Design in the command line: Recipes for tasty outcomes

Manufactura Independente

Ana Isabel Carvalho & Ricardo Lafuente
Why?
• fast
• repeatable
• composable
• scriptable
• shareable
Great for graphic workflows
But hard to get into
Best practice: Document your recipes
(shell history is good enough if you’re lazy)
Batch operations with ImageMagick

# resize all images in place
mogrify -resize 25% *.png

# create png copies of all jpg files
for f in *.jpg; do convert $f ${f/.jpg/.png}; done

# create a pdf with all the jpegs in this dir
convert *.jpg out.pdf
# remove all metadata from an image
exiftool -all= -overwrite-original *.jpg

# add/replace author metadata
exiftool *.jpg -overwrite-original -artist="Luther Blissett"
Scriptable gradient maps
animation with ffmpeg
Inkscape has a command line mode too

# convert SVG to PDF
inkscape poster.svg --export-filename=poster.pdf

there’s lots of options, run inkscape --help
# merge many PDF files into one
```
pdftk *.pdf cat output all.pdf
```
# turn an A5 pdf into an A4 booklet ready for folding+stapling
`booklet-impose-a5 source.pdf`

# get the total number of pages
`booklet-get-page-total source.pdf`

# insert an A5 blank page at page 3
`booklet-insert-blank-page source.pdf a5 3`
This CPU waits for a different one

Return true if the calling CPU is allowed to print oops-related info. This is a bit racy.

Called when the architecture enters its oops handler, before it prints anything. If this is the first CPU to oops, and it's oopsing the first time then let it proceed.

This is all enabled by the pause_on_oops kernel boot option. We do all this to ensure that oopses don't scroll off the screen. It has the side-effect of preventing later-oopsing CPUs from mucking up the display, too.

It turns out that the CPU which is allowed to print ends up pausing for the right duration, whereas all the other CPUs pause for twice as long: once in oops_enter(), once in oops_exit().
can't trust the integrity of the kernel anymore:

64-bit random ID for oopses:

Called when the architecture exits its oops handler, after printing everything.

This thread may hit another WARN() in the panic path.
100 usecs 1 second
0 usecs 1 second

CONFIG_SMP

filled in by handler
CONFIG_NUMA_BALANCING
CONFIG_SCHED_DEBUG

sys_waitd() overwrites everything in ru
booklet-impose-a5

A script to turn an A5 PDF into a set of 2-up A4 pages for double-sided printing. It’s a way to quickly perform imposition so we can print A5 booklets in a double-sided A4 printer so that they come out ready to be folded and stapled.

The script uses podofimpose with a .plan file with the imposition logic, which is directly based on an original recipe by Pierre Marchand, the author of podofimpose. We are also indebted to stdin for pointing the right way.

Read the .plan file source for an explanation of the logic to arrange and sort the pages so that they’re laid out in the correct order.

Usage

```
booklet-impose-a5 source.pdf
```

and it will create `source-a4.pdf`.

booklet-get-page-total

Returns the total number of pages on a PDF file.

```
5 booklet-get-page-total source.pdf
24
```
# download and link to our config files

git clone https://gitlab.com/manufacturaind/dotfiles.git
cd dotfiles
stow */
Scraping sources for inspiration

```bash
# decompress any archive
aunpack archive.tar.bz2
# and also compress
apack archive.zip files/*
```
# get photo albums

gallery-dl https://flickr.com/photos/fdctsevilla/
gallery-dl https://instagram.com/presentandcorrect/
gallery-dl https://twitter.com/lubalincenter/
# youtube-dl / yt-dlp

# download videos

```bash
yt-dlp https://www.youtube.com/watch?v=1M9VA0UBRQ0
```

# or only the audio

```bash
yt-dlp -x https://www.youtube.com/watch?v=1M9VA0UBRQ0
```
plain wget

```bash
# download image sequence
for n in `seq 1 80`; do wget https://example.com/img/$n.jpg;
```
For mostly everything else there’s the Internet Archive <3
montage *.png montage.png
$ cat ~/libregraphicsmag/issues/*
$ cat ~/libregraphicsmag/issues/*
Libre Graphics magazine Archive
Issue 2.4 — Capture

Cover, Issue 2.4

This issue looks at Capture, the act of encompassing, emulating and encapsulating difficult things, subtle qualities. Through a set of articles we explore capture mechanisms, memory, archiving and preservation of volatile digital information, physicality and aesthetization of data.
Extract text from Scribus files

See the `scripts/scribus-extract.py` script.

Extracting image paths from Scribus files

```bash
cat 61-original-scribus-files/lgmag-1.1-p* | \n  grep -oE "\"\"|\(jpg|png\)\"" | \n  sort | uniq | \n  sed 's/^//g; s/\.///g' > images-1.1.txt
```

Re-downloading images from repos

After extracting the image lists from the previous recipe, we did this (example for 2.4):

```bash
mkdir -p images/2.4
cd images/2.4
while read f; do wget "https://gitlab.com/libregraphicsmag/vol2issue4/raw/master/\$f"; done < ../..image-lists/1.1.txt
```

Resizing images to a max width and height

We wanted max 1800px width and 1400px height, while keeping smaller images as they are.

```bash
cd images/2.4
for f in *; do echo \$f; mogrify -resize 1800x1400\> \$f; done
```

or, to alter all images in subdirectories:

```bash
cd images
find . -name "*" | xargs mogrify -resize 970x4080\>
```

Review text files with a particular string

We needed to check every instance of a lowercase acronym (e.g. "svg") and edit it manually if it needed to be made uppercase again.

```bash
grep \"\$symh\" issue* \-lr | xargs vim -n
```
Turn all images into image links

We wanted to change all images into a thumbnail which would then link to the original image.

So this

```markdown
[!!](/images/2.1/picture.png)
```

should become

```markdown
[!!](/images-small/2.1/picture.png)[!!](/images/2.1/picture.png)
```

So we use the following sed script:

```bash
cd content
find . -name "*.md" | xargs sed -i 's/\((\(/images\)/\)\))/\1\)/g; s/1/images\//1/images-small/'
```

The sed operation comes in two parts: first we create the 'image inside link', and after that we fix the directory of the thumbnail so that it points to the `images-small` directory.
figlet / toilet

$ figlet "libreplanet"

libreplanet
badger
gnu.txt --font 5x8
--leading=-1 --tracking=-1
--textcolor="#dddddd" --bgcolor="#222222"

https://gitlab.com/manufacturaind/badger
gnu.org  gnu.org  GNU.ORG
## Subgroups and projects

<table>
<thead>
<tr>
<th>Subgroup</th>
<th>Shared projects</th>
<th>Archived projects</th>
<th>Search by name</th>
<th>Updated date</th>
</tr>
</thead>
<tbody>
<tr>
<td>typebits.gitlab.io</td>
<td></td>
<td></td>
<td>★ 1</td>
<td>12 hours ago</td>
</tr>
<tr>
<td>Documentation for the Type:Bits workflow in MkDocs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>font-skeleton</td>
<td>★ 0</td>
<td></td>
<td></td>
<td>1 year ago</td>
</tr>
<tr>
<td>Skeleton for bitmap fonts, made to be forked and edited</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>font-flamingo</td>
<td>★ 0</td>
<td></td>
<td></td>
<td>2 years ago</td>
</tr>
<tr>
<td>A wavy typeface with a whimsical attitude.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>font-mozblock</td>
<td>★ 0</td>
<td></td>
<td></td>
<td>2 years ago</td>
</tr>
<tr>
<td>A typeface designed inside Minecraft at the &quot;Design open web fonts with Minecraft&quot; wor...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>font-setperset</td>
<td>★ 0</td>
<td></td>
<td></td>
<td>2 years ago</td>
</tr>
<tr>
<td>A 7x7 bitmap font created at the fonts.txt workshop in Barcelona (2014)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>font-spalte</td>
<td>★ 0</td>
<td></td>
<td></td>
<td>2 years ago</td>
</tr>
<tr>
<td>font-roseta</td>
<td>★ 0</td>
<td></td>
<td></td>
<td>3 years ago</td>
</tr>
<tr>
<td>A spiraling typeface with an attitude. You can hear it singing if you get close enough.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>font-ladrillo</td>
<td>★ 0</td>
<td></td>
<td></td>
<td>3 years ago</td>
</tr>
<tr>
<td>A stern font that nevertheless still knows how to have fun.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>font-muralla</td>
<td>★ 0</td>
<td></td>
<td></td>
<td>3 years ago</td>
</tr>
<tr>
<td>A sturdy typeface which won't take no for an answer.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Creating a new font

Want to dive in the wonderful world of bitmap font drawing?

There are two available methods to quickly get started with a ready-to-edit base font template:

1. **Fork the font-skeleton repository.** Faster to get started, but needs some tweaking to the font's configuration.

2. **Use the cookiecutter-bitmapfont template.** Requires you to install cookiecutter, but makes it easier to define the configuration values through a series of simple questions.

Additionally, you can try forking an existing font to have an existing base to work from -- just follow method 1 and instead of font-skeleton, choose another repository to fork from.

**Method 1: Fork the font skeleton repo**

You can get started with a bare version of a bitmap font by forking the font-skeleton repository on the GitLab.com web interface.

If you want an easier start, try forking one of the existing fonts in the Type:Bits repository list.
Building font files

```bash
sudo apt-get install python-virtualenv
```

Now, just run `make install` to pull and install the remaining dependencies.

Finally, clone the `bitmapfontbuilder` repository to the same directory where you've pulled this font, so that both are inside the same folder.

```bash
cd ..
git clone https://gitlab.com/typebits(bitmapfontbuilder)
```

**Building locally**

Whenever you want to re-generate the font files from the source, just type

```bash
make build
```

This will generate the files in FontForge (.sfd), TrueType (.ttf) and OpenType (.otf) formats.

**Building and deploying to a repository**

A better workflow is to create a Git repository from this font directory and running

```bash
make deploy
```

This command is identical to `make build` but pulls changes from the repository before building, generates the font files, commits the newly built files and pushes the changes back into the remote repository. This is particularly handy if you’re working with more people and want to incorporate their changes quickly.
transpacing

a script to transplant spacing from one font to another.

font with no spacing 
+ similar font with good spacing  
------------------------
= font with decent spacing
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.
fontconvert Douar.sfd --otf --ttf --woff
import fontforge
font = fontforge.open('douar.sfd')
for glyph in font:
    font[glyph].stroke('circular', 30, 'square', 'bevel', ('cle
font.generate('douar-outline.ttf')
Thank you!

manufacturaindependente.org