lpu.sh $(BINDIR)
rm -r $(BINDIR)/lpu

~/lpu-0.9-26/src> make
cc -c devapi.c
cc -c afmsup.c
cc -c tdbfuncs.c
cc -c expr.c
cc -c utils.c
cc -o lpu lpu.c devapi.o afmsup.o tdbfuncs.o expr.o utils.o

~/lpu-0.9-26/src> su
Password:

~/lpu-0.9-26/src# make install
cp lpu lpu.sh /usr/local/bin

~/lpu-0.9-26/src# exit

~/lpu-0.9-26/src>
```

3 What else comes with the program

- Documentation - lpu-0.9-26/doc/en : English documentation, as sgml, HTML, ps, txt The documentation is a pretty good reference manual
- Examples - lpu-0.9-26/demo : Example lpu scripts, example form,paper,label,pen definitions, where to find fonts etc. Also a **README** file.

4 Setting up.....

Create a working directory and copy all the files from **lpu-0.9-26/demo** into it. e.g.

```
~> mkdir labels

~> cd labels

~/labels> cp /usr/src/lpu-0.9-26/demo/* .
```

Now edit the file **lpu.dev** and make sure it contains the correct path to the font directory. The default is......

```
%AfmPath /usr/share/ghostscript/fonts
```

5 Getting started......

Have a look at the examples provided. LPU scripts are in files with the **.lpu** suffix. It is a fairly simple script structure and the documentation explains things fairly completely.

Let's look at the file **example.lpu** to get a feel for the scripting. The start of it looks like......

```
##
##  example.lpu -- a simple example of some lpu features
##      jrs     07.06.1998
##

.requires 0.9-21

; select a paper format
.form   DinA4

; begin the first label (the first page in this case)
.label

; define a border of 1/4 inch on all four sides
.border 0.25i, 0.25i, 0.25i, 0.25i

; select a style, in this case Courier, 18 points, bold italic
.style Cour 18 BI

; output an empty line
.nl
```

Lines starting with "##" or ";" are comments, and lines starting with "." are commands.

.form specifies the name of a layout for a page that is defined in the **lpu.tab** file. **DinA4** is the name given in the demo **lpu.tab** file for a standard form of portrait A4 paper.

.style specifies the font, its size and any attributes **B** for **bold** and **I** for **italic** .

To generate the postscript for **example.lpu** use this command...

```
~/labels> lpu example.lpu > example.ps
```

or

```
~/labels> lpu -o example.ps example.lpu
```

Then view the output with a postscript viewer..

```
~/labels> gv example.ps
```

Now edit **example.lpu** and change some of it or add extra lines. Regenerate the output and re-view it.

6 Label Definitions....

Page sizes and names are defined in the **lpu.dev** file....

```
;-------Id-------Xmax---Ymax---Lm---Rm---Tm---Bm----Options-------------------
%Page   A0P      84.1c 118.9c  0    0    0    0
%Page   A1P      59.4c  84.1c  0    0    0    0
%Page   A2P      42.0c  59.4c  0    0    0    0
%Page   A3P      29.7c  42.0c  0    0    0    0
%Page   A4P      21.0c  29.7c  0    0    0    0
%Page   A4L      29.7c  21.0c  0    0    0    0     .rotate=90 .shift=21.0c,0
%Page   A5P      14.8c  21.0c  0    0    0    0
%Page   A5L      21.0c  14.85c 0    0    0    0     .shift=0,14.5c
%Page   A6P      10.5c  14.8c  0    0    0    0
%Page   A7P       7.4c  10.5c  0    0    0    0
%Page   A8P       5.2c   7.4c  0    0    0    0
%Page   A9P       3.7c   5.2c  0    0    0    0
%Page   Letter    8.5i  11.0i  0    0    0    0
%Page   USEnv10  24.1c  10.5c  0    0    0    0     .shift=0,19.2c
```

You can add other names and definitions of other sizes of paper here is you wish, but the default set is fairly comprehensive.

Labels are defined in **lpu.tab**. There are several label formats pre-defined, have a look in the file. Here are a couple of definitions I created...

```
; compslip  3 complement slips per A4P page
;
form    compslip     A4P
    label   Comp        190m    87m
        base            5.0m    8.0m
        yrep            3       10m
```

This defines 3 “labels”, useful for generating a handy sized complement slip. **compslip** is the name of the form, it contains one label format called **Comp** , the size of which is 190mm x 87mm, repeated 3 times down the page (in the **Y** direction) with a gap 10mm. The **base** gives the x and y offset for the start of the first “label”.

Here’s another.......

```
; Labels 2 x 7 on A4
;
form    L2x7          A4P
    label   l1          90m     36m
        base            10m     20m
        xrep            2       10.1m
        yrep            7       2.1m
```

This is for using a common address label format, 14 to the sheet, arranged 2 x 7.

7 Generating Labels from DataBase output

LPU scripts can read data from standard CSV (coma seperated variable) style text files. Here is an example showing how to generate address labels

```
;   format of labs.tdb is
;   surname, forename, address1, add2, add3, postcode, phone, email

.form   L2x7

.db open 1 /; labs
.use 1 OneLabel
.db close
.eject

:: OneLabel
.label
.style Times 12 B
.nl
.ltext :\@2 \@1
.nl
.style Times 12
.ltext :\@3
.nl
.if "\@4" = "" goto l5
.ltext :\@4
.nl
:l5
.if "\@5" = "" goto l6
.ltext :\@5
.nl
:l6
.style Times 12 B
.ltext :\@6
.nl
.db next
```

The file **labs.tdb** contains lines of the form.....

```
Jackson;Jim;666 Any Road;East Ardsley;Wakefield;WF3 SOB;01924 123456;jj@franjam.org.uk
```

The phone and email fields are not used.

8 Makefile Recipe.....

You can use **make** to ease the creation of the postscript files. This generic recipe allows the postscript **file.ps** to be generated by make from **file.lpu**

```
%.ps:   %.lpu %.tdb
        lpu $< > $@
```

allows you type **make labs.ps** to generate the postscript for the **labs.lpu** script, as long as **labs.lpu** or **labs.tdb** is newer than **labs.ps** .

9 Other Capabilities

- Drawing - lines, boxes, circles
- Variable Line widths and styles
- Windowing
- Embedded postscript (eps) support
- Preprocessor **#include FILENAME**
- Full logical, numeric, string and many system functions
- A simple debugger and tracing facilities

10 Other uses........

- Labels for Homemade Preserves, Pickles etc
- simple complement slips & business cards
- audio cassette & CD case inserts

HomePage http://www.comp.leeds.ac.uk/jj