



Full Circle

THE INDEPENDENT MAGAZINE FOR THE UBUNTU LINUX COMMUNITY

SPECIAL EDITION



Command & Conquer



Volume Three

Issue 51 - 75



Recently, I made the decision to move from WMFS (Window Manager From Scratch) to XMonad, since WMFS had started to present some issues when handling certain windows and layouts. Once I had made the switch, I was fighting with xmobar to get it working. Luckily, a guy on the ArchLinux Forum made the suggestion that I use Conky with dzen2 for my panel instead of Conky with xmobar, as I was trying to do. And so, I will be covering how to create your own status bar using dzen2 and Conky. Before I start I'd like to note that I am using a version of dzen2 that has xft support enabled. If you happen across a line in my configuration files/examples that is in the format "Togoshi Gothic:size=9", you'll need to replace it with a font from xfonts-terminus, or else try the dzen2 packages from <https://launchpad.net/~justinbognor/+archive/ppa/+packages>, which seem to have XFT support. For

those of you interested in my entire xmonad.hs, it's listed in the further reading section.

Below is my .conkyrc that I use for the status bar. I'll cover the important lines and explain what the scripts do. I won't be including my scripts, since they are either only for ArchLinux or are used for programs (like MPD and Dropbox) that not everyone uses. If you want a specific script, feel free to email me.

```
background no
out_to_console yes
out_to_x no
update_interval 2
total_run_times 0
use_spacer none
```

```
TEXT
${execi 1 /usr/bin/mpd-info}
| Dropbox: ${execi 5 echo
$(dropbox status)} |
$memperc% ($mem) | Updates:
${execi 300 python
~/Dropbox/Scripts/conky/packages-short} | ${execi 60
python
~/Dropbox/Scripts/conky/gmail
.py} Email(s) |
```

```
^fg(\#9F6B00)${time %a %b %d
%H:%M}^fg()
```

The first line disables the background, and the next two disable the graphical aspect, so that Conky simply returns a string. The update_interval tells Conky how often to refresh the information. Total_run_times tells Conky to exit after a certain number of runs. Set it to 0 for infinite number of runs. User_spacer none tells it to not space out the commands below TEXT, since I do it by hand.

The following line of commands does the following:

```
<artist>-<song> | Dropbox:
<status> | % (<used RAM>) |
Updates: <# of updates> | #
new Email(s) | <clock>.
```

The clock is wrapped in ^fg(\#9F6B00)^fg(), so that dzen2 prints it in a nice gold colour, which matches my currently selected workspace (separate

dzen2 instance). To see a screenshot, check the second link in the Further Reading section.

Once you've decided on your .conkyrc, you'll need to decide on the switches you want to use with dzen2. For that, you'll need to know the following switches:

- fg <hex> - sets the foreground colour using the hex value for the colour
- bg <hex> - sets the background colour using the hex value for the colour
- fn - sets the font
- h <size in pixels> - sets the height
- y <y-coordinate> - shifts the bar up/down
- x <x-coordinate> - shifts the bar left/right
- w <pixels> - sets width of the bar
- sa <l,c,r> - set alignment of slave window
- ta <l,c,r> - set alignment of title windows
- xs <screen> - set the screen to display on.

Santana - Sideways (Feat. Citizen Cope) | Dropbox: Idle | 9% (372MiB) | Updates: None | 1 new Email(s) | Fri Jul 15 17:15

An example of how I call dzen2 for my workspaces (not the dzen2 instance with Conky):

```
dzen2 -fg '#9c9c9c' -bg '#0c0c0c' -fn 'Togoshi Gothic:size=9' -h 18 -y 0 -w 660 -ta l
```

An example of how I pipe Conky (it's a little more complicated the way I do it in my config file, but it's just easier to manage that way):

```
conky -c  
~/.xmonad/.conkyrc_dwm_bar|dz  
en2 -w 1040 -x 660 -ta r
```

The x-coordinate is the same as the width of the first bar, so that it lines up. You can also configure some default options for dzen2 using your .Xresources file in the format of:

```
dzen2.<property>: <setting>
```

Example:

```
dzen2.font: "Togoshi  
Gothic:size=10"
```

Hopefully you've found this useful. For those of you who are going to use this to pretty up your Conky without using lua, or to those of you who run a window manager where there is no

integrated status bar, I'd be interested to see how you guys put this information to use! If you have any questions, comments, or requests, you can reach me at lswest34@gmail.com. Please put "C&C" or "FCM" in the subject line of the email, so it doesn't get lost.

Further Reading:

<http://pastebin.com/3q5TGQJJ> – my xmonad.hs
<http://lswest.deviantart.com/#/d3lalg7> - Screenshot



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



Server Circle is a new question and answer site run by techies.

Users with any level of experience can ask technical questions for free about anything server related, and receive answers from trusted experts, who are rated by the community.

With time you can earn reputation points, and even financial rewards, by contributing your answers to questions from other people.

<http://www.servercircle.com>



NOTE: Server Circle is not affiliated with, nor endorsed by, Full Circle magazine.



I recently received an email from a reader, Chris, who can be found on launchpad.net as MrChris. He was asking me about Japanese input in LaTeX, as a follow-up to my article in FCM Issue #50. I didn't know the answer off the top of my head, but, together, we figured out a solution, which I felt I should share with anyone who may have the same question. Also, since the latex-cjk-* packages stand for "Chinese, Japanese, Korean", I've covered (roughly) how to get each of the 3 languages working.

Packages:
`texlive`
`texlive-latex-extra`
`latex-cjk-common`
`latex-cjk-japanese`
`latex-cjk-japanese-wadalab`

For Chinese (using Method #1 listed below), you'll also need:
`latex-cjk-chinese`
`latex-cjk-chinese-arphic-gkai00mp`

For Japanese, I used the following template:
<http://pastebin.com/tasDkhZ3>

As a brief explanation to the code:

`\usepackage{ucs}` – unicode support
`\usepackage[utf8x]{inputenc}` – extended UTF-8 encoding (includes asian characters)
`\usepackage[english]{babel}` – sets the typographical rules (in this case, to English). This includes, among others, where to break words if it flows over a line. By assigning a typographical set of rules that doesn't apply to Asian characters, you ensure that groups of Kanji aren't separated when breaking over a line. In using a language that doesn't support asian locale, you ensure that Kanji that need to be grouped together, aren't split.

`\usepackage[overlap, CJK]{ruby}` – This is the package required to get CJK working
`\usepackage{CJKulem}` – Used for certain options for CJK

`\renewcommand{\rubysep}{-0.2ex}` – changes the default

spacing
`\newenvironment{Japanese}{\CJKfamily{min}\CJKtilde\CJKnospace}{}` - This creates a new environment (like "document", "enumerate", "itemize", etc.), which configures the options for Japanese input.

The rest of the code is pretty self-explanatory. Once you compile the file, the PDF should contain the Japanese, in a nicely-formatted font. The only issue is that the text is not compiled top-to-bottom and right-to-left, as is the standard for Japanese. I haven't yet found a solution that works well for this. If anyone has a suggestion, feel free to pass it along.

For Chinese:

Method #1:

The same template that works for Japanese would work for Chinese too, with the following adjustments:

`\newenvironment{Chinese}{`

`\CJKfamily{gkai}`
`\CJKtilde`
`\CJKnospace}{}`

Also, the line "`\begin{Japanese}`" would logically have to be changed to "`\begin{Chinese}`". Besides that, it will compile. As mentioned above, this will result in left-to-right and horizontal text. For top-to-bottom and right-to-left formatting, move on to Method #2.

Method #2:

Disclaimer: I have not tried this method, nor do I know who originally wrote the script. However, Chris, the reader who originally posed the question, has tried it and gotten it to work, so I felt it deserved to be included here.

Download the following script:
http://scripts.sil.org/cms/scripts/request_download.php?&format=file&media_id=xetex_chinese_sample&filename=xetex_chinese_sample.zip

According to Chris, he had to

make the following changes:

“Unfortunately, the uploader created it on a Mac where the fonts are different.

However, in XeTeX you can specify fonts that are available in other programs and system-wide - so you can just pick a nice font in LibreOffice and specify that one. After changing the fonts, using "xetex <file>.tex" [it] actually compiles successfully! See <http://pastebin.com/WiiXsbNz> for a modified file.”

For Korean, the same method as used for Japanese can be used. You'd simply need to pick a font family for Korean, and install the following package:

latex-cjk-korean

Since I don't speak Korean, I haven't been able to try it out and find the correct font family, but I would suggest trying it without a font family specified, or else searching for a test document online.

I hope you have found this article useful. If anyone has a specific issue (or question), you can reach me at

lswest34@gmail.com. Please be sure to include “FCM” or “C&C” in the subject line, so that I don't overlook it. Any suggestions, or corrections, can also be emailed to me at the above address. If your enquiry requires a large excerpt of code, I would appreciate it if you would upload it to pastebin.com, or some other such site, and simply supply the link in the email. If the code isn't too long, please quote it or clearly denote the text from the email, in order to allow for quick referencing.

References:

Original Japanese test document is from here:

<https://bbs.archlinux.org/viewtopic.php?pid=622937#p622937>

Information on UCS is from here:

<http://kile.sourceforge.net/Documentation/html/lang.html#ucs>

A blog post by Chris on Japanese:

<http://blog.mindfall.net/?p=221>

A blog post by Chris on Chinese:

<http://blog.mindfall.net/?p=236>



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



ServerCircle

Server Circle is a new question and answer site run by techies.

Users with any level of experience can ask technical questions for free about anything server related, and receive answers from trusted experts, who are rated by the community.

With time you can earn reputation points, and even financial rewards, by contributing your answers to questions from other people.

<http://www.servercircle.com>



NOTE: Server Circle is not affiliated with, nor endorsed by, Full Circle magazine.



Regarding FCM#52: Gord Campbell was kind enough to point out to me that, while Chinese can be written vertically, it seems that it's not as common as I had assumed it to be.

If you, the readers of Command & Conquer, are anything like me, you will have probably enticed one or two people you know into trying Linux. My first and foremost aim is always to make the operating system run as stable as possible, so that the user will have less to fight with (in order to fend off complaints about how Windows had fewer problems). Once I have that taken care of, I give the user a short run through some things they need to be aware of – what this list consists of is generally dependent on the distribution I installed. However, most recently I installed ArchLinux on a netbook (after we decided Chakra didn't quite fit the bill). Shortly thereafter, the KDM wasn't starting properly on run level 5, dropping the user into a tty screen. This is generally one of

those moments where the user will feel lost as to how to proceed, which brings me to the purpose of today's article, the Message of the Day (motd). The Message of the Day is a file on the system that is printed upon login, located at `/etc/motd` and `/etc/motd.tail`. There is also a package called `update-motd` for Ubuntu, which seeks to offer more options. Usually used by server administrators to list the terms of use, or list help information (rooms, phone numbers, emails, etc.). Since it's a text file, absolutely anything can go in there. I felt it might be useful to cover a list of things you can put in your Message of the Day, so that if your user needs to log in on a tty shell, he (or she) will not be completely lost.

About the files

For Ubuntu Server (and probably the Desktop version), you need to edit `/etc/motd.tail` and `/etc/motd`, so that the `.tail` file doesn't overwrite the `/etc/motd` file after a reboot. For the newer

versions of Ubuntu, the files may be located in `/etc/update-motd.d/`. This seems to be a result of the `update-motd` package, but the files are clearly named and should be easy enough to edit.

Useful commands

The first thing I did was add a list of commands for getting around in the shell (`cd`, `cat`, `vim/nano`, `pwd`, etc.) and a brief explanation of each command. After that, you may want to supply some system-specific commands (i.e. if you have a command-line email client configured, how to access that); however, it is completely system-dependent.

Important locations

You could also include a list of important files/directories, where the user can either access help pages/tutorials, or else a file with more commands, or a file containing contact information (if it's not included in the MotD already).



The Message of the Day is a file on the system that is printed upon login

Troubleshooting

I would also list one or two of the most common fixes (such as restarting the Xserver or login manager) in order to get the User out of the tty shell. If you feel that the user would be comfortable with it, you can also explain how to capture debugging information, or log files, that they should then send to you.

Mounting external devices

If the user needs to send you log files, or needs to back up a file (or restore a backup), it would be extremely helpful for them to connect an external drive of some sort to the computer. Since `hal` or



If anyone has an interesting or unique use for their Message of the Day, feel free to share it with me.

udev generally handle that, it may still work in the tty shell, although I find it rather unlikely. Therefore, I would list a few commands (label them something useful, like "USB" (generally FAT32), "external hard drive" (usually ntfs or ext3/ext4), or even specific devices if you know the hardware). I wouldn't advise adding all this extra information in unless it's extremely important, or if the user lives in a location where you, or someone who is comfortable with Linux, cannot easily intervene and help, be it in person or via SSH.

Contact information

If you're employing this as a system administrator for a company, or for someone you don't know well, I would include a phone number, email address, and

maybe a website where they can contact you.

That essentially covers all the items I would usually include in a Message of the Day, but it's by no means extensive. If anyone has an interesting or unique use for their Message of the Day, feel free to share it with me at lswest34@gmail.com. If I find it interesting as well, I may compile a list of them for next month's article. As always, suggestions and questions are also very welcome. Any emails sent to me should include "C&C" or "FCM" in the subject line, so that they get sorted properly.



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



ServerCircle

Server Circle is a new question and answer site run by techies.

Users with any level of experience can ask technical questions for free about anything server related, and receive answers from trusted experts, who are rated by the community.

With time you can earn reputation points, and even financial rewards, by contributing your answers to questions from other people.

<http://www.servercircle.com>



NOTE: Server Circle is not affiliated with, nor endorsed by, Full Circle magazine.





In the process of configuring ArchLinux on a PC, I realized just how much more efficiently I work in Vim than I do in any other text editor. So, when Article Day rolled around, I checked to see if there had been an article on Vim. Lo and behold, there was only a brief introduction to Vim. To start with, I should probably differentiate a text editor from an office suite like OpenOffice/LibreOffice. A text editor is used for editing stuff like *.txt, *.rtf, *.conf, *~, and basically any other plain text document format (i.e. no formatting). An office suite, on the other hand, works on *.doc and *.odt files (among others). I usually use plain text documents for quick to-do lists, and for writing code. I'm fairly certain there are many of you out there who either write quick notes to themselves, write code (be it software or websites), and perhaps edit some configuration files.

What's the difference between Vim and GVim?

Vim is the console-oriented

version of the program (for use in gnome-terminal, rxvt-unicode, tty screens, and so forth). GVim is a graphical front-end for Vim, and allows easy access to some of the more esoteric commands.

Why should I use Vim/GVim?

I'm not going to say that Vim/GVim is the only text editor anyone should use, but it's the one I prefer. Why do I prefer it? Simply because it offers sane defaults and keybindings that allow for very quick editing - without the need for a mouse. I find that keeping my hands on my keyboard actually allows for faster work than pausing to move the mouse (or keeping a hand on the mouse most of the time). If you find yourself nodding along with this, you should give Vim/GVim a shot. For example: Say you wanted to delete 250 sequential lines in some source code because you need to move the method elsewhere. In Vim you would simply type in 250 and then hit the "d" key twice (for delete line). If you then want to paste it somewhere else, simply move

there and hit "p". No need for a mouse or any highlighting whatsoever.

How do you get around if you can't use a mouse?

In GVim you are able to use a mouse, though it's really not necessary. In Vim you get around using page up/page down and the arrow keys (or hjkl where h=left, j=down, k=up, and l=right).

What about syntax highlighting?

Vim and GVim are fully customizable in a configuration file, and allows for custom colors and formatting for code according to file extensions. It also offers a very good syntax engine by default that simply needs to be enabled.

What other useful features does it offer?

- It allows you to:
- Set a custom size (number of spaces) for tab characters, which is useful when coding python.
 - Copy, paste, delete, and insert

controls that allow for careful formatting (dw = delete word, dd=delete line, yw=copy word, yy=copy line, p=paste, etc).

- Printing (with a sane printing default)
- A tutorial that can be accessed with the command 'vimtutor'
- Macros (easy creation and execution).

Some basic commands:

- hjkl – movement (h=left, j=down, k=up, and l=right) – this works only outside of any mode. Inside a mode you'll need to use the arrow keys.
- y – copy (yy = copy line)
- d – delete (dd = delete line)
- w -word (used in conjunction with y or d mainly)
- 1-∞<command> - repeat the command that number of times (i.e. 250dd = delete 250 lines starting at, and including, the line currently selected)
- :w – write (can be combined with q to write and quit)
- :e – open
- :q – quit
- Esc – exit any mode
- i – insert mode (inserts before current selection)

a – add mode
r – replace mode
u – undo
ctrl+r – redo.
p – paste anything copied or deleted in Vim/GVim
:hardcopy – the command to print
:%s/<regex>/<to replace>/ - allows you to substitute anything you search for in the first block with what is in the second block. This also accepts regular expressions.
:set <option> - allows you to set an option for the running instance. See my configuration file for some examples of settings (which I set for any instance).
ctrl+v – visual mode, allowing the highlighting of multiple lines inside visual mode:
I – insert before all highlighted letters
X – delete all highlighted letters
middle-mouse button (or shift + insert) paste from external linux clipboard after selecting the text to paste.

This is just a basic list of commands I frequently use. The line marked with asterisks is actually a linux shortcut, but it's very useful in Vim/GVim.

How do I use Vim/GVim?

To open a file, you can either append the path after the Vim/GVim command, or open it using :e once it loads. Once you have opened the file that you want to edit, you can then hit “i” (the key on the keyboard) to enter insert mode. It will tell you what mode is active (blank means there is currently no mode being used) in the bottom-left corner. For the sake of this example, let us simply write “Com3mand 6 Conque”. Once you've entered the text, it's time to fix it up. Move to the 3 using the hjkl keys, and then hit the “x” key to delete it. Head over to the 6 and hit the “r” key, and then enter the ampersand (“&”). Once that's done, you should have: “Command & Conque”. Notice that the “r” is missing! Move the cursor over the e and you'll notice that you can't move the cursor past the already entered text. How do you enter the r then? You can either hit the “i” key for insert and move the cursor using the arrow keys and then enter it. Or, you could “append” the text using the “a” key, which moves the cursor to the space after the selected letter. Then type “r” and you're all set. If you want to save it, do so with :w (and if you haven't given it a path or name yet, you must do so after the w). If you want

to just quit, use “:q” (in this case you'll need to add an exclamation point after the q to ignore the changes made). After the “:q!”, you will find yourself back at your desktop (from GVim) or in your terminal if you used Vim.

A few things to remember:

The Vim shortcut keys that use letters (u, ctrl+r, y, d, w, etc) all require you to be in the “blank mode” (i.e. without any editing mode active). Otherwise, you'll just type the letter. Get in the habit of hitting escape after you've made a change so that you don't start typing in gibberish.

Using numbers to repeat commands also works only outside of a mode.

If, when pasting something, you find that it spaces it all out oddly, simply activate the past mode (:set paste) and when you're done, disable it with :set nopaste.

If you want to copy something, and you find you copy the line numbers too, simply shut them off with :set nonumber, and back on with :set number.

If you have any questions, or would like a follow up article with a step-by-step example, please let me know by emailing me at lswest34@gmail.com. If you do email me, please include C&C or FCM in the subject line, so I don't overlook it!

Files:

My .vimrc (I deleted some functions that require external software or are intended for use with Mutt, so it may result in one or two settings not working):
<http://pastebin.com/wv260CJk>
The colorscheme I use (see screenshot):
<http://dengmao.wordpress.com/2007/01/22/vim-color-scheme-wombat/>
Screenshot (full-size):
<http://lswest.deviantart.com/art/Screenshot-October-2011-262486679>



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



Last month, I wrote a brief introduction to Gvim/Vim, which I hope to expand upon in this article. I recently received an email from a reader asking me if it was worth learning your way around Vim or not. The relevant points in the email were:

- a) If you use Windows during the work day, is there even a point to learning how to use Vim?
- b) If you don't program much (or at all), is Vim going to be helpful?

I sent back quite a long email, but it ultimately boiled down to:

- a) Gvim/Vim have a client for Windows, so you're more than welcome to use it there too.
- b) I find that Vim offers a set of features that makes repetitive tasks extremely easy to accomplish. This is most prominent in coding (i.e. headers, function calls, methods, formatting, refactoring, etc.), but if you do any kind of task where you find yourself repeatedly making the same change to a certain word or to large chunks of text at a time, Vim will definitely make your life a little easier.

As such, I will be covering Macros, search/replace, find, and a basic introduction to regular expressions this month.

Before I begin, please take the following to heart: Use Vim.

By that I mean simply do your day-to-day tasks in Vim for a week or so, and once you notice yourself repeating a task, do a quick search online to find out how to automate the task (or, at the very least, reduce the number of key strokes). It may slow you down at the start, but it will ultimately cut down on the time you need. The reason I say to search online is simply because, in my experience, finding the answers to questions on your own tends to improve the ability to recall the solution, as opposed to getting the information fed to you from someone.

Now, to the article...

Vim offers a lot of features, not all of which will apply to you. As such, I recommend skipping over any parts you don't expect to need

or use, in order to reduce the amount of information you need to take in. For all the following shortcuts, anything in "<>" are variables that you must decide, and anything in "[]" is a physical key on the keyboard you must press. Also, unless otherwise specified, all commands and key presses are entered in the default mode of Vim (the "blank" one).

Macros:

Vim offers the ability to create Macros on-the-fly. This means you can record a set of commands in Vim so that you can easily repeat them. The basic method is:

```
[q]<letter><commands>[q]
```

The [q] key begins the capture of a Macro, which gets saved to the letter you supply. Once you have entered [q]<letter> you can then begin using any of Vim's commands to make the necessary edits to your text. Once finished, hit the q key again (outside of any mode). An example case could be:

```
[q][b]
<series of commands>
[q]
```

This will bind the macro to the "b" key. The way to then execute a command is to enter "@<letter>", which in this case would be:

```
@b
```

As is the case with any command in Vim, you can repeat the command by appending a number before it. If you then typed "55@b" instead, it would then execute the "b" macro 55 times. Typing "@@" will also re-run the last macro. If you want to learn more about Macros, I recommend the article on the Vim Wiki: <http://vim.wikia.com/wiki/Macros>

Search:

In Vim (and programs similar to Vim, like more, less, mutt, etc.), you can search the text using the following format:

```
/<term>
```

The slash tells the program the

following is a search term (and in Vim the entire term including the slash is displayed on the bottom of the window). It will then move to the first occurrence of the word. You can move through the results using [n] to move to the next one and [shift]+[n] (henceforth referred to as [N]) to move to the previous one.

and Replace:

By default Vim supports regular expressions. This is extremely useful when replacing something (called "substitution" in Vim), because you can match the maximum number of results possible. First we'll cover normal search and replace behaviour:

```
:%s/<term>/<replacement>/
```

This will find the first occurrence of <term> and replace it with <replacement>. If you want to make this change to all occurrences, you'll need to change the command to this:

```
:%s/<term>/<replacement>/g
```

So if you wanted to replace all occurrences of "vim" with "Vim", your command would read:

```
:%s/vim/Vim/g
```

Regular expressions:

With regular expressions, you could replace all occurrences of "vim, VIM, vIm, viM" with "Vim" using the following command:

```
:%s/[vV][iI][mM]/Vim/g
```

As most of you can probably imagine, anything written in "[]" results in either possibility (or range of possibilities) being matched (henceforth called a set). You may ask yourself "why not put it all in one set?". If you do it (go ahead and try it), you'll notice that it replaces each letter with the word "Vim" instead of replacing the entire word. This is because the square brackets denote a character/position in a word. If you tell it to replace all the letters, without specifying the location within a word (which is done by splitting it into separate sets of square brackets), it will simply replace each letter.

So, if you want to match all upper, lower, and numerical cases, you could use [A-Za-z0-9]. The way

it works is that anything next to each other is taken as a new series, and anything on the opposite ends of a hyphen is a range. So your 3 ranges are: A-Z (capital letters), a-z (lower case letters), and 0-9 (numbers). If you want to match every single word that begins with the capital letter "T", you could use T[a-z]*. The asterisk tells Vim that the last set can be repeated indefinitely. Since we didn't include space in the set, it will then stop at the end of a word.

A great number of options opens up to you in this way. You can run a search for all numbers between 1000 and 9999 with [1-9][0-9]\{3\}. In this case, the braces contain a limiter (i.e. number of repetitions of the search term before it). You can also supply it as a range. For example [1-9][0-9]\{2,3\} will search for any number between 100 and 9999. You need to escape the braces with the backslash so that Vim will not include them as part of the search term.

This is just a brief overview of a few regular expressions. They can become a lot more complicated as they become more advanced. If you want to learn more, I highly

recommend this Tutorial:

<http://www.regular-expressions.info/tutorial.html>.

There are also a number of blog articles on how to wrap your head around creating expressions to do what you want.

I hope you've found this article to be interesting. I plan to continue along this path next month, with an overview of Pentadactyl (a Vim-like interface for Firefox). If you have any questions, comments, or suggestions, feel free to email me at lswest34@gmail.com. If you do email me, please include "FCM" or "C&C" (or, as a regular expression: [fFcC][cC&][mMcC]) in the subject header.



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



Due to the large interest presented in this topic by a reader, I've decided to write another one or two articles on Vim (including this one). This month I'll be focusing on a tangible example (file can be found here:

<http://pastebin.com/EqrfBFhF>). I'll cover using visual block mode, some tricks for commenting large numbers of lines, a couple of tricks for using the mouse, and copying/pasting to/from external programs from/to Vim. If you're familiar with all of these topics, you can safely skip this article.

Before we begin, I will briefly explain what an abundant number is, so that everyone can roughly follow the script. An abundant number is a number for which the sum of all factors (a factor is a number which divides a value without a remainder) is greater than the number itself. Example: The factors of 12 are: 1,2,3,4,6; the sum of the factors: 1+2+3+4+6=16; 16>12. What the script does is simply calculate which numbers (from a supplied range of numbers)

are abundant, and which are not. The function is part of my solution to an Euler Project problem.

In order to follow the article, I'd highly recommend opening a copy of the file from Pastebin in Vim (or GVim) so you can work along.

Area 1 (Commenting)

For those of you who are programmers, you'll be familiar with the concept of commenting out all code besides a small segment you want to test, when things aren't working. My approach to doing this is to use Visual Block Mode. The steps are as follows (from the beginning of the first line you want to comment):

```
<ctrl>+[v]; [j];  
<shift>+[i], [#]; [Esc]
```

The first step enters Visual Block mode, the j key acts as a down arrow key, and <shift>+[i] enters insert mode for all selected lines. After these steps, you then hit the key for the comment symbol (in Python's case it's the octothorpe, a.k.a. hash symbol). For uncommenting, check Area 2 for Deleting in Visual Block Mode.

The reader who contacted me offered the following script to do the same:

```
" COMMENTING OUT A #  
CHARACTER IN BASH SCRIPTS  
function! AddDelBashComment()
```

```
let char=getline('.')[0]  
if char == "#"  
    s/^#//g  
else  
    s/^#/g  
endif  
endfunction  
vmap <silent> # :call  
AddDelBashComment()<CR>
```

This script has to be added to your .vimrc. Once it has been added, you can call it in the following way (same process for commenting and uncommenting):

```
<ctrl>+[v]//[v]//[V]; [j];  
[#]
```

As you can see, the only thing you save by doing this is entering and exiting insert mode (and possibly having to press the control

| Format | Meaning |
|--------|---|
| <key> | Modifier key (ctrl, alt, shift, etc.) you need to press. |
| [key] | Key on the keyboard you need to press. If the key is capitalized, it is akin to me writing <shift>+[key] where key is the lowercase letter. |
| + | Combination of key (i.e. <ctrl>+[v] means press both the control and v keys). If I mean the + key, I'll write [+]. |
| // | Delimits alternative key presses |

key). I've included this script for the sake of those for whom every keystroke counts. You will need to adjust the substitute lines for each comment character you frequently use. For SQL you would replace if char == "#" with if char == "-" and s/^#//g with s/^--//g (same for the other substitution command). You must also replace the octothorpe in the vmap line, otherwise you'll be using the same key for multiple functions.

Area 2 (Visual Block Mode)

Since we covered inserting in Visual Block Mode in Area 1, I will not repeat it in this area.

Deleting in Visual Block Mode:

```
<ctrl>+[v];  
[j]//[h]//[l]//[k];  
[d]//[x]//[x]
```

Which key you use in the second step is entirely dependent upon which direction you want to go (down, left, right, up, respectively). The key in the last step is entirely up to you, they all do the same.

Yanking (copying) text in Visual

Block mode:

Section of a line:

```
[v]; [h]//[j]//[k]//[l]; [y];  
[h]//[j]//[k]//[l];  
<ctrl>+[v];  
[h]//[j]//[k]//[l];  
<shift>+[i]//<shift>+[a];  
<ctrl>+[r]+["]; [Esc]
```

<shift>+[i] inserts at the start of the line/selection, and <shift>+[a] appends to the end of the line/selection.

Copying and pasting an entire line in multiple lines doesn't work with this method (at least not for me). As such, we won't cover it. As a side-note: <ctrl>+[r]+[" works in any insert mode and pastes the contents of the Vim register (the local clipboard).

Area 3 (Mouse usage)

Just a brief tip: If you want to highlight something in Vim using Visual Block Mode, hold <shift>+<alt> as you select.

Area 4 (Copying and Pasting to/from external programs)

You may have noticed that the yank and paste methods work only within Vim. To copy text from Vim to another program (firefox, for example), you can select the text with the mouse and use the middle-mouse button paste. If, however, you are at another computer that runs a different operating system (or lacks that function), you can copy text to the system clipboard with:

```
[v]//[V]; [y];  
[h]//[j]//[k]//[l]; ["][+][y]
```

A quick explanation: you select the text you want (first two steps) and then you hit the quotation marks key (on German keyboards it's <shift>+[2]), and then the plus key, and then the y key. Do this one after the other, not all at once. Then to paste in the external program, just use <ctrl>+[v], as per usual.

Pasting:

```
["][+][p]
```

That's it. Press those 3 keys and it will paste the clipboard onto the line you have selected (so you may need to start a new line if that's what you want).

To copy text from Vim to another program [...] select the text with the mouse and use the middle-mouse button paste.

You can also set the clipboard to autoselect, which should automatically copy to the system clipboard when you highlight something, and automatically paste from the clipboard when you press the middle-mouse button.

Area 5 (Extra tips)

Syntax Highlighting:

You can enable syntax highlighting in Vim by using:

```
:set syntax=on (in Vim itself)
```

or

```
syntax enable (in your .vimrc)
```

Hide Vim in the terminal:

<ctrl>+[z] will suspend a task

COMMAND & CONQUER

into the background (tested in Zsh and Bash). Once you've suspended a task, you can re-open it using the command `fg` in the terminal.

In keypresses:

```
<ctrl>+[z]; [f][g][Enter]
```

Encrypt files with Vim:

```
vim -x <file name>
```

This command will prompt you for an encryption key before you view the file (if the file is empty/new, it will then store the password you enter).

Viewing History:

```
[q][:]
```

This will pull up a list of past commands. You can enter the number from the list in order to pull up the command, or enter `:[q]` to quit the list.

Execute system commands from within Vim:

```
[!](command)
```

An example:

```
:w !sudo tee %
```

This will save the file with sudo rights (in case you open a system file and made changes before realizing you had no rights to save the file). Vim will ask you afterwards if it ought to re-load the file, which you'll want to do.

Vimdiff:

Vimdiff is an extended version of Vim where you can open multiple files to compare them.

Usage:

```
vimdiff file1 file2
```

For horizontal split:

```
vimdiff -o file1 file2
```

For more information:

<http://vimdoc.sourceforge.net/htmldoc/diff.html>

Specify that a Tab is 4 spaces (useful for Python users):

```
set tabstop=4
```

Write this into your `.vimrc`, and any time you press the tab key, it will actually insert up to 4 spaces.

This should be more than

enough to keep everyone busy until next month. If you have any questions, comments, or requests, feel free to email me at lswest34@gmail.com. If you do email me, please be sure to include "C&C" or "FCM" in the subject, so that I don't overlook it.

My `.vimrc` file:

<http://pastebin.com/wv260CJk>

I hope you've found this article to be interesting. I plan to continue along this path next month. If you have any questions, comments, or suggestions, feel free to email me at lswest34@gmail.com. If you do email me, please include "FCM" or "C&C" (or, as a regular expression: `[fFcC][cC&][mMcC]`) in the subject header.



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



The Ubuntu Podcast covers all the latest news and issues facing Ubuntu Linux users and Free Software fans in general. The show appeals to the newest user and the oldest coder. Our discussions cover the development of Ubuntu but aren't overly technical. We are lucky enough to have some great guests on the show, telling us first hand about the latest exciting developments they are working on, in a way that we can all understand! We also talk about the Ubuntu community and what it gets up to.

The show is presented by members of the UK's Ubuntu Linux community. Because it is covered by the Ubuntu Code of Conduct it is suitable for all.

The show is broadcast live every fortnight on a Tuesday evening (British time) and is available for download the following day.

podcast.ubuntu-uk.org



Back in issue #37, I wrote about configuring an SSH server on your computer, in order to use it as a SOCKS proxy. Since I imagine not all that many people want to use it as such, I decided to focus on my second-most used command (my first-most used command is “pacman” - ArchLinux’s solution to package management). Before I get into what that command is, I’ll briefly explain why you might be interested in this solution. Specifically, it lets you sync directories and files between two computers over the LAN (and, if properly configured, over the Internet as well). I use it in order to keep my music synced between my laptop and my PC, keep my configuration files up-to-date, and to copy anything I need from one device to the other. There are a few choices of commands you could use, two of which would be `scp` (secure copy), and `rsync`. I’ll be focusing on `rsync` in this article, because it offers progress information, update features, and useful switches like `--ignore-existing`.

A few of you may be asking why I don’t just use Dropbox, an external hard drive, or a USB stick (for smaller files). The answer is quite simple: Dropbox offers a limited amount of space, and the other options require me to remember to do this regularly. If you have SSH configured on your “sender” (in this case, my PC from which I transfer the files), and an SSH client (no server required) on your “receiver” (my laptop, in this case), then you can easily write a small script to run a cron (in other words, regularly, and without any input). If you want to automate this, you will need to configure SSH to use keys instead of passwords, so that you can access your server without having to input anything. This is fairly simple (using `ssh-keygen` to create the keys, and then copying the public key to the server), and is explained in plenty of places (see the Links section below for a link to a Wiki).

Once you have SSH configured, it’s time to think about how the script should appear. I won’t supply

an example script, simply because I haven’t implemented a decent one yet. There are some things you should take into consideration when designing your script, such as:

The script should only do something if you’re on your home network (this can be done by checking the ESSID of your wireless, or, if you connect your laptop to the LAN by cable when at home, checking if `eth0` is active, or simply deciding on a specific time the script should run). The reason for this is because otherwise you’ll have lots of failed SSH connections when doing this in a location besides your home network. I recommend thinking about your habits, and finding a solution that works best for you. Then write it into an if-statement in the script.

How many files/directories do you want to sync, and which ones exactly? You can either hard-code each file or directory into the script, or create a text-based list of locations on your machines, and then use a while statement and `readline` to handle each line

separately. A few files I would recommend: `.bashrc` (or your rc file for the shell you use), `.Xdefaults` (for terminal colors), `Music`, `Pictures`, any configuration files for window managers (`XMonad`, `DWM`, etc.)

Do you want to update (meaning newest copies of the files are the ones to keep), or ignore files if they already exist on the receiver (useful for music and pictures), do you need to be recursive (that means following a directory tree). There are some other useful options to consider that `rsync` offers (see the second section of this article).

Is the destination folder and the source folder in the same location? If not, you’ll need to keep track of where each file is supposed to go (similar to #2).

Space – do you have enough space on your receiver for all the files from your sender, and, if not, what are you going to do about it? You can either reduce your list of sync files, or build a check into the

script using `df -h` to set a limit (i.e. if there are only 9GB left, stop syncing entirely, and email you/prompt you).

Once you've taken these points into consideration, it's time to write the script. I recommend you have at least 2 checks in the script (if you're connected to the right network, and if the sender is currently online). The rest of the script is entirely up to you, including when and how to run it. Back in issue #24, I wrote an article on cron, and since then have used plenty of examples, so I will only briefly discuss your options. When configuring the cron job, you can either dump the script in `/etc/cron.hourly`, or `/etc/cron.daily`. The other option is to edit your crontab (`crontab -e <username>`). In the crontab you can then create a line for the script that either runs every set number of hours/days, or set it to run at a specific time (or a specific date), and so on. I think a script that runs once a day is going to be quite enough for this.

rsync

As you can see from point 3 above, rsync offers a lot of checks

to avoid copying more files than necessary. Some useful ones are:

- `-u (--update)`: Skips files that are newer on the receiver
- `--inplace`: updates files in-place
- `--append`: adds data on to the ends of shorter files
- `-x`: Avoid crossing filesystem boundaries (i.e. stick to one partition)
- `--existing`: Do not create new files on the receiver, only update existing files
- `--ignore-existing`: Ignore files that already exist on receiver
- `--max-size=SIZE`: Don't copy any files larger than this (`--min-size` also exists, though less useful in this case)
- `--exclude=PATTERN`: Excludes any file matching the pattern
- `--exclude-from=FILE`: Reads the pattern(s) from the file
- `--partial`: Keep partially copied files

Some other useful switches for rsync:

- `--delay-updates`: Puts updated files into place at the very end.
- `-r (--recursive)`: Follows directory trees.
- `-d`: Copy directories without recursing (by default rsync doesn't enter any directory at all)
- `-l (--links)`: Copy symlinks as symlinks

- `-E (--executability)`: Keep files executable (useful for scripts)
- `-h`: Human-readable sizes and output
- `--progress`: Display a progress bar for each file

For the full list, check rsync's manpage.

The basic format for rsync commands is:

```
rsync <switches> <source>
<destination>
```

So, if I wanted to update all files from `~/scripts` on my PC with `~/bin` on my laptop, I'd write:

```
rsync -ru
lswest@127.0.0.1:/home/lswest
/scripts ~/.bin
```

This will then copy it over. Logically, you'll want to use the actual IP of your PC instead of the localhost IP, but this is only an example.

As we round off this article, I'd like to make a few notes on off-site syncing: Syncing over the Internet, while useful, should be kept to a minimum, simply because the traffic, while encrypted, will be rather large, and might cause

issues with an admin, or any kind of data limit you might have. Also, ssh keys are (generally) more secure than passwords, so I highly recommend using them wherever possible.

If there is a large influx of requests for an actual example script, I will happily deliver it next month. I do, however, recommend you try writing your own, or customize any example scripts you find to suit your needs. If you're of the opinion you'd like one, please let me know in an email (address is below). If you have some concrete questions about a script you're writing yourself, you're also welcome to email me about it.

If anyone has questions, concerns, or simply wants to share a script they've implemented, feel free to email me at lswest34@gmail.com. If you do email me, remember to include C&C or FCM in the title, so that I don't overlook it.

Links:

https://wiki.archlinux.org/index.php/SSH_Keys#Generating_an_SSH_key_pair



I realize that many of our readers are students (as am I). Depending on the country, a new semester has just started, is about to start, or, in the case of Germany, is nearly over. It's often a good idea to keep a listing (alone or in collaboration with your colleagues) of questions, and their answers, pertaining to the courses you're taking.

If you are in the habit of doing so (or are resolved to start this habit), then the following script will help make your revisions much easier. This script was written by a university professor (and FCM reader) who likes to be called "Magic Banana". You can find it on his website:

<http://dcc.ufmg.br/~lcerf/en/utilities.html#trivialibre>

It aims to ask the questions, organized in categories, in a random order, and lets you set a fixed delay to answer each of those questions. It is named "Trivialibre" - because it was initially written to ask the questions of an eponymous game (see:

<http://trivialibre.humanoidz.org>), which is a variation of the famous "Trivial Pursuit" for free software enthusiasts. The script is interesting in its various uses of the Shell tests, Zenity, the rather new 'shuf' command, and of 'sed'.

I'll briefly discuss the functionality of the script, and then explain what each command is used for (including an explanation of the uses within the script).

What does it do?

This shell script takes the questions from the files within the categories folder, and poses them (at random) in a zenity-formed GUI window. It displays the question, waits, and then displays the correct answer (much like flashcards work).

How does it work?

It uses shuf to take the files from the categories folder, shuffle the contents, and then save it in the \$XDG_DATA_HOME/trivialibre folder (so that the pending

questions can be recorded somewhere without losing any).

Once this is taken care of, a Zenity window will then ask the user to choose a category (you can choose a specific one, or have the script choose a random one).

A zenity window asks the next pending question in the chosen category. This window contains a progress bar that reaches 100% after a delay chosen by the player (the only argument of the script), or 20secs by default. Once a category has been exhausted, it is then shuffled again and moved back into the \$XDG_DATA_HOME/trivialibre folder.

Conditionals:

Throughout the script, conditionals are used in order to navigate the varying branches of the file. They are used in if-statements, while-statements, and for-statements.

```
16. if [ -n "$1" ]
```

This if-statement checks whether the argument \$1 is NOT empty (-n stands for "not zero length"), so that the delay for the progress bar can be set to something other than the default 20secs.

```
23. if [ -z "$XDG_DATA_HOME"
```

This is the exact opposite of line 16, where you check if \$XDG_DATA_HOME is empty (-z stands for "zero length"). In this case, you then set the variable to "\$HOME/.local/share" (which is the value it should be).

```
28. if [ ! -d
$XDG_DATA_HOME/trivialibre ]
```

This comparison checks to see if the directory doesn't exist, and then creates the folder (and all the category files it will need).

```
39. while [ -n
$categoryName" ]
```

Here, we use the conditional we used in line 16, but in a while-

statement. This means the loop will keep running until the variable is empty (which happens when the player clicks the “cancel” button). Once the variable is empty, the program effectively quits.

```
42. if [ "$categoryName" =  
    'Random!' ]
```

This is a simple String comparison, where we check if the user has selected Random! (as a category) or not.

```
48. if [ ! -s "$questionList"  
    ]
```

Usually -s is used to check if a file exists and is not empty. Since we're negating it (the "!"), we are actually checking to see if the file is empty, i.e. if the questions within the category are exhausted. If it is, then the next step is to re-populate the file.

Shuf:

Shuf is a program included in the coreutils package, and creates permutations (variations) of the lines within a file. It's used in the following ways in the script (the numbers before the commands are the line numbers):

```
33. shuf -o  
    "$XDG_DATA_HOME/trivialibre/$  
    category" "$category"
```

In this instance, shuf takes the contents of the file (whose name is saved in variable “category”), creates a permutation, and then saves it in \$XDG_DATA_HOME/trivialibre under the same name. This permutation is then the one used in the script.

```
44. `shuf -n 1 -e *`
```

In this case, shuf displays only the first line of the permutation (“-n 1”), and accepts the arguments as the contents of the file (“-e”). This means it takes the list of categories, creates a permutation, and selects the top one. This is the function used when “Random” is chosen as a category.

```
50. shuf -o  
    "$XDG_DATA_HOME/trivialibre/$  
    categoryName" "$categoryName"
```

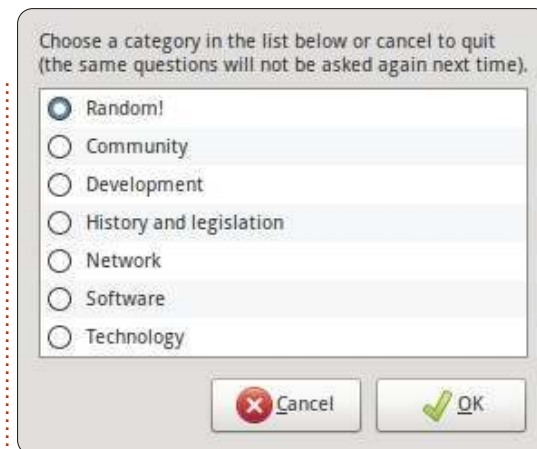
This is the same as in line 33, as it is simply replenishing the exhausted category.

Zenity:

“zenity is a program that will display GTK+ dialogs, and return (either in the return code, or on standard output) the user’s input. This allows you to present information, and ask for information from the user, from all manner of shell scripts.” (taken from the zenity manpage).

```
11. zenity --list --radiolist  
    --window-icon=../token.png --  
    height=247 --  
    title='Trivialibre' --  
    text='Choose a category in  
    the list below or cancel to  
    quit\n(the same questions  
    will not be asked again next  
    time).' --hide-header --  
    column='' --column=''
```

This creates a GTK window with a radio list, using the token.png file as the window icon. It also includes the window title, the text (outside of the radio button), and offers some configuration (such as geometry). The radio buttons are populated by piping the output of ls -1 through sed and then zenity. Sed is used to format the ls output so that Zenity can use it. It should be noted that Zenity expects each string to be preceded by a TRUE or FALSE, where it denotes which string is the default choice (the line marked TRUE).



```
51. zenity --info --window-  
    icon=../token.png --  
    title="Exhaustion of the  
    questions \"$categoryName\""  
    --text="All questions in the  
    category \"$categoryName\"  
    having been asked, they are  
    reused (in a different  
    order)."
```

This is the window that notifies the user when a category has been completed.

```
60. zenity --progress --  
    window-icon=../token.png --  
    title="In the category  
    \"$categoryName\""  
    --text="Question of  
    $author:\n\n$question"
```

In this example, zenity creates a progress bar (using a for loop to run through a percentage for the time between the question and the answer, defined earlier in the script). The percentage for the

progress bar is read in through the standard input, which is adjusted every second in this script.

```
64. zenity --info --window-  
icon=../token.png --  
title='Answer' --text="The  
answer is:\n\n$answer"
```

Once the progress bar has reached 100% and the user has clicked on the OK button, this command is then called, providing the answer to the question. It will also be displayed if the user clicks the "cancel" button, interrupting the progress bar and displaying the answer.

For a better introduction to Zenity, you can check out my article in **FCM#46**.

Sed:

Sed is a command that allows you to use regular expressions in order to modify text that you can input. I will be writing an article on both sed and awk in the near future.

```
11. sed iFALSE
```

This is used after the "ls -l" command, and inserts (hence the



"i") the word "FALSE" before every line, so that Zenity can correctly format the data into the radio list.

```
62. sed -i 1d "$questionList"
```

Here, sed edits the list of questions in-place ("-i") and simply removes the first line of the file, thereby effectively removing the question that was just asked.

How do I add categories?

You can add categories using LibreOffice Calc (or any other program which can create CSV (comma separated values) files). Instead of commas, a tab should be used as the separator, and nothing to separate text. Be sure to give the file the name of the category you want it to represent.

I hope you've found this article

interesting (and the script useful). I would like to once again thank Magic Banana for supplying the script and being extremely helpful when organizing this article. If anyone has any questions, suggestions, or requests, feel free to email me at

lswest34@gmail.com. Be sure to include "C&C" or "FCM" in the title, so that I don't overlook it.

Further Reading:

- "Linux Shell Scripting with Bash" - Ken O. Burtch (Developer's Library)
- Corresponding manpages
- FCM#46 of FCM (for Zenity)
- info coreutils 'test invocation'
- info coreutils 'shuf'
- info sed



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



Two months ago I wrote about using SSH and Rsync. This month, I'd like to introduce you all to a trick I've recently started using for wirelessly syncing my android tablet, as well as outline a few other useful tricks I use, which others might find useful. Specifically, how to display a custom date in a foreign language (in Conky), how to spruce up your Vim interface (and PS1 prompt), and a script to randomly select an image for your background every 15 minutes. I realize that GNOME can probably take care of your backgrounds for you, but someone may still find a use for it. For anyone who wants an idea of the date and PS1 adjustments, check out this month's screenshot on my deviantart page (<http://lswest.deviantart.com/#/d4se2tv>)

Android

The things you'll need for this are as follows:

On the Android device:

QuickSSHd (as far as I remember, it's free). Once it's installed, simply check the IP address of the device, and give it a password/port. To start the server, hit the button that says "off" (so that it reads "on").

On your PC:

sshfs, and a mount-point (preferably in your home folder) for the device.

A brief explanation of the process:

I simply mount the sdcard folder using sshfs, and then copy files over as I would using USB (which never worked very well for me and my Asus TF101 Transformer eeePad for media files). As for how to do this, see below.

Mounting can be done with:

```
sshfs root@127.0.0.1:/sdcard/
~/tf101 -p 2222
```

Just like with rsync, you need to specify the path after a colon, and

to avoid input/output errors on the mounted folder, you must have a closing slash at the end of the path. The second path is the mountpoint, and the -p 2222 is simply the port that QuickSSHd is listening on.

After this, copy the files to the folder with whichever method you prefer.

As an optional final step, I find that my Music app does not refresh after these files are added. The only way I've found to fix this is to go to the File Manager, and rename the copied file/folder, forcing the tablet to re-load the information.

Custom date

I've been learning Japanese for a while now, and decided to display the date in Japanese (both for practice, and to save space). However, it was not so easily done, as I use Conky for my status bar in XMonad. My solution to this is to write the following code-segment, which I call every few seconds from my conkyrc. You may need to run the following command to get the

right formatting (this applies to any locale you may decide to use).

```
locale-gen ja_JP.UTF-8
```

Code-segment:

```
#!/bin/sh
```

```
LC_ALL="" LC_TIME=ja_JP.UTF-8
date +%A, %-d %B%Y %H:%M'
```

You'll of course need to change the date format into the string you'd like (thoroughly documented in the man page).

Vim & PS1 (Powerline)

Powerline homepage:

<https://github.com/Lokaltog/vim-powerline>

The easiest way to install Powerline is to use vundle. Instructions for installing Vundle are on the github page, here: <http://github.com/gmarik/vundle>. Once you've installed it, you'll need to add the following three lines to your .vimrc:

```
set
rtp+=~/vim/bundle/vundle/
```



```
call vundle#rc()
```

```
Bundle 'Lokaltog/vim-  
powerline'
```

After which you'll need to open vim and type :InstallBundle (requires git to install)

Once you've installed it, you'll need to patch the font you're using with the script found under the folder fontpatcher (full path will be something like ~/.vim/bundle/vim-powerline/fontpatcher). The readme file there will guide you through the patching.

Once your font is patched (and your font-cache reloaded with the command fc-cache), you can add the final line to your .vimrc to get arrow shapes.

```
let  
g:Powerline_symbols='fancy'
```

If you find this to be something you want for your prompt as well, you simply need to make sure you're using the patched font, and copy the arrow from the vim line into your .bashrc prompt line, which I sadly can't supply, as it would require a patched font. Make sure you set the background

behind the arrow glyph to the next color, and the foreground to the previous, to achieve the arrow appearance.

Wallpaper

This script, as I have it written, requires fluxbox to be installed (for the fbsetbg program). However, if you know of a command-line wallpaper setter that offers a random option, feel free to replace it.

The script:

```
#!/bin/bash  
  
while true  
do  
  
    fbsetbg -R  
    ~/Pictures/Hyperion/Wallpapers/Guitars  
  
    sleep 15m  
  
done
```

The script runs in an infinite loop, calling the random command on my guitar wallpaper folder every 15 minutes. If anyone has any adjustments (or improvements upon the script), feel free to send me an email about it!

Hopefully some of you will find some of these tips useful (and hopefully I've interested one or two readers into using Vim a bit more). If anyone has any questions, comments, or suggestions, I can be reached at lswest34@gmail.com. Any emails should include "C&C" or "FCM" in the subject line, so that I don't overlook it. Happy customizing!



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



The Ubuntu Podcast covers all the latest news and issues facing Ubuntu Linux users and Free Software fans in general. The show appeals to the newest user and the oldest coder. Our discussions cover the development of Ubuntu but aren't overly technical. We are lucky enough to have some great guests on the show, telling us first hand about the latest exciting developments they are working on, in a way that we can all understand! We also talk about the Ubuntu community and what it gets up to.

The show is presented by members of the UK's Ubuntu Linux community. Because it is covered by the Ubuntu Code of Conduct it is suitable for all.

The show is broadcast live every fortnight on a Tuesday evening (British time) and is available for download the following day.

podcast.ubuntu-uk.org



As a follow-up to last month's article, I decided to supply a few more tips and tricks that mesh well with previous articles. I'll cover (briefly) how to configure Conky for Japanese, a reader's solution for ToDo lists, and a useful script for displaying your current terminal colour scheme.

Displaying Japanese in Conky is fairly simple, and consists of the following three steps:

- Override any UTF-8 locale with the setting "override_utf8_locale yes" (minus the quotes)
- Enable XFT (X FreeType) fonts using the setting "use_xft yes" (minus the quotes)
- Then choose a font that supports the language, for example: "xftfont VL Gothic:size=10" (minus the quotes).

This is useful if you, for example, have Japanese in your ToDo list (if it's displayed by conky), in case any of your scripts return Japanese characters, or if you use Japanese in any other way within conky.

A reader, John, recently explained his ToDo list setup to me (while suggesting I write the above Conky tips in an article: done and done), and I thought it was a very versatile way to go about it. As I've explained in the past, my system simply utilizes a python script and a folder of reminder files, meaning I can update it on-the-fly with any text editor. The script John uses, however, offers an Anroid and iPhone app, which is a bit easier to do quickly. The script he uses is called "todo.txt", and is written by Gina Trapani (her homepage is: <http://ginatrapani.org/>). Combined with the following script (that he found on the Ubuntu Forums – I was unable to find the exact thread, otherwise I'd have linked to it), you can have a nicely formatted list in Conky. The script is here: <http://pastebin.com/FnHGkcXw>. Once you've added some items to the todo.txt file, you need to simply test the script (and possibly adjust the path to todo.sh), and then add it to conky using, for example, `${execi 60 todo-list}` (which runs the script every 60

seconds).

The colours script I use is the last script on this page: <http://tldp.org/HOWTO/Bash-Prompt-HOWTO/x329.html>. It essentially writes the colour codes to the terminal, resulting in a table of the colours, helping you when you're either customizing your Bash prompt, or else when you're trying to find colours that you like. I save this script in my scripts folder as "colors" and defined an alias within my .bashrc for "sh \$pathToColors" (replace \$pathToColors with your actual path), to save myself having to write it every time. You can also create a symbolic link in /usr/bin/ or add your script folder to your \$PATH, but seeing as the colors only matter for your terminal, it's simply easiest to create an alias. The script looks something like the image shown right.

For anyone interested in the actual colors I'm using, they can be found here: <https://github.com/lswest/dotfiles/blob/master/.Xdefaults>

I hope that this information has been useful, and, as always, if you have any questions, comments, or suggestions, you can email me at lswest34@gmail.com. If you do email me, please include "C&C" or "FCM" in the subject line. I've been debating whether or not I should do an article of C&C on my configuration files and favourite programs – if any reader is interested in reading that, please send me an email and let me know.



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



In last month's article, I posed the question whether or not readers would be interested in a thorough article covering my configuration files. The response I had to this question made me feel like there may be some interest in the files, and so I will be posting links to my configuration files, and explanations of what they do and why I need them. The scripts are in my github here:

<https://github.com/lswest/dotfiles/tree/master/C&C> (this folder is not going to be updated unless I have to make corrections – my actual current files will always be in master/).

An example screenshot of my system:

<http://lswest.deviantart.com/art/March-2012-Screenshot-289550803>

I won't be going in-depth about all the options I use in my files, since it would result in a very, very long article, and most of it wouldn't apply to every reader. Instead, I feel most of the files are clear (and, where I have discussed them in previous articles, I have

referred back to them). Any specific questions can always be directed to me at the email address listed below.

Window Manager

The `.xmonad.hs` is the configuration file for my window manager of choice (XMonad), and is unimportant for anyone who isn't using it. I combine this with the `.conkyrc_dwm_bar` file to create my task bar (it was originally used in DWM, hence the name).

General terminal settings:

The rest of the files are more general. The `.Xdefaults` file contains my colors and settings for `rxvt-unicode`. I also use `Zshell`, so my `.zshrc` file is a bit more detailed than a similar `.bashrc` file. I do, however, use the `Vim-Powerline` symbols (see issue #59), so be aware of that if you carry over my `PS1` prompt.

Conky files

(all lua scripts are in `scripts/`):

The `.conkyl_mpd` file creates a lua-based conky instance with

Music Player Daemon information. This could be adapted, but it's essentially useless to anyone who doesn't use MPD (though, I don't see a reason why you wouldn't use MPD!).

The `.conkyrc` file is my “main” conky instance with a bit of system information in it. It uses a lua file to create a smooth semi-transparent background, but could easily be used without it.

The `.conkyrc_reminders` file contains a front-end for my python To Do scripts (see issue #46). It formats text documents into a bulleted list using the file names for the section names.

Further scripts include:

`Wallpapers.sh`, which selects a random wallpaper from a specific folder every 15 minutes (easily extendable).

Mpd-info: a script to yank music state & song information, and format them for a single-line conky.

Dvol: a script to adjust volume (including an on screen display using `dzen`)

volstate: A volume state script that pulls information about the volume (muted, or a percentage of total volume if unmuted).

Mpd-cover: A script (not created by me) that downloads album art for a song using information from MPD, and saves it into a `.covers` folder. Works fairly well, unless you have weird symbols or exotic music.

Packages-short: An indicator script for new updates (using `pacman`, an alternative with `apt-get` would be:)

jDate: a customized Japanese-language date and clock that I use with conky and `dzen` as a taskbar.

That essentially sums up my list of configuration files. It may seem like a lot (or very little, depending on what you're used to), but it covers a rather wide range of settings. Occasionally, I find myself



tweaking a single file or adding a small script to solve a particular issue I have, but otherwise I'm quite content with the current setup. I also find this combination to be the most efficient I've used to date. So, for anyone who's looking for a way to work more efficiently, I highly recommend using conky to display relevant information on the desktop, and a tiling window manager for actual work.

If you have any questions, comments, or suggestions, feel free to email me at lswest34@gmail.com. If you do

email me, please include "C&C" or "FCM" in the subject line, so that I don't overlook it. For anyone who does use one of my configuration files, consider letting me know if you find a new use for it (or adjust it any way). Who knows? Maybe someone else will find it useful – or maybe I've been trying to do something similar for a long time.



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.

CODEWORD

Every number in the codewords grid is 'code' for a letter of the alphabet. At the end you should have a different letter in each numbered box, and a word in English in each of the horizontal and vertical runs on the codeword grid.

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 22 | 5 | 3 | 19 | 25 | 2 | 26 | 1 | | 24 | 22 | 11 | 8 | | | | | | | | | | | | | |
| | 3 | | 25 | | 24 | | 19 | | 10 | | 3 | | | | | | | | | | | | | | |
| 2 | 18 | 14 | 3 | 19 | 21 | | 22 | 3 | 11 | 24 | 2 | 9 | | | | | | | | | | | | | |
| | 11 | | 21 | | 19 | 6 | 19 | | 21 | | 15 | | | | | | | | | | | | | | |
| 22 | 11 | 18 | 14 | | 14 | | 21 | 26 | 26 | 20 | 24 | 2 | | | | | | | | | | | | | |
| | 24 | | | | 24 | | | | 2 | | | | | | | | | | | | | | | | |
| 25 | 2 | 19 | 11 | 24 | 25 | | 7 | 2 | 24 | 24 | 22 | 14 | | | | | | | | | | | | | |
| | | | 18 | | | | 21 | | | | 14 | | | | | | | | | | | | | | |
| 22 | 19 | 21 | 19 | 19 | 16 | | 19 | | 19 | 13 | 26 | 17 | | | | | | | | | | | | | |
| | 16 | | 1 | | 19 | 22 | 22 | | 15 | | 2 | | | | | | | | | | | | | | |
| 22 | 11 | 4 | 18 | 1 | 10 | | 4 | 3 | 2 | 2 | 19 | 8 | | | | | | | | | | | | | |
| | 21 | | 22 | | 18 | | 24 | | 24 | | 15 | | | | | | | | | | | | | | |
| 23 | 24 | 22 | 14 | | 16 | 19 | 22 | 14 | 24 | 2 | 24 | 25 | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | | | | | | | | | | | | | |
| | | | | Q | | | | | X | | | | | | | | | | | | | | | | |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | | | | | | | | | | | | | |
| | | M | | | | | | | | | | | | | | | | | | | | | | | |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |

Solutions are on the second last page.

puzzles are copyright, and kindly provided by,
The Puzzle Club - www.thepuzzleclub.com



COMMAND & CONQUER

Written by Lucas Westermann

Unison

Before I begin with the actual article, I thought it might be useful to address a question I got via email. A reader was asking where best to place my `wallpapers.sh` script, and, in doing so, reminded me that I haven't ever explained how best to go about this. My approach is to store all my scripts in a single folder (either a folder called "scripts," or, if I want to save some visual space in my home folder, I'll call it `.bin` or `.scripts` so the folder is hidden). Make sure any script you place here is executable (`chmod +x`), and that they specify in the first line which interpreter is required (`#!/bin/bash`, `#!/usr/env python`, etc.) - otherwise calling it will not work. Once that is done, you could create a symbolic link to the script in `/usr/bin` with

```
sudo ln -s /path/to/script
/usr/bin/script
```

However, this can be complicated if you start removing scripts (it will leave broken links in your `/usr/bin` directory). Another

method is to add the folder into your `$PATH` variable. This can be done with

```
export PATH=$PATH:<folder>
```

(replace `folder` with the actual path to the folder you created). To make it permanent, you can either place it in your `.bashrc`, your `.xinitrc` (if you start your interface with `startx`), or in any other script that gets called when you log in. I've had some hit-and-miss experiences with this, but generally `/etc/environment` works well. Since, however, the `PATH` is created in `/etc/profile`, you can also simply add the path to the scripts folder at the end of this path. I recommend avoiding any files in `/etc/`, and instead organize all your scripts within `$HOME`. This is due to the problems that can arise when creating system-wide settings using files that not all users can access. The problem is that anything in `$HOME` is restricted access wise to your user account. And now, on to the actual article.

The rest of my family has recently moved over to MacBooks for mobile workstations, and, for most of them, keeping files organized is fairly easy. For one of them, the MacBook is the only computer, and, for the other, it's a work machine where only company-related files need to be synchronized. However, for the latter person, whose PC runs both Arch Linux and Windows, it's a little more complicated. At first, I thought I'd use `rsync` in a custom script to synchronize the folders one after the other, but, since changes may be made on either machine, I needed a robust method to synchronize them (including deleted files). `Rsync` may be able to do this, but I have yet to get it working at 100%. As such, I began looking at `unison` again, which has resulted in me adopting it for my backup needs as well. Below, I describe my method for configuring `Unison` (which is available from the universe repository).

From the `Unison` manual
"Unison is a file-synchronization

tool for Unix and Windows. It allows two replicas of a collection of files and directories to be stored on different hosts (or different disks on the same host), modified separately, and then brought up to date by propagating the changes in each replica to the other."

To synchronize across multiple machines, the first thing you need to do is configure `SSH` on one of the machines (I did this on my mother's PC, to minimize any possible security holes which would arise from a laptop with `SSH` enabled in public networks). It is also extremely important that you enable key-based logins on `SSH`. This is fairly easy with the following two commands:

```
ssh-keygen -b 521 -t ecdsa
-C"$(id -un)@$(hostname) -
$(date -rfc-3339=date)"
```

This creates the key. Follow the on-screen instructions. Keep in mind that entering a passphrase requires you to type this in when you use it - which defeats the point of a key-based login.

```
scp ~/.ssh/id_ecdsa.pub
username@remote-
server.org:~/.ssh/authorized_
keys
```

This will copy the file into the authorized keys of the server, meaning that if you try `ssh <IP>`, without a username, it will automatically log you in using the key, without prompting for a password. Note: If your version of `ssh` doesn't support `ecdsa`, `rsa` keys are fine too.

Now then, on to Unison. The easiest way to manage multiple synchronization folders is to create multiple profiles (.prf files). Since the options are all the same, I will cover only a single example. My Music sync profile looks like this:

```
# Unison preferences
label = Music sync
root = /home/lswest/Music
root =
ssh://user@IP//home/lswest/Mu
sic/Hyperion/
fastcheck = true
dontchmod = true
ignore = Name *.ini
ignore = Name *.jpg
ignore = Name *.jpeg
sshargs = -C
```

From top to bottom, the settings do the following:

label – assigns the name of the profile, for the GUI's list of profiles.

root – These are the sender/receiver (in that order). It shouldn't be possible to use more than 2 roots, though I haven't tested this.

fastcheck – I find that this reduces the search time immensely. It is supposed to be the default for Unix systems, but it doesn't hurt to be certain. The way this works is that it reads a file's modification times during the first pass, so it can ignore any files that weren't changed (meaning the modification times are the same as what was stored in the database). On the second pass, it generates a fingerprint, and compares it to the last contents to be synchronized.

dontchmod – Unison tries to keep permissions the same, which is fine for syncs with Linux/Unix systems on either end, but for anything stored on NTFS (or with a Windows server), you will need to turn this off, to prevent dozens of permission warnings. Leaving this option isn't bad, as the permission settings simply fail, but it does slow down the transfers.

ignore – This option tells Unison what files to ignore. In my case, I ignore all the .ini files that Windows is so fond of creating (since my media is on an NTFS hard drive shared between Windows and Arch on my PC), and any of the cover art, since I use a different method in Linux.

sshargs – passes arguments to the `ssh` session. The option “-C” tells SSH to compress any of the information being sent, theoretically reducing transfer time per file (I haven't tested to see if this slows the entire process down, since the files need to be compressed, but for my rather large Music folder, it doesn't take too long to synchronize the changes these days).

Keep in mind that the first run-through will take a while, since Unison has to assume every file is new.

The first few times you run Unison, I highly recommend manually checking the sync settings for each file that pops up (options are: right to left, left to right, skip), to make sure you're not losing/gaining any files you don't want. Once you're satisfied with

how Unison is handling it, you can configure it to automatically accept non-conflicting changes. Do this by adding “`auto = true`” to the .prf file. You can also have Unison accept the non-conflicting changes, while skipping any conflicts with “`batch = true`”. Whether or not you can make this a completely automatic process, I'm not sure. However, you can minimize the input that is required using the above steps. To find out more about Unison, check the Further Reading section for a link to the user manual. If you want to simply mirror a folder in one direction (i.e. copy it from one machine to the next, without synchronizing), then I would still highly recommend `rsync` instead, due to the fact that it doesn't require any input when run properly. It's also extremely well documented in the manpage, making it quite easy to use. As a final note, I want to point out that both Unison and `Rsync` work for local folders too, meaning you can use it to create backups on a USB drive as well. If you decide to run these commands automatically on a schedule, I would recommend creating a script to first check if the USB drive is connected, or to check that you have the correct

Unison

remote IP address (or can ping the router's IP). This is to avoid using resources to run a command that will simply fail. It also avoids opening SSH connections with computers that have the same IP but aren't your intended target, which could lead to problems down the line.

I hope at least some of you have found this interesting or helpful. I'm going to continue expanding my preferences files as I get more comfortable with Unison, and will mention any further tips I discover as time goes on. As always, if you have any questions, suggestions, or comments, you can reach me via email at lswest34@gmail.com. If you do decide to email me, please put "C&C" or "FCM" in the subject line, so that I don't overlook it.

Further Reading:

Unison Manual:

<http://www.cis.upenn.edu/~bcpierce/unison/download/releases/stable/unison-manual.html#usingit>

Ubuntu Wiki page on SSH keys:

<https://help.ubuntu.com/community/SSH/OpenSSH/Keys>



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.

CODEWORD

Every number in the codewords grid is 'code' for a letter of the alphabet. At the end you should have a different letter in each numbered box, and a word in English in each of the horizontal and vertical runs on the codeword grid.

| | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|----|----|----|----|----|
| 7 | 5 | 22 | 21 | 1 | 26 | 5 | 16 | | 2 | 10 | 24 | 17 |
| | 26 | | 10 | | 5 | | 2 | | 15 | | 22 | |
| 3 | 17 | 16 | 18 | 1 | 26 | | 25 | 20 | 9 | 2 | 20 | 26 |
| | 17 | | 2 | | 18 | 10 | 15 | | 26 | | 26 | |
| 6 | 2 | 21 | 17 | | 21 | | 17 | 21 | 15 | 22 | 9 | 26 |
| | 20 | | | | 2 | | | | 5 | | | |
| 2 | 17 | 21 | 5 | 26 | 15 | | 19 | 2 | 16 | 26 | 25 | 15 |
| | | | 26 | | | | 10 | | | | 20 | |
| 12 | 26 | 25 | 4 | 8 | 10 | | 23 | | 21 | 11 | 22 | 4 |
| | 8 | | 22 | | 21 | 10 | 25 | | 11 | | 21 | |
| 4 | 26 | 15 | 7 | 22 | 5 | | 26 | 5 | 22 | 13 | 22 | 15 |
| | 15 | | 5 | | 10 | | 15 | | 17 | | 26 | |
| 14 | 16 | 5 | 26 | | 17 | 21 | 2 | 25 | 21 | 26 | 17 | 21 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| | | | | | | | | | | | | |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| | | | | | | | | | | | | |
| A B C D E F G H I J K L M N O P Q R S T U V W X Y Z | | | | | | | | | | | | |

Solutions are on the second last page.

puzzles are copyright, and kindly provided by,
The Puzzle Club - www.thepuzzleclub.com



COMMAND & CONQUER

Written by Lucas Westermann

In the past two months I've been involved in group work at university. At first, we had to simply organize a group of five people (of which, ultimately, only three did any work). We managed this quite easily with a single shared Dropbox folder, and (for group writeups) a Google Doc file (shared with the "anyone with the link can edit" setting). However, as the project progressed we had to work first with a single other group, and then 5 other groups. Meaning at the end of the project, we were trying to manage 30 different people (and a smattering of operating systems, and people of varying degrees of technical proficiency). Generally, for any kind of group work where multiple people might edit the same file (likelihood of this increases as more people join the project) you should use Git. However, this can quickly become complex, and requires a certain level of technical proficiency to avoid unnecessary management by the repository owner. As such, I thought for this month's article I'd cover a few ideas I've had for

managing data between groups. At the end of the article, I also run through a few useful tips I've stumbled upon for learning/practicing Japanese.

The "Cloud"

This includes anything like Dropbox, Ubuntu One, Google Drive, etc. This is by far the easiest method, and works like a charm for groups of 2 to 10 people, in my experience. Once you're past that upper limit, or you need to allow anyone to access the shared files, it's no longer a viable option. The reason for this is simple – it's designed to be user-friendly, and forgoes some of the more advanced management features (robust version control, merging options, public access with merge-back capabilities, and so on).

For anyone concerned about security, it's possible to create a Truecrypt volume and store it in the cloud (in Dropbox at least, I haven't tested this with any of the others). Offering the passcode for the volume to the other group members should be done in person

or via phone for maximum security.

Revision Control

This is by far the most versatile system I've used to date. You can use something like Github to create and manage your repository, and it is publicly available for forking (creating a copy of it) for other people to work on. Once the changes have been made, they will be merged back "upstream" after which the repository owner will need to go through the commits/merges and accept them into the final version. Each merge is saved as a version, allowing you to easily move back to an older revision if necessary. I used this to great success last year when I wrote a LaTeX-based script for a mathematics lecture at university, allowing anyone in the class to make changes or corrections to the files. By the end of the semester our script was often more thorough than that of the professor himself.

Security-wise I'd list this as one of the "best" options, simply due to the fact that any change has to

be approved by someone who's in charge. It's also possible to create a custom server on a personal network for use with svn or git, for completely in-house solutions.

Single File Group Work

By this I'm referring to collaboration on a single file, which needs to be (or simply will be) edited by multiple people at once in real-time. For this, I've not found anything that does it as easily as Google Docs. If, however, anyone is against Google for such things and has access to a server, you could create an implementation of Apache Wave or Walkaround, which are based off of Google Wave. They should offer a similar real-time collaboration option, but with a more limited range of document formats.

Security is on-par with Dropbox, in my opinion. This is especially true if you require every collaborator to have an account, as opposed to the "anyone with the link can edit" option I mentioned above.

If anyone has used a program or system that they think deserves to be mentioned, please let me know via email (address is below), and I'll be sure to mention it at the beginning of my next article. Please mention anything like cost, availability, etc.

Language-learning Programs

For anyone who owns an android phone/tablet, I highly recommend Human Japanese, which has been one of the best explained introductions to Japanese I've read. It doesn't, however, use any Kanji, relying instead on Hiragana/Katakana. The benefit of this is that you can build up vocabulary faster (since you don't need to tell Kanji apart or learn them), and also that, if you're aiming for a solely verbal communication level, you don't learn anything unnecessary. It also offers great tips for tourists when visiting Japan. At the time of writing, it's available in the Google Play store for 6,99€. Also worth mentioning is that there is a Mac OS X version of the same application available in the App Store, for anyone who owns a Mac. Also, the iOS App Store offers

Human Japanese, albeit for 7,99€.

Flashcard programs of any sort are extremely useful for remembering vocabulary. I can't name any specific programs, since I haven't used any recently. Trivialibre might also be a valid option for doing this (introduced in Command & Conquer from Full Circle Magazine Issue #58).

Book-wise, I've read and used Japanese in Mangaland 1 & 2 (and the corresponding workbooks) by Marc Bernabe. The books are well-written, and the illustrations help underscore the main points of every chapter. The illustrations also help your ability to read hand-written (or at least, the style of writing used in manga) Japanese, which may be useful in the future. The workbooks, as well, help drill the vocabulary and grammar into your head. The stories used in the books are also well-drawn and written, for anyone who is interested in that sort of stuff.

Listening to as much Japanese as possible helps to improve your own pronunciation. This can be either music, audiobooks, or television shows. If you decide on the television/movie route,

definitely watch it with subtitles, as you can also start linking meanings to phrases. Finding Japanese books you find interesting (manga, Shogi books, novels, etc.) will also help you improve, once you're capable of reading basic sentences. If the book truly interests you, it will help balance out the tediousness of having to read while having a dictionary close at hand. Also, for dictionaries I highly recommend electronic versions (gwaiei, for example). The reason for this is simple – they offer stroke-based input/searches, meaning you don't need to be able to recognize the Kanji, or even read it, if you can figure out the stroke order. Paper dictionaries tend to require both these things.

Last, but not least, regardless of what language you're learning, it's a wasted effort unless you use it! To this end, I present to you Lang-8: <http://lang-8.com/>. This is a website where each user gets a journal, in which they can write entries, in both the language you're learning and your intended meaning in your native tongue. The journal is then corrected by other users. Each correction is split by sentence, and can be commented

on by the person correcting it. This means that you'll be getting the opinion of a native speaker on how to speak his or her language in the most natural way. This is a wonderful option for anyone who doesn't have friends who speak the language they're learning, don't want to bother their friends with constant questions, or who don't have the option to go to that country and “jump into the deep end”, as it were.

As always, I hope you found this article helpful. I do realize that the two topics I covered in the article are not at all connected. However, this means that there should be something in here for everyone. If you have any questions, comments, or suggestions, feel free to email me at lswest34@gmail.com. If you do email me, please put FCM or C&C in the subject line, to make certain I don't overlook it.



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



This month, I thought I would move away from the tutorials for once, and instead introduce my readers to the concept of an Integrated Development Environment (IDE), and explain a few of the reasons why an Integrated Development Environment could be useful for programmers. Following this, I will name the IDEs I use, and also name the languages for which I don't feel I need an actual development environment.

What is an IDE?

An IDE is basically an editor that also contains a compiler/interpreter and easy access to documentation, and usually offers plug-ins for

debugging and testing. Some of the more advanced ones also offer various forms of auto-completion.

Why use an IDE?

- Learn languages
- Makes managing larger projects easier
- Easy access to compiler/interpreter, debugger, and testing environments

Using an IDE can make it much easier to start using a language, as it can offer corrections for typos (or point out when you use function calls from a different language). If your IDE doesn't highlight mistakes, it will at least make it easy to quickly compile/execute the code to check for exceptions and run-time errors.

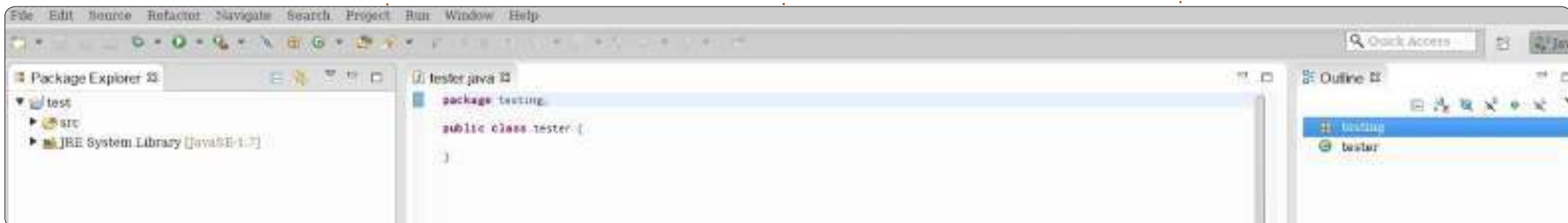
When you're working on larger projects, it's usually helpful to use an IDE that offers an integrated file tree to keep track of what files are where, and because they offer tools to make debugging easier. Once you have more than 100-200 lines of code, I would generally not recommend trying to do your debugging manually, and instead use an actual testing and debugging framework.

The sole exception to this, for me, is Java. I always try to use Eclipse with Java where possible, simply because a number of functions have long names, and auto-completion can make life easier. An alternative to this is simply redefining functions into shorter names, but that can get extremely complicated. Since I'm

already in Eclipse when working with Java, I tend to use the integrated debugger regardless of file size. Using something like JUnit is overkill for most basic projects – I use it only when I have multiple files of a few hundred lines of code – or if I'm required to do so for class, or work.

What IDE should I use?

Ultimately, it comes down to what you want to use. There are some very advanced environments like Eclipse (for Java or C++), which will take up more space than a basic IDE. The current version of Eclipse requires just over 200MB of space in Linux. Keep in mind that the more features you have running at a time, the more RAM is used. Due to this, I find Eclipse too



unwieldy. I use it with Java, and Java alone, simply because that's what my university required, and as such it's what I'm used to. Netbeans is another option for Java/C/C++/PHP development. However, I tend to not write many large programs, and I find a text editor and a terminal to be sufficient for C, C++, and PHP. Similarly with Python, although I tend to use the interactive interpreter for testing ideas before I start coding.

The bottom line is simple. Think about what you want to have easy access to, how large your programs usually are, and weigh the pros/cons of learning a new interface. Also, if you have time-

constraints, you may want to use an IDE, as they can cut down development and testing time (so long as you are comfortable with the interface). Once you've picked out your reasons, find a few IDEs that fulfil them, and try them out. Repeat this until you find one you're comfortable with (or that meets your needs perfectly).

Light Table: A new IDE concept

I recently saw a post by Chris Granger on his take on an IDE called Light Table. For a link, see the Further Reading section below. Since that post, he has successfully funded the project on Kickstarter,

promised Python support after Clojure, and released a prototype (the "Light Table Playground"). I'm extremely interested in the final product. He's introduced some new ideas (such as his "light" workspace), and offered extremely appealing implementations for all other ideas. I've been using the Light Table playground since it came out, and I find the interface to be cleaner than any other interface I've ever seen. At the moment, most of the functionality in the concept video is unavailable, offering mainly Instant Feedback at this time. However, for anyone who's fond of Clojure, or just wants to take a look at something new, I would recommend having a look.

I hope some of you have found this article interesting, and that I have answered at least a few of your questions concerning IDEs. If you have any more questions, comments, or requests, you can email me at lswest34@gmail.com. If you decide to email me, please include "FCM" or "C&C" in the subject line, so that it doesn't get lost in my inbox.

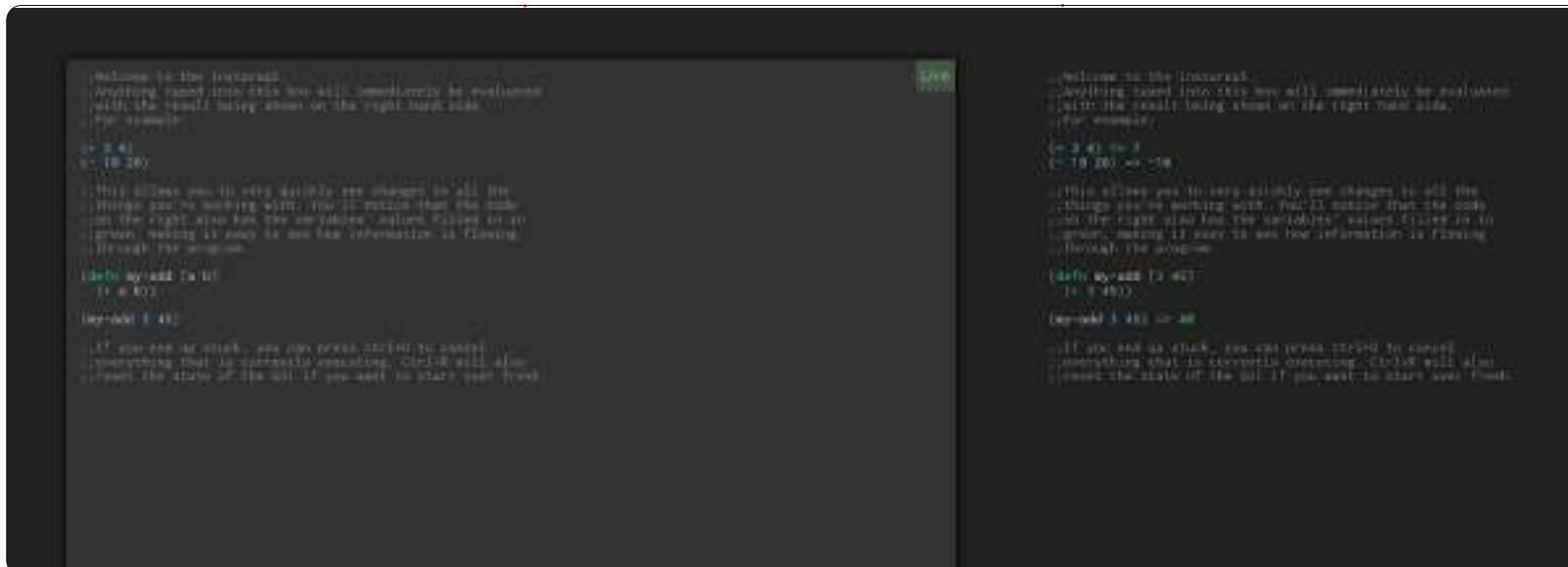
Further Reading:

<http://www.chris-granger.com/2012/04/12/light-table---a-new-ide-concept/>

<http://app.kodowa.com/playground - Light Table playground>

<http://www.eclipse.org/downloads>

<http://netbeans.org/>



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



These past few weeks I've gotten hooked on Guild Wars 2, which was released on August 28th. However, as I'm on vacation, I was placed in the position of having to rely on Wine.

I tend to prefer running games natively. However, I've gotten it working fairly well via PlayOnLinux (the only exceptions are the embedded webpages for the Black Lion Trading Company and looking with the mouse, both are solved using patched versions of Wine). Long story short, if you want to play the game in Wine, keep on reading. For anyone who isn't aware of it, Guild Wars 2 is a Massively Multiplayer Online Roleplaying Game from ArenaNet. It offers 5 different classes, 8 professions (called classes in most MMOs), and features such as dynamic events (as opposed to classes, these are events that spawn semi-randomly over the map, giving you a more immersive game), and a personalized story for each character you create. For anyone skeptical about it: I've

started about 8 different characters since the 3rd Beta Weekend event, and each one has had a significantly different story. The game is buy-to-play (as opposed to free-to-play or pay-to-play). What that means is that once you buy the game, you own it and can play it (like most non-MMO games). The game is currently going for between 45 and 65€ (60-80 USD).

Versions of Wine I used: 1.5.9-raw3 originally (the raw input patch is required to be able to rotate the camera via mouse). However, at the time of writing, PlayOnLinux also offers a -guildwars2 series of Wine versions, that include patches for raw input and the embedded webpages (see this bug report: http://bugs.winehq.org/show_bug.cgi?id=27168#c41). I get slightly fewer FPS with this version of wine (1.5.12-guildwars2), but I haven't been using it for long. There are numerous versions of -guildwars2 patches, so some older versions may offer slightly better performance. According to the

PlayOnLinux forums, 1.4.1-guildwars2 offers the best performance, and 1.5.11 can lead to issues with alsa. Anyone not using PlayOnLinux can patch their version of Wine with the corresponding patches.

The screenshots were taken at the highest graphical settings possible on my laptop, but they were by no means the maximum settings available.

The easiest way to set up the Wine installation is to use PlayOnLinux, and, after enabling the testing repository, installing

Guild Wars 2 via the interface. If, however, you dislike using PlayOnLinux, you simply need to download the installer and execute it. It will begin a download, roughly 10MB, if I remember correctly. Once the download finishes, you may be greeted by a black screen—if this is the case, you'll need to work blind for a second. The button you need to press is in the lower-left quadrant of the window. It was roughly 1/5th of the way from the bottom, and 1/5th to 1/6th of the way from the left of the window for me. It will pop open a new window asking you for the install location. I wouldn't



recommend changing the default path too much, but so long as it ends up in the correct Wine folder, it shouldn't be an issue. Once you've selected the path, hit the "okay" button. In newer versions of the installer, the button may be the direct "install" button, but if it says "okay" and nothing happens, the install button should be in roughly the same spot as the first button, but on the other side of the window.

For anyone not interested in fighting with a black screen, you can easily start the installation from a Windows PC, and copy the resulting Guild Wars 2 folder. It also works in a Virtual Machine. I wouldn't recommend letting the game patch in a VM though, since the resulting folder can get rather large. Once you've gotten the Gw2.exe, Gw2.dat, and Gw2.tmp, you're basically ready to patch your game.

In order to patch your game, you'll need to create some sort of launcher, so that you can pass an argument to Gw2.exe. If you're using PlayOnLinux, you can do this by choosing Configure, and adding -dx9single to the arguments field. This stops the black screen issue

(and also the perpetual connection attempt). If you're using plain old Wine, you'll just need to either create a .desktop file (I haven't tested arguments in these sorts of files, but it should work), or else a bash file that looks something like this:

```
#!/bin/bash
wine $PATH/Gw2.exe -dx9single
```

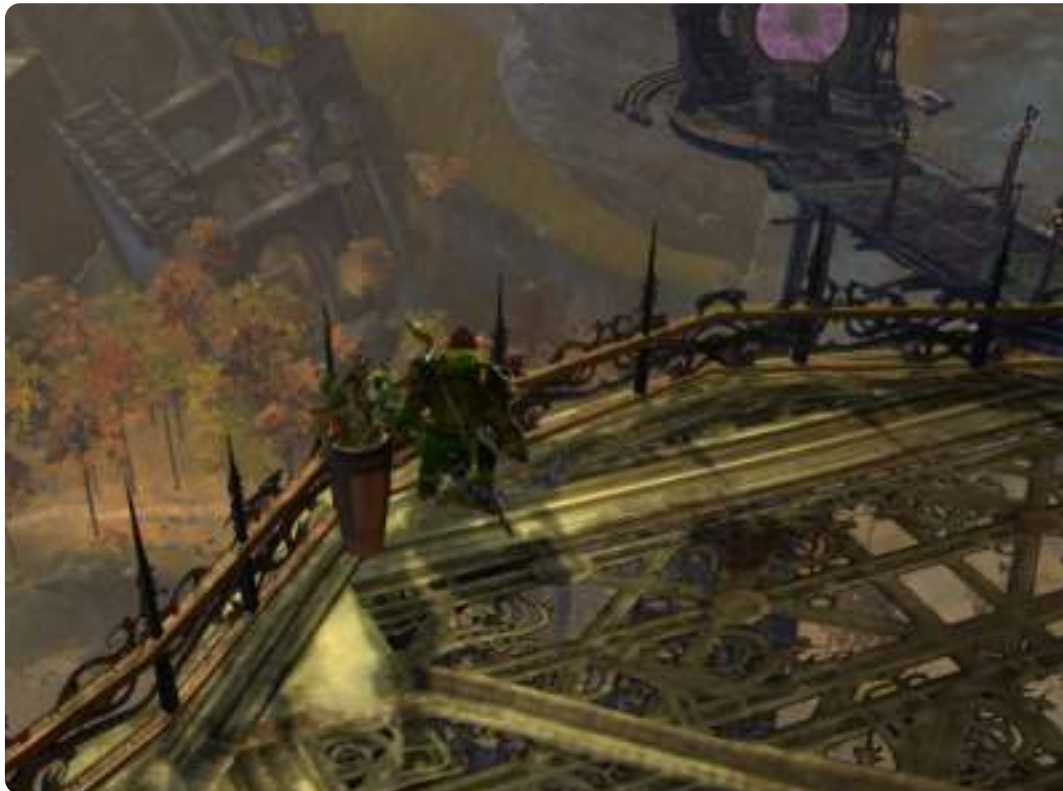
This worked for me during the beta weekends (since then, I've switched to PlayOnLinux in order to have easy access to the raw

input patches). It's also recommended that you have d3dx9, gecko, and mono installed. (Though both gecko and mono don't appear to have an impact, it's better to have them and not need them).

Once your launcher is created, you're all ready to start patching (trust me, it can take a while). Some people report intermittent crashes of the launcher when using Wine, but I didn't have this issue. If you want to risk leaving it unattended as it downloads, you

can give it a shot, but you may need to restart the launcher occasionally.

Hopefully I've appealed to any MMO fans with this article. If I have, and you'd like to see these sorts of articles continuing, please let me know (along with any names of games you'd like to see covered – if I can get it working, I'll cover it). In case you have any questions, comments, or suggestions about this article (or ideas for future articles), you can reach me at lswest34@gmail.com. If you do email me, please include FCM or C&C in the subject line, so I don't overlook the email. Also, if anyone is playing Guild Wars 2 and has any suggestions to make it run better (or want to share some tips/ask questions), you're also welcome to email me. My home world is Gandara, but, thanks to the guesting feature, odds are I should be able to help out just about anyone.



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



COMMAND & CONQUER

Written by Lucas Westermann

Wunderlist & AirDroid

This month, instead of my usual command-line ramblings, I've decided to share with you two very useful pieces of software I've started using: **Wunderlist**, which is cross-platform, and **AirDroid**, which is an Android app. For those of you wondering whether this article is worth reading, I leave that up to you. To save you reading the entire article though, I will explain that Wunderlist is a To-Do list application that offers syncing among most any commonly found OS, and AirDroid is a web-based interface for managing your android device. This is especially useful for tablets like the Asus Transformer (TF101), which doesn't play nicely with Linux due to it using MTP. If you have no need for either, feel free to move onto the next article.

In the recent few months, I've looked for a method of managing To-Do lists on the go. My usual system of Dropbox, a text file, and conky/geekTool (on OS X) is lightweight, but doesn't offer the ability of actual reminders

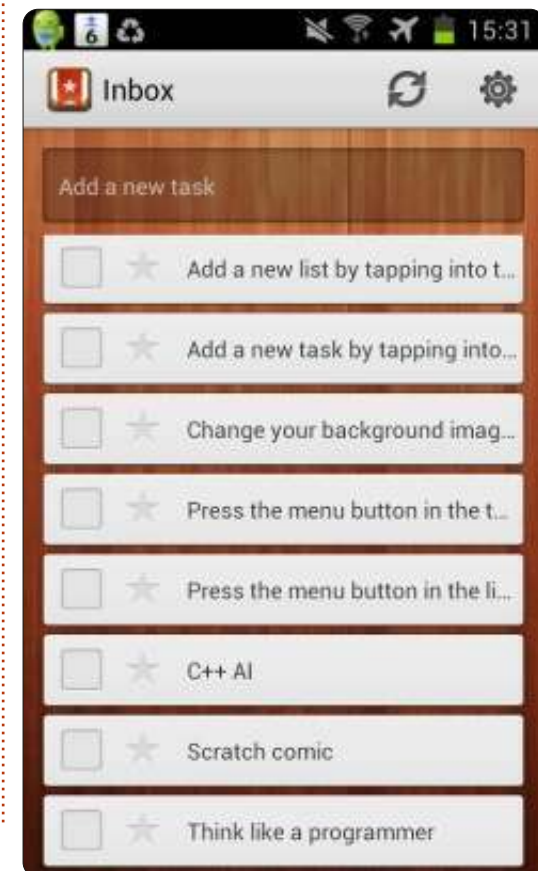
(without extra coding), and, unless I plan ahead and favourite/download the files on my phone, unavailable to me without Internet access. Beside that, not everyone wants to piece together a system like this.

Enter Wunderlist. It's a program available for everything from iOS and Android to Linux and Windows. It requires you to create a free account, after which you can sync your lists amongst any and all systems that you are logged into. It also offers a web-based login, so you can check it from computers outside of your usual circulation (library, public area, work, borrowed computer, etc). The astute amongst my readers may want to point out that Dropbox allows similar access to files, however, I've found formatting issues abundant when using Windows or OS X text editors, due to them using different formats for non-printing characters.

The main menu will drop you into an area called "Inbox"—this is where all of your tasks are listed.

Other options are: Today, Overdue, Starred, Done, Tomorrow, Next 7 Days, Later, No Due Date. You can configure notifications for your important tasks, and forgo it on any less important (or time-sensitive) tasks. The only feature I'd like to see implemented is the ability to see your tasks for today on your lock screen when it comes

to mobile devices, and, possibly, lightweight/minimized views, or widgets, for normal PCs. If, however, you have no issue unlocking your phone to check the widget, or opening the program on your PC/laptop (or leaving it open constantly), you'll find this to be a great little tool to better organize your (work) life.



COMMAND & CONQUER

An alternative to Wunderlist is, of course, Remember The Milk. The upside to Remember The Milk is the plethora of third-party, and command-line, software available for it. That being said, I do find the interface on Wunderlist to be a little nicer. If, after continued trial of both options, I make a distinct decision, I will make a note of it in Command & Conquer. If anyone has a preference (or another option I haven't named), feel free to email me about it—see the concluding paragraph for my email.

The other piece of software I want to tell you about today is AirDroid. It's a web-based interface for Android, which allows managing of all multimedia, files and apps on a device. For those of you having a mini heart-attack at the thought of all that information being available on a network—relax, at least a little. AirDroid requires a QR Code to be scanned, or the password from the device to be entered at the login screen. Now, this doesn't mean I would use the program in a publicly accessible network, but for home use, so long as your network is encrypted/password protected, it should be just as safe

as anything else you do online. Transfer rates, of course, depend upon the LAN you're using on either end of the line (sending/receiving), but will generally be slower than USB. That being said, I was able to put a 4 GB movie on my tablet via a regular ol' 802.11g network in about 5-15 minutes, depending on whether my source device was connected via ethernet or wirelessly.

If you're unable to connect your tablet to Linux, or the trade-off of security for ease (no cables) is worth it to you, I highly

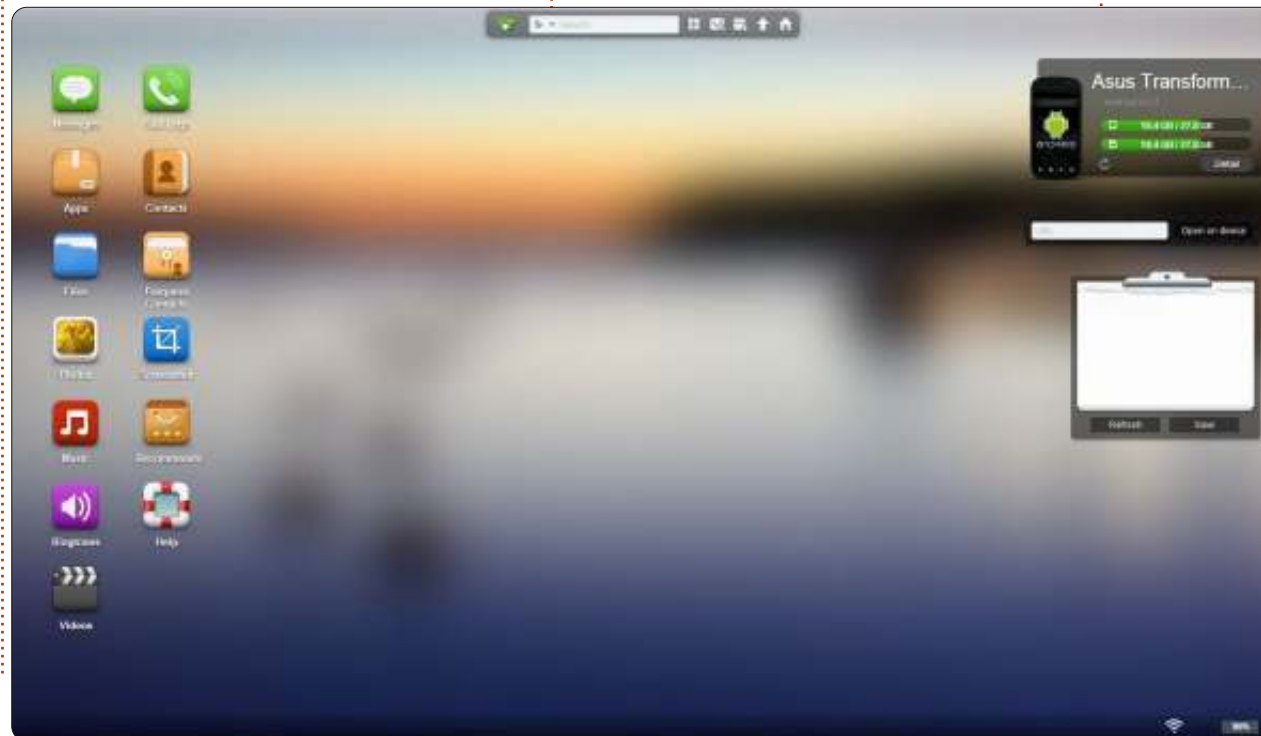
recommend giving AirDroid a shot. However, if you're a stickler for security (or have alternatives), I would suggest using something else. As always, if you want the best transfer speeds, using a wired option will serve you best. Lastly, I should note that quitting the app (and killing the server) is done by hitting the back button when at the main window. I, at least, needed a minute to figure that one out.

As always, I hope you've found my opinions useful—and that I've introduced you to some new (and

possibly desperately needed) alternatives/solutions. If you have any comments, questions, or suggestions, feel free to email me at lswest34@gmail.com. If you decide to do so, please include FCM or C&C in the subject line, so that I don't overlook it.



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.





This past month saw me installing Windows 8 on my dual monitor PC setup, in order to test it thoroughly for a client. Naturally, I didn't remove ArchLinux, but there are certain steps required to keep both operating systems happy. As such, I felt this month it may be useful to explain some of the extra steps, in order to save some people headaches when dealing with this.

Trick 1: Hard disk order

Note: This is important only to people with multiple physical hard disks in their PC.

Windows has a major complaint any time the primary hard disk is not the one Windows is installed in. It works just fine, so long as you don't do anything with the partition layout. In order to run an upgrade, for example, you need to have the main partition (Windows C: drive) set to active. This can be done in the disk management menu, but won't work if you've made the Windows drive

secondary. Solving this is as easy as switching the order in your BIOS.

Trick 2: GRUB is gone

As one would expect, Windows replaces any boot manager with its own. After this happens, you have two options: either create a menu for Linux from the Windows boot manager (this is possible), or re-install GRUB. If, like above, you have two disks, my preference is to simply install GRUB on the Linux drive, and leave the windows boot manager intact on the other. The boot manager used is always the one on the primary drive, so you can effectively switch between both managers by switching the hard disk order. This saves you the trouble. If, however, you don't have two drives, you'll need to re-install GRUB. The easiest way to do this is to boot from a live CD, and, for GRUB2 (which is used on all the newest Ubuntu versions), run:

```
sudo grub-install /dev/sdX
```

in the terminal. Remember to replace sdX with the actual hard

disk number (for example, /dev/sda). You can check this with either gparted, fdisk -l or any similar program. If you have internet in the live CD, you can also install a graphical tool called boot-repair.

Trick 3: System time is constantly wrong in both Windows and Linux

This is often the case when Ubuntu is set to use UTC time. Coordinated Universal Time (UTC) is a modern successor to GMT, and is the de facto standard for the Network Time Protocol and quite a bit of the internet. The problem? Windows refuses to offer an easy option to enable UTC. You can edit the registry in order to enable it (see UbuntuTime link in Further Reading below). The, probably easier, option is to set Linux to local time. To do so, simply do the following:

```
sudo vim /etc/default/rcS
```

Then edit, or create, the following line:

```
UTC=no
```

If you're someone who isn't comfortable editing files like this, or editing the registry, you can also simply adjust your timezone in Windows to balance the offset. For example, if you live in GMT+1, and your clock is 2 hours behind, switch to GMT+3, and it should work just fine. However, this may create unforeseen problems further down the line.

The "Developer-recommended" solution is to set both Windows and Linux to UTC, and disable Internet synchronization in Windows (it doesn't seem to work with UTC), instead relying upon NTP in Linux. I'm currently testing this solution, and it seems to be working alright so far.

Trick 4: Keeping media organized

If you want to keep your media

(music, videos, pictures, etc.) available to both systems without having to constantly copy and update the files, the easiest solution is to have a media partition. I've split my 1TB hard disk into 200GB for Linux, and 700GB for media. The Linux stuff is just a / and /home (both EXT4), and the 700GB partition is NTFS. The 700GB partition contains all my music, videos, pictures, and backups. Once you've relocated the files onto the media partition, you need to create symbolic links to them in Linux. My preference is to create a sub-folder link, as opposed to replacing the Music, Pictures and Videos folder completely. To create a symbolic link, do the following:

```
ln -s /media/Media/Music  
~/Music/External
```

This would create a link to your Music folder (on the partition called "Media"). The link would be called External, and is located in the Music folder within your home folder. You can, naturally, change the command however you see fit. You can do something similar in Windows 7 or 8 (possibly in Vista), by adding a new folder to your "Libraries" for Music, Pictures,

Videos. Instructions can be found in the Further Reading section.

A quick note: If Ubuntu doesn't automatically connect the Media partition, you'll need to do that manually, and instructions are below in the Further Reading section.

This trick for media files can also be used for any other type of file you would need to share. For example, you can stick your Dropbox folder on it, and avoid having to download any updates twice.

I hope at least a few of my readers find this article useful, and for those of you who didn't, odds are I'll have something to interest you next month. If you have any questions, comments, or suggestions, feel free to email me at lswest34@gmail.com. This month I've also written a review on Think Like A Programmer by V. Anton Spraul, so if you're interested in programming, you may want to give it a look.

Further Reading:

<http://windows.microsoft.com/is-IS/windows7/Customize-a-library> – Customize Windows Library

<https://help.ubuntu.com/community/UbuntuTime> – UbuntuTime

<https://help.ubuntu.com/community/RecoveringUbuntuAfterInstallingWindows> – Restore GRUB

<https://help.ubuntu.com/community/AutomaticallyMountPartitions> – Automatically Mount Partitions

<http://superuser.com/questions/482860/does-windows-8-support-utc-as-bios-time> - UTC in Windows



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.

Full Circle Podcast Episode 31, The Difficult Third Episode!!



It may be a new team of podcasters, but the format is still the same.



Your hosts:

- Les Pounder
- Tony Hughes
- Jon Chamberlain
- Oliver Clark

All members of the Blackpool (UK) LUG
<http://blackpool.lug.org.uk>

Olly & Tony bring everyone up to date with what's been going on in the "Summer Break", Olly talks about his Gnomebuntu install, Code-academy and Tony talks about his new Nexus 7.

Download



This month, I thought I'd do something slightly different from the usual tutorial-style articles. Specifically, I'm going to share my experiences with the Synology DS213+ Network Attached Storage (NAS) that I recently purchased and set up, to work with Linux/Windows/Mac OS X, though I'll only be focusing on the Linux side of things. As such, if you already have a NAS set up (or don't intend on getting one), this article probably won't be for you.

The Product

The Synology DS213+ is the "small and medium business" option of the 2-bay NAS systems offered by Synology. As such, it's more expensive than the DS213/DS213j, but offers a few more features. For an exact list of what isn't included in the cheaper versions, you can compare the products on the Synology web page. At first glance it seems that the DS213+ simply offers features like LDAP and Windows AD,

hardware encryption, and slightly better performance. The NAS does not come with hard drives, so you will need to factor that in cost-wise. The model discussed in this article is using two Seagate ST3000DM001 (3TB) drives.

The Setup

Installing the drives into the device is painless and easy (even if it does require screws and a screwdriver). However, make certain to seat both drives properly, as not doing so can result in a drive not being recognized by the system.

Once you've plugged it in and inserted the drives, you'll need to start up the device by pressing the power button. For the software setup, you have two options: using the Synology Assistant software (available for Linux, Mac, and Windows), or use the web interface at <http://diskstation:5000>. Regardless of your choice, you'll need to let the software complete the first-run setup. This includes creating an administrator account.

It's a relatively painless process, if somewhat slow. Once you can log in, you'll need to decide on the storage manager volume you'd like to use. You have the choice between RAID 1 and RAID 0 (or to let the DSM system decide on its own, which generally means RAID 0 for single drive setups, or RAID 1 for dual drive setups). For those who don't know:

RAID 0 uses "striping". To explain what striping is, simply imagine you have a set of values you want to write (A1-A8), where the system writes A1 and A2 at the same time, but to the separate disks. This results in a high level of performance, but offers absolutely no failure recovery. If you're using

the NAS for temporary storage or a high throughput system, you'll probably want RAID 0. With RAID 0 you'll also have access to, in our case, 6TB of storage.

RAID 1 uses mirroring without parity or striping, meaning that Disk 1 is an exact copy of Disk 2 – this means that if one drive fails, you can simply replace it and keep going. The downside is that you have access to only one drive's worth of storage (3TB in this case). Generally you'll want to use RAID 1 or higher for backup systems. However, note that RAID 0 and 1 are the only options for two-drive systems.

The system discussed here uses RAID 1, due to the fact that it's primarily used for backups.

Diskstation Manager (DSM) also allows you to create multiple volumes by selecting specific drives, though it seems largely useless in a two-drive setup. Once your volume has been created, you're left with the biggest question of all: how do you want to use the space?



The Usage

Due to the fact that the test network included Mac, Windows and Linux computers, both the Mac File Service and the Windows File Service were enabled. Linux, however, can treat the Windows File Service as a simple Samba share (accessed by visiting `smb://<IP>/<share>/` in Nautilus). DSM also offers an NFS (Network File System) service, although the functions and features weren't tested.

In order to allow each person of my family's network access to a personal (and private) share, I had to create a separate user for each, disallowing access rights for all other users (save the admin account, in the case that IT help was required). Taking it one step further, a TimeMachine account was also created, due to the fact that we had multiple MacBooks which could all benefit from TimeMachine. To avoid TimeMachine filling up the entire 3TB available, a data cap was set in place (it's suggested to use 2-3 times the size of the actual Mac drive, which was impossible for 4 MacBooks – the total would be

about 4TB!). As such, it was set at 1TB. For anyone using something similar (snapshot backups), it would be recommended to exclude folders that won't change much (Videos/Music, for example), and instead manage a single backup/copy of these folders, to avoid unnecessary duplication of efforts.

Overall, the DSM interface is well designed and extremely easy to use. However, the clock and date do not, apparently, automatically synchronize with the network. As such, you'll need to change that setting before expecting scheduled tasks to run at the proper time.

The Noise

For anyone considering a NAS, it's important to consider location. If you're going to be keeping it in a bedroom, you'll probably want to invest in server-specific hard drives (they're quieter, although performance may be impacted), or set up a scheduled downtime, allowing you to sleep without the fans and disks whirring away. Also, the disk standby can result in loud spin-ups/spin-downs (at least with

the Seagate drives, although according to reviews these disks are simply a bit louder). Thus, you may want to balance power-saving with peace of mind. Another alternative would be to simply soundproof the device – keep it in a padded enclosure, suspend it somehow, or set up some sort of sound barrier between the device and the rest of the room. These are minor details, and also largely dependant on the specific drives and usage of the device. The score result disregards these minor annoyances, as the NAS enclosure itself isn't actually causing them. Indeed, the enclosure itself sports a suspended drive bay, and rubberized feet, which should help to minimize noise as well.

The Score

Overall, the NAS succeeds in offering an easy-to-use experience, while offering a great many features (both advanced and basic). It could easily be used as a personal server, as it offers web hosting as well as FTP options. Through 3rd party software, you can also easily install PHPMyAdmin and other common tools. The only downside to this is the cost,

especially due to the lack of drives. This is mostly offset by the wide range of features it offers, in comparison to similar NAS setups. Due to the form factor and design, along with the features and performance, this device gets a 4.5/5. The half a mark is simply because, while an excellent device, it could offer certain features to make life easier, such as hot swapping drive bays.



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



COMMAND & CONQUER

Written by Lucas Westermann

Google Music

As many of you may already know, Google Music was recently rolled out to countries outside of the USA. As such, I finally had access to it in Germany, and promptly began uploading my music collection to the cloud. Due to the fact that I am not necessarily always on one of my own personal computers, and own multiple Android devices, having access to my music from anywhere was rather appealing. Nothing is worse for me than being stuck doing IT work at someone else's computer without music to listen to, especially if I can expect to be there for a long period of time. I've now been using Google Music for about 2 months, which means it's high time for an article!

For anyone who is unsure what Google Music is, it is a free service offered by Google. It essentially allows you to upload 20,000 songs to your personal cloud, and allows you to stream your music from any web browser, synchronize the cloud with personal computers using the Music Manager application, and allow Android

devices full access to your cloud, from which you can pick and choose specific songs to keep locally on the device.

THE SETUP

Setting up your cloud is fairly easy. You start up the Music Manager application, select what you want to synchronize (and whether it is automatically synchronized or not), and then begin the process. The application seems to upload a maximum of two or three songs simultaneously, and for my roughly 2,000 songs, it took between 4 and 6 hours to upload (at an average upload speed of 50KB/s). As such, this may not be suitable for anyone with a slow upload rate.

MANAGING

The Google Play store now also offers music downloads, with a similar pricing scheme to the iTunes store or Amazon. In Germany, it seems that the selection isn't quite as complete as it could be, especially for more obscure artists. However,

some MP3s are available for free, although there is no easy way to filter these out. Therefore, increasing the size of your music collection should be fairly easy!

Managing your playlists and song information is just as simple as with any other music program. The nice thing is that if you edit any playlists that you have set as "keep on device" on any Android systems, they will automatically download the new files, which is a nice wireless solution. My music collection is thoroughly organized with album art and completed song

information; thus I can't tell you if the cloud service automatically updates the information, or how easy it is to complete missing information. I would imagine it should be rather painless. Each song can be downloaded from the web page a maximum of 2 times, and an infinite number through the music manager, as far as I can tell. This is nice if you need quick access to an MP3 without going through any additional setup.

As far as I can tell, you need to configure your computer for uploads before being able to automatically synchronize new files. I haven't yet configured uploads on my laptop, making it difficult to tell for certain. I remember initiating the process and having the laptop begin uploading the MP3s as normal, although no duplicates appeared in my cloud database. This probably means that Google sorts out duplicates after they've been uploaded – so keep in mind that you are effectively uploading your library twice for no particular reason. I could be wrong, but it



certainly seems that way. If anyone has different experiences, let me know via email.

QUALITY

The uploaded files are converted to 320 kbps MP3 if they are in an unsupported format such as FLAC or OGG. Otherwise they seem to retain their file types (my MP3s are all highest-quality variable bit-rate for my mobile devices). Streaming on a mobile network results in a lower quality, in order to save your bandwidth/handle the lower speeds of a mobile network. When using a wireless network, the quality of audio is akin to playing the local file on any of my devices. However, initiating the queue can occasionally take a couple of seconds before it's buffered. Once playback has started, it seems there are no pauses between songs (with the exception of when there are massive drops in bandwidth/available speeds).

PLAYBACK

Playing back music directly from your cloud storage is possible only through a web browser on

non-mobile devices (laptops, PCs, etc). If you have a mobile device that runs Android, accessing and playing the music directly from the cloud is possible with the Google Music app. This does not appear to be the case with any non-Android operating systems. There is also no support for media keys, or playback information in Conky. There is, however, an extension for Chrome called Music Plus (developed by the editor-in-chief of Lifehacker), that offers further features, such as a popup controller, last.fm scrobbling, html5 notifications, etc. For a link, check the further reading section.

Google Music does offer the ability to create playlists based on a song, including both music you own and music that is available on the music store. Not all MP3s seem to stream in full length when doing this, but it's an excellent option for finding new music.

CONCLUSION

This is a very useful free service to use if you listen to your music often on mobile devices (such as a phone or tablet), where you may not want to have local copies of

every song taking up space. It's also a wonderful solution for anyone who has issues syncing music with an Android device, as it effectively avoids the issue of compatibility. If you own a laptop with limited space, you may also very much appreciate this system, although there are a few restrictions when using Google Music purely in the cloud. Lastly, if you have massive amounts of music, or a slow internet connection, this is probably not for you. It could definitely fulfill the role of a cloud backup for music purchases, or for backing up certain favorite songs, as you have a lot of control over what files are ultimately uploaded.

Due to the playback restrictions, I don't see Google Music replacing a local music player any time soon, but it definitely makes life a bit easier when organizing your music across multiple devices. If you're someone like me who is constantly on the hunt for more music to listen to, you may also like some of the more advanced features offered by Google.

FURTHER READING

<https://chrome.google.com/webstore/detail/ipfnecmlncaiipncipkqijboddcdmego> – Music Plus

<http://music.google.com> – Google Music website



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



COMMAND & CONQUER

Written by Lucas Westermann

Suggestions

I haven't had many issues to fix this month, nor have I had any emails on topics I should cover. As such, I figured it could be time for another series of useful suggestions. This time around, I'll cover: recording electric guitar easily, using an android phone as a barcode scanner, transferring pictures from your android phone to your PC wirelessly, and a small tip for cable management.

RECORD ELECTRIC GUITAR

As many of you have probably heard, Ubisoft recently released a game called *Rocksmith*, which features a 1/4" jack to USB cable. Surprisingly enough, the cable seems to work without drivers on any OS (I've tested this on Mac OS X, Windows 8, and ArchLinux). So, for anyone who owns Rocksmith and an electric guitar – all you need to do is install Audacity and record! Due to the fact that the cable contains a normal 1/4" jack, you can run your signal through the usual chain (pedals, preamps, etc) before heading into your PC.



It's also possible to buy just the Real Tone Cable, though, for the price, you could probably find other solutions. If anyone is interested in a review of Rocksmith (for Xbox), send me an email and I'll get right on it.

Note: The quality of the recording is decent, but I found that it is sometimes too quiet. However, it's easily fixed with a bit of post-processing. Logically, the quality will also be dependent on the pickups in your guitar. All testing was done with my PRS SE Custom 24.

Sound samples (I apologize for the quality of the playing...):

<https://soundcloud.com/lswest/chords-pure> – Chords, played directly from the guitar into the PC
<https://soundcloud.com/lswest/santana-g3> – A little bit of a Santana tune, played through my Zoom G3 multi-effects pedal and then into the PC.

Comparing the quality of the G3 (which can act as a USB interface) and the Rocksmith Real Tone Cable, I'd have to say I prefer the cable – it sounds just as good (if sometimes a bit quieter), and it's a lot less of a hassle to set up.

BARCODE SCANNING

Anyone with an android phone will probably know that there are a

multitude of barcode/QR code scanners available for Android. However, there is one app that seems especially useful – WiFi Barcode Scanner. This app uses VNC to enable wireless scanning of barcodes to your PC from your phone. For €0.99 in the German store, it's definitely an alternative to look at when contemplating some sort of wireless scanner. Why would you want a barcode scanner? If you're someone who needs to manage a large collection of books, magazines, etc, or want to manage any collection with custom barcodes/QR codes, then you may appreciate the ability to scan barcodes in order to input them into spreadsheets or databases. This is definitely something I'd recommend to anyone thinking of buying a barcode scanner, as it's by



fullcirclemagazine.org

far the cheapest solution I've seen, so long as you already own a camera-equipped Android device. Due to the fact that Google Play offers full refunds (within 15 minutes of downloading an app), it can be a fairly risk-free attempt.

TRANSFER PHOTOS WIRELESSLY

Google has very recently enabled full-size photo uploads to Google+ via instant upload on the mobile app. Originally, it didn't seem like such a useful option to have. However, while writing the HTC One S review (see later in this issue), I needed to take a few photos to demonstrate the camera. Once taken, I would have had to hunt through my room for the micro USB cable in order to transfer them (or else install/use something like Dropbox or AirPlay). Instead, within a few seconds (time will vary depending on your upload speed), they were present and available on my Google+ private album – from which I simply downloaded the photos. So, for anyone who likes quick and easy solutions like this, and if you have a Google+ account, this may be worth noting.



CABLE MANAGEMENT

For the past few years, I've had a constant humming from my (admittedly older) Logitech speakers. Originally, it wasn't a big deal – the humming was quiet enough to be ignored. However, I recently had to replace our router (which sits on my desk, along with all the other electronic stuff I own), and after which my speakers were

humming like crazy, even on the lowest volume setting. I chalked this up to signal interference from power cables, and so I spent a day labelling every cable I had, and then grouping power cables together, ethernet cables together, USB, and audio cables. Once I grouped them, I created a separate “pathway” for each group to run through under my desk. Ethernet cables were wrapped around the desk's supports, power cables ran down the left side of my desk and then along the floor, and the USB and audio cables ran down the right side, spaced a bit apart. The result? No hum whatsoever. I've also noticed fewer disconnection issues via ethernet, meaning I probably had some interference there too. For anyone experiencing similar issues, sorting through your cables may be a very good idea.

I hope at least a few people have found this article useful, and, if you have any questions, comments, or suggestions, you can email me at lswest34@gmail.com. If you do email me, please put “C&C” or “FCM” in the subject line, so that it doesn't get lost in my inbox. I'd be extremely interested to know if any readers find the

occasional music-oriented articles interesting – if they do, I would be happy to do an article on using Audacity to record guitar (electric, acoustic, whatever). If anyone wishes to see some of these articles, just send me an email letting me know.



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



I originally wanted to write this article on my first impressions for the developer preview of Ubuntu Touch. However, as I couldn't get the work-in-progress version working on my TF101 (Asus Transformer), and as I wasn't willing to sacrifice my brand new Nexus 7's warranty (and data) to test a very basic version of Ubuntu Touch, I had to head back to the drawing board. When sharing my situation with Ronnie, we ended up having a discussion on whether or not Ubuntu Touch would be replacing Android for us. Ronnie was skeptical about it, while I was fairly open to the idea. Long story short, I decided to write up my opinions on why Ubuntu Touch could be interesting to mobile users, and why it might not manage to lure you away from Android. I will be ignoring any other mobile operating systems, as Ubuntu Touch can, at least for the moment, be installed on only Android devices. For those of you who were expecting my typical how-to articles: they should be back as of the next issue!

WHAT IS UBUNTU TOUCH?

For those of you who missed the announcements, Ubuntu Touch is a version of Ubuntu being developed for android-based devices with screen sizes between 4" and 10" screens. The Further Reading section has links to the official announcement pages. A few interesting features that the end product is aiming to have are as follows:

- The ability to dock to a keyboard, mouse and screen for a full Ubuntu desktop experience (similar to Ubuntu for Android).
- The ability to seamlessly move work between devices (phone, tablet, TV, desktop).
- Utilize the entire screen space (accomplished by utilizing swiping in from edges, and hiding the

interface buttons and options until required).

- A lock screen that is "tailored to you". Essentially, the lock screen should display information relevant to you based on your frequent tasks.
- Minimize steps in order to get to the apps you want. For example, the app tray is revealed by swiping in from the left side, regardless of where you are (even on the lock screen), allowing you to jump straight into the application, instead of having to first unlock and then locate the launcher for the app.

- Ability to run phone apps (such as texting/chatting) alongside tablet apps (media applications, video chat, etc.). This is available only for the larger screen sizes, but appeals greatly to me, who is an avid tiling window



manager user.

There are, of course, many other features too. However, these are the ones that had me nodding along in agreement when they were presented.

WHAT'S KEEPING YOU ON ANDROID?

I used an Asus Transformer TF101 for a good couple of years, and only recently replaced it with a Google Nexus 7. While I mainly used either device for productive tasks (checking email, testing websites, reading articles or books, and, occasionally, for writing up notes/articles), I did occasionally enjoy a game or two. Productivity tasks should be as easily accomplished on Ubuntu Touch as on any Ubuntu desktop, or any Android device. The "problem" lies with those apps that occupy roles we wouldn't usually associate with desktops. This includes touchscreen-based games (for example, the Simpsons Tapped Out, Draw Something, or anything

of the sort), and any productivity apps that were developed with a touchscreen device in mind. Due to the fact that Ubuntu Touch is aiming to offer desktop applications which can be found in Ubuntu on a mobile device, many applications geared towards the mobile will probably be more error-prone when the first devices are released. This is simply due to the fact that Ubuntu Touch is new, while Android has been around for a number of years now. As such, some apps are more matured, and could very well cause some adopters to hesitate before looking for alternatives in Ubuntu. Canonical is trying to improve this situation by integrating small-screen support into their API, in the hopes that developers will start adding support for Ubuntu Touch long before the stable release. Ultimately, you can't do anything besides waiting for the release before deciding whether or not it's worth it.

Do you have any Android apps you don't think you could live without? Let me know in an email (address in the last paragraph of this article).

DOESN'T UBUNTU TOUCH USE ANDROID?

Ubuntu Touch is based off CyanogenMod 10.1 (jelly bean). However, besides the device support, and the system with which Ubuntu is flashed/booted, there isn't a lot left of the ROM. Ubuntu itself is being run from a chroot environment the moment all device drivers are loaded from the Android framework. The reasoning for this, I imagine, is that it enables wider support for devices, and avoids the need to write new drivers for every piece of hardware. I don't see a lot of potential downsides with this method.

DOES UBUNTU TOUCH HAVE A FUTURE?

I think it most definitely will continue to be developed. If the speed with which the developer preview was ported to different devices is any indicator, it may grow very, very quickly. I doubt it will replace Android, just as Web OS, Windows Phone and iOS failed to eliminate other mobile OSes. It will probably carve out a market

share for itself, and be especially interesting to businesses. The reason for this is the fact that, once enough phones meet the requirements, Ubuntu Touch can easily be used as a desktop (with the correct docking tools). It will probably appeal to those companies who are currently assigning each employee a work phone, a work laptop, and/or a work tablet. With Ubuntu Touch, you could issue a phone and a dock, and simply fill your offices with docks, keyboards, mice and monitors. How feasible this is depends entirely on the actual implementation of this desktop feature in the final product.

CONCLUSION

There is little anyone can say for certain. However, knowing FOSS development, we'll be sure to see rapid progress as we approach the estimated late 2013/early 2014 release date for Ubuntu Touch devices. The basic premise, and the features Canonical is trying to integrate, are both things I am extremely interested in. If they can pull it off, I will definitely be giving the Ubuntu devices a fair chance. The moment it becomes more

functional (and device support is a little better), I'll be sure to give installing it on my TF101 another shot.

As always, I hope at least a few readers have found this article interesting. If you have an opinion on Ubuntu Touch (or simply agree/disagree with my opinions), feel free to send me an email at lswest34@gmail.com. Please put "C&C" or "FCM" in the subject, so that it doesn't get lost in my inbox.

Further Reading

<http://www.ubuntu.com/devices/tablet> (Tablet page)

<http://www.ubuntu.com/devices/phone> (Phone page)

<https://wiki.ubuntu.com/Touch/Devices> (List of devices and their current port progress)



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



After my article last month on Ubuntu Touch, I received a relative flurry of responses from readers (more so than for any other article I've ever written). Due to the response, I thought it might be nice to share readers' responses in a follow-up article. The first weekend of April also saw a big change in the TF101 Ubuntu Touch packages, and I was finally able to get the developer preview running on my tablet. However, as hardly any functionality exists at this time, I will save a full-on review for a later date (an official Beta release, for example).

The first response I received was from a reader who shared his opinion that Ubuntu Touch absolutely must have integrated support for Google accounts (Google Mail, Google Contacts, Google Calendar (with emphasis on the reminders), Google Maps and Navigation). Upon reading the article, I realised I had neglected to mention anything of the sort. That being said, I imagine Ubuntu will ship with some form of mail client (Evolution/Thunderbird). Google

Mail support was very effective in either of those applications, the last time I used them. Thunderbird also offers the Lightning plug-in for calendars, which works well with Google Calendar. There are also numerous plug-ins for Thunderbird that allow the syncing of contacts. As such, I think it highly unlikely that Ubuntu Touch releases will be lacking a tool with similar features. That leaves Google Maps and Navigation. My fear is that Canonical will decide to include some new map/navigation tool, instead of trying to include Google's tried-and-tested applications. There is always the argument "Google Maps can easily be accessed through a browser", but it defeats the purpose of a quick and handy maps tool. At this point, all we can do is wait and see.

This same reader also stated that he wished to see a highly functional/usable keyboard in Ubuntu. While I haven't had issues with the default Android keyboards, there seems to be a large number of Android users who prefer keyboards such as Swype. In

the reader's case, he mentioned TouchPal's Keyboard, Dialer and Contacts app. This does more than just replace the keyboard, but also offers a custom dialer and contacts app. His reasoning for why this app is worthwhile was because he found typing on the typical Android keyboard to be "slow, boring and error-prone". He found the "swipe to type" method of TouchPal's keyboard much more intuitive, and, due to the learning algorithm, less error-prone. The reader also notes that while TouchPal's keyboard app offers predictive features, they are turned off by default, and it's not necessarily clear how to enable them. According to a Sprint user guide, sliding right on the space bar enables predictive typing, and sliding to the left disables it again.

The Dialer and Contacts apps included in the TouchPal package are useful for the reader, because the default Contacts app tends to lose data included in the Notes sections of contacts when they contain a lot of text. As the reader creates these Contacts on the PC,

it's impractical to deal with, and TouchPal's Contacts and Dialer app (one shortcut for both tools) apparently does not suffer from this limitation, and the reader once again finds it more flexible and easier to use.

The second email I received was short and to the point. The reader felt the possible lack of Aldiko (an e-reader application), and Kindle apps on Ubuntu Touch would be something to consider. Now, I'm not sure Aldiko will be ported/available for Ubuntu Touch, but I think Canonical would be remiss if they didn't include some form of Kindle app. Once again, Amazon does allow reading of Kindle books online ("Amazon Cloud Reader"), but it's hardly an ideal solution. That being said, the list of dummy applications in the Developer Preview does include an Amazon store shortcut, which makes me believe Ubuntu will do its best to integrate Amazon services. I would supply further information about the apps included, but, as the keyboard isn't working properly, I can't connect to

the wireless, making nearly every app throw “not connected” errors, or disabling those features I'd be interested in testing.

Lastly, I had a reader explain to me that he wanted to install Ubuntu Touch on a Samsung Galaxy S III, but cited the Ubuntu Wiki page where it was listed as a “work in progress”. So, for anyone else who was wondering something similar, I say this: if you're willing to give up a perfectly functional device in order to install a pretty much non-functional Developer Preview just to see what it looks like, you can generally expect any of the Wiki pages for your device to include decent instructions. That being said, Ubuntu Touch is not in an alpha, beta, or release candidate stage. It is quite literally a snapshot of their demo device, including multiple accounts you can't access, a guest account you need to use, and examples of applications they want to include. If, however, you have a second or third Android device just sitting around (such as my TF101), then feel free to flash it and play around a little, just don't expect to be breathing new life into an old device (just yet). Lastly, the Wiki

page is split up in this way:

- Working with phablet-flash (official Ubuntu method, contains just the Galaxy Nexus, and the Nexus 4, 7 and 10);
- Working, but not available from cdimage.u.c (unofficial ports, maintained by the community, but fairly functional); and
- Work-In-Progress (basically the waiting point for any port that hasn't yet gotten unlocking/installing instructions, working code or an image).

Unless you want to help debug and develop, Work-In-Progress

devices are probably not worth flashing.

On a completely unrelated note. After my article on the Rocksmith Real Tone Cable for recording electric guitar, I had a reader request an article or two on using Audacity to record acoustic guitar. These articles will happen as soon as I have a working audio interface available to me, and I will be covering recording acoustic guitar with a microphone, with pickups, electric guitar (direct-in), and with a microphone in front of the amp. If all goes well, this will happen within the next two issues

of FCM. If any musicians out there have suggestions on audio interfaces they have had working with Ubuntu, or if anyone has specific questions they'd like me to cover in the articles, feel free to email me at lswest34@gmail.com. Please include the words “FCM Guitar Article” in the subject line, so I can separate them from usual C&C emails.

I hope at least a few people have found the reader responses useful and/or interesting! I had definitely never heard of the TouchPal Keyboard, Dialer and Contacts app. Got any questions? Suggestions? Then feel free to email me at lswest34@gmail.com. Please include either “FCM” or “C&C” in the subject line, so it doesn't get lost in my inbox!



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



As many of you have probably heard at one point or another, an accepted practice for keeping your skills as a programmer sharp, is to learn a new language once a year. I do my utmost to stick to exactly that, and as such, I'm constantly on the lookout for ways to make it easier and less time-consuming. This month, I'd like to introduce you to two websites I've found, and give some general tips on how to go about learning new programming languages (whether you have any prior knowledge or not).

CODECADEMY

By far the most user-friendly option I've found is called Codecademy, and it offers an interactive editor/interpreter method for learning Web Fundamentals, jQuery, JavaScript, Projects (Web), Python, Ruby, PHP and APIs. Each of the listed topics has an entire track that is generally laid out nicely, and is focused towards teaching you very specific skills. Each of these offers a very

good basis for working with any further language, and as such, I can't recommend it highly enough. There are also a number of User-created projects, outside of the typical tracks, that you can jump into, in case you're the sort of programmer who has trouble deciding what you want to focus on.

<http://www.codecademy.com>

Further features:

- A point-based (and percentage-based) tracking system for progress between the tracks, to entice you to stick with it.
- The ability for any user to create lessons (teaching is a great way to cement your understanding of any topic).
- Quick, interactive online tools for programming. This avoids the hassle of setting up a programming environment, especially for those who have multiple devices.
- A useful hint system that supplies just enough information to help you when stuck.
- It presents the full, working code

at the beginning of the project, to help you understand other people's code without comments or compiling it yourself.

Who would I recommend this to? I would recommend this to anyone who wants to learn a programming language. If you have no prior experience, it should offer you all the information you need, so long as you are willing to thoroughly read everything written. For those who already have one (or more) programming languages under their belt, and want to simply become more comfortable with another, Codecademy is a great choice. It combines explanations and exercises, allowing you to fly through the tracks quickly. For example, it took me about 3 hours to get through the Ruby track, which consisted of 10 chapters, and about 238 exercises. They aren't the most challenging exercises for those with experience, but they're still enough to give you a grasp of the syntax and semantics of a given language.

KHAN ACADEMY

Khan Academy offers a fairly large selection of free online courses, covering a multitude of topics. This includes, but isn't limited to, Computer Science, Math, and Science. It also offers a few test preparation options for those studying for exams. If you're looking to learn more about programming (or other topics) in a typical classroom-type method, Khan Academy is for you. I haven't personally done more than one or two courses, but it's mainly due to the fact that I have covered these topics before – both in high school and university. From what I've seen, it's well done, and should serve as a useful service for anyone interested in that sort of learning.

<https://www.khanacademy.org/>

Who would I recommend this to? Anyone who learns well in a university-like environment with like-minded individuals and basic exercises and some lecture-type videos. It lends itself most to those

people who haven't studied courses like this before in high school or university, or who want a refresher for those classes. There isn't, admittedly, a lot of content yet for Computer Science, but it appears to be growing steadily.

PROJECT EULER

Project Euler is a website that has simply one goal: to supply you with mathematical problems. These problems range from the mundane, to mind-crushing feats of calculation. The kicker? Each problem is designed so that manually calculating the solution is so time-consuming as to be impossible. As such, you're required to program an algorithm (in your language of choice) to calculate the answer. Many questions require you to find and implement the correct algorithm in order to reach a solution in a reasonable amount of time (under a minute). Their system allows access to the discussion thread of that problem only when you've solved it, meaning you can happily discuss and share your algorithms with others in that area. If you're not a fan of mathematics, or don't know any programming languages,

or lack a firm grounding in mathematics as applied to programming (and aren't willing to pick up any of these skills), then Project Euler isn't for you. For anyone to whom this sounds like fun, I wish you luck with some of the more complex topics.

<http://projecteuler.net/>

Who would I recommend this to?

Fans of programming, mathematics, and/or problem solving. If you're willing to learn, you can manage fairly well, but a certain degree of background knowledge is helpful. I will say this: The problems do not seem to be sorted by difficulty. So if you get stuck, try the next one – it might be easier!

GENERAL TIPS

For anyone looking for my advice on learning to program, or to keep your skills sharp, I have only one main piece of advice: Program! The more you use a language, the firmer a grasp you will have on the theory, and the theory is often the most difficult to understand. Once you understand the theory, you can

generally apply it to any language, and learning the syntax and semantics of a language isn't usually very difficult.

If you're not sure what to program, Project Euler can help. Or else writing small scripts for use in Conky can also be a great way to get into a habit of writing script. You do often learn the most from larger projects, but keeping up with the basics is useful until that flash of inspiration that leads to a large project comes along.

What language should I learn? It depends entirely on what your goal is. If you're trying to find a language that will help to automate your Linux environment (or else hoping for something to integrate into Conky), I could recommend Bash, Lua, or Python. If you're into web programming, Ruby on Rails, PHP or Python could also be useful. If you're hoping to jump into an open source project in order to help out, most are written in C or C++, though it depends on what project in particular you have your eye on. Whatever you learn, keep in mind that the theory behind the methods are mostly universal, so pick whatever feels most comfortable for you!

I'm stuck on an error, and can't find a solution on Google! A lot of programming languages have IRC channels dedicated to them. In such rooms there are generally a few people who are happy to try to help you solve your issues. Sometimes you might even bump into the developer of the language! The best way to get an answer is to a) pastebin your code and b) be concise!

I hope I've managed to give anyone who is interested in programming a starting point. There are probably many other relevant websites out there. If anyone has a personal favourite, feel free to email me at lswest34@gmail.com with the website, or any questions, comments or suggestions you have for C&C in general. If you do email me, please include "FCM" or "C&C" in the subject line!



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



When going through last month's survey results, I was pleasantly surprised to see a number of interesting suggestions, and what seemed to be an overall positive response to C&C. However, I also noticed a few suggestions and requests for topics I have already covered. As such, I decided I'd devote this article to directing those to the past issues, and to also answer some of the quicker questions that were aimed in my direction.

LaTeX: I covered this in C&C in FCM#50 and #52 – where 52 was aimed more at Asian languages in LaTeX.

Virtualbox: Someone asked for advice on how to install Virtualbox in Ubuntu 12.04 because they were having issues. I had a short-lived virtualization series in FCM#38-45, but the short answer is, go to this web page: <https://www.virtualbox.org/wiki/Downloads>

Install the current version of VirtualBox for Linux hosts, then

the Extension Pack. You have to select the VirtualBox package which corresponds to the version of Ubuntu you are using, and 32-bit or 64-bit.

Automation: A surprising number of you seemed to be interested in writing bash scripts to automate things (either for yourself or for family members). I introduced that topic in C&C in FCM#24, but judging by the response, I'll probably be writing another few articles on that at a later date.

Conky: There were a fair amount of requests on getting started with conky. I have covered Conky in the following issues of FCM (in the C&C column): #44, #45, #46, #48, #51. If you have any particular issues with Conky, feel free to email me about it at lswest34@gmail.com. Please put "conky question" in the subject, and include the output of both `lsb_release -a` and `conky -v`

IDEs: Someone asked for a review of IDEs – without knowing

exactly what languages you want to develop for, it would be a fairly confusing article. That being said, I did discuss IDEs in FCM#64.

CLI Coloring: A couple of people asked about styling your prompt and terminal. I have discussed this in FCM#27, #35, #36

Command-line "cookbook": Someone asked for a collection of useful command combinations and I thought it might be interesting to open this up to the readers. If you have a favourite command (or ones you simply find yourself using a lot), feel free to add them to this google doc here: goo.gl/fp09r – please include why the command is useful (i.e. what you use it for). Once we have a list of sufficient length, I'll go through the commands and compile it into an article "cookbook".

And now, on to some questions that were fairly brief and would not have been enough for a whole article in and of themselves.

Create a folder hierarchy with a

one-liner (Bash scripts): First off, this doesn't require a script of any sort. Mkdir (make directory) is the command-line tool Linux offers for this. Usually it will create only a single folder at a time; if you tell it to create a path, it will fail. However, it has a command-line argument "-p" (how I remember this is p=path, so make path, instead of directory), which does exactly what you want. Take this scenario, for example: You want to create a folder Summer in your Pictures folder, with two folders within it called Tahiti and Montreal. This can be done with one single command:

```
mkdir -p
Pictures/Summer/{Tahiti,Montr
eal}
```

The -p argument will literally create directories as needed to reach the directories you want to create. So if Pictures or Summer (or both) are missing, they will be created. The only caveat is that you can't put spaces between the commas. You can certainly make names with spaces though, as you can see below:

```
mkdir -p
Pictures/Summer/{“Tahiti
2013”, “Montreal 2012”}
```

As you can see, you can't place a space between the items and the comma (mkdir understands it as two paths then: Pictures/Summer/{Tahiti 2013 and ./Montreal 2012} – as you can tell, they won't even end up in the same folder). Naturally, this also works on Mac OS X systems, as they offer the typical mkdir command.

Another question I had was to create a bash script to automatically mount a second and third hard drive. Once again, this isn't a Bash problem. Linux offers you a configuration file called /etc/fstab. This is in charge of mounting all your drives on boot-up. It will look something like that shown below.

The following mounts an NTFS drive to /media/Windows while giving permission to access it to

the user lswest and all users in the group users. Edit the uid and gid as you see fit. You'll also need to change /dev/sda5 to the correct partition:

```
/dev/sda5
/media/Windows ntfs-3g
uid=lswest,gid=users,dmask=02
2,fmask=133
0 0
```

Example of mounting a FAT32 USB stick using UUID – again, change the UUID to the correct one:

```
UUID=47FA-4071
/home/lswest/USB vfat
defaults,noatime 0 0
```

If you want to use spaces in the paths, you need to replace the spaces with \040 in the file. Once you edit /etc/fstab, you can test to see if it's working by running:

```
sudo mount -a
```

If the drives show up where you expect, and you can access them, you're all set. You can check UUIDs by running:

| # | <file system> | <dir> | <type> | <options> | <dump> | <pass> |
|---|---------------|-------|--------|------------------|--------|--------|
| | /dev/sda1 | / | ext4 | defaults,noatime | 0 | 1 |
| | /dev/sda2 | none | swap | defaults | 0 | 0 |
| | /dev/sda3 | /home | ext4 | defaults,noatime | 0 | 2 |

```
sudo blkid
```

Or, if you want the UUID for a single drive/partition, you can use:

```
sudo vol_id -uid /dev/sda2
```

Replace, of course, sda2 with the actual drive you want.

Lastly, there were a few requests for covering GRUB2. While that would be an article in itself, I felt I should at least offer a little bit of information for those who don't want to wait. For GUI-based editing of some settings, there's grub-customizer: <https://launchpad.net/grub-customizer>

If you're just after a different theme, they are discussed at the following links:

<https://help.ubuntu.com/community/Grub2>

<http://askubuntu.com/questions/66183/how-can-i-get-some-nice-eye-candy-themes-for-grub>

<https://help.ubuntu.com/community/Grub2/Displays>

Hopefully I've answered the questions sufficiently. If anyone has any further questions, feel free to email me at lswest34@gmail.com. Please include “C&C” or “FCM” in the subject line. If anyone has any suggestions for topics I should cover, you're also more than welcome to email me about them. Judging by the survey, it seems a great deal of you have suggestions or questions. The problem with some is that they are too specific for an actual article – but if you email me I can probably set you on the right path to fixing it.



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.



Last month, I started working my way through survey suggestions. However, to start with, I felt it prudent to move through the most common questions in a short-answer format. Starting this month, I will be devoting these articles to one or two topics from the survey results, beginning with setting default applications for programs, creating new MIME types, and assigning icons to said MIME types. For those of you wondering what MIME types are, they are the name given to the system for managing file types by extensions. For example, the MIME type for a text document (*.txt) is text/plain. All MIME types consist of a type and a subtype (in the format type/subtype).

DEFAULT PROGRAMS

The first spot to check for changing a default application is in System Settings -> Details -> Default Applications. Here you can set Web, Mail, Calendar, Music, Video and Photos application

defaults.

The second spot to check is the properties list of a file. You can right-click a file whose default you wish to change, and then head to the tab Properties/Open With. If you see the application you wish to use in the list, select it and hit the button "Set as Default". I will note – this is how it appears in Nautilus, and I'm not sure if the wording is the same in KDE/XFCE. If you can't find this option, or the application you want to use isn't listed, move on to the next step.

The last method I would suggest using is xdg-mime. It's a command-line tool to manage the XDG settings (xdg is the framework for dynamically choosing a program to load a file type – just try it by running xdg-open file). You could also edit .local/share/applications/default.list or /usr/share/applications/default.list manually – but I would recommend using xdg to make these changes, as it avoids duplicates and formatting issues.

The way you would go about checking a file's default application is as follows:

```
xdg-mime query default  
application/pdf
```

evince.desktop is the output telling you the default application. In this case, Evince

```
xdg-mime default kde-  
kpdf.desktop application/pdf
```

changes the default to KPDF.

```
xdg-mime query default  
application/pdf
```

kde-kpdf.desktop is the output in this case.

After this, running xdg-open on a pdf file should open it in KPDF instead of Evince. You must, of course, have installed the application first. If you're unsure what the desktop file is, you can run the following commands

```
sudo updatedb
```

updates the database for locate

```
locate *.desktop|grep  
application | more
```

will locate all the desktop files on your system, and search through them for your supplied application

If you're unsure what the MIME type is, you can run the following command:

```
xdg-mime query filetype file
```

Replace file with your actual file (including the path), and it will spit out the information you need for creating rules for that type. Keep in mind that for the web browser, you'll need to set the default for both x-scheme-handler/http and x-scheme-handler/https.

MIME TYPES

Very, very occasionally you may stumble upon a file type that doesn't have a MIME type. If this is the case, you can make certain by running the following:

```
grep 'extension'  
/etc/mime.types
```


COMMAND & CONQUER

If you don't get a result here, the extension (for example, txt for text files, py for python, etc.) isn't assigned to a type. To remedy that, you can edit the file using:

```
gksudo gedit /etc/mime.types
```

Once you've opened the file, add a new line at the end that reads:

```
text/extension extension
```

Replace the word "extension" with the extension of the file, minus the period. As I said earlier, this shouldn't be a regular occurrence – if it is, something is most likely wrong with your system.

ADD AN ICON TO A MIME TYPE

If you want to assign a different icon to a MIME type (adding a different symbol for python files, for example), you can do the following:

- Find a suitable svg file (or make one)
- Name it text-extension.svg

(replace extension with the corresponding text from the mime.types file). Logically, of course, you can't use slashes in the file name (otherwise Linux will assume it's a subdirectory), hence the hyphen instead. It must also all be lowercase.

- Copy the file to /usr/share/icons/gnome/scalable/mimetypes with

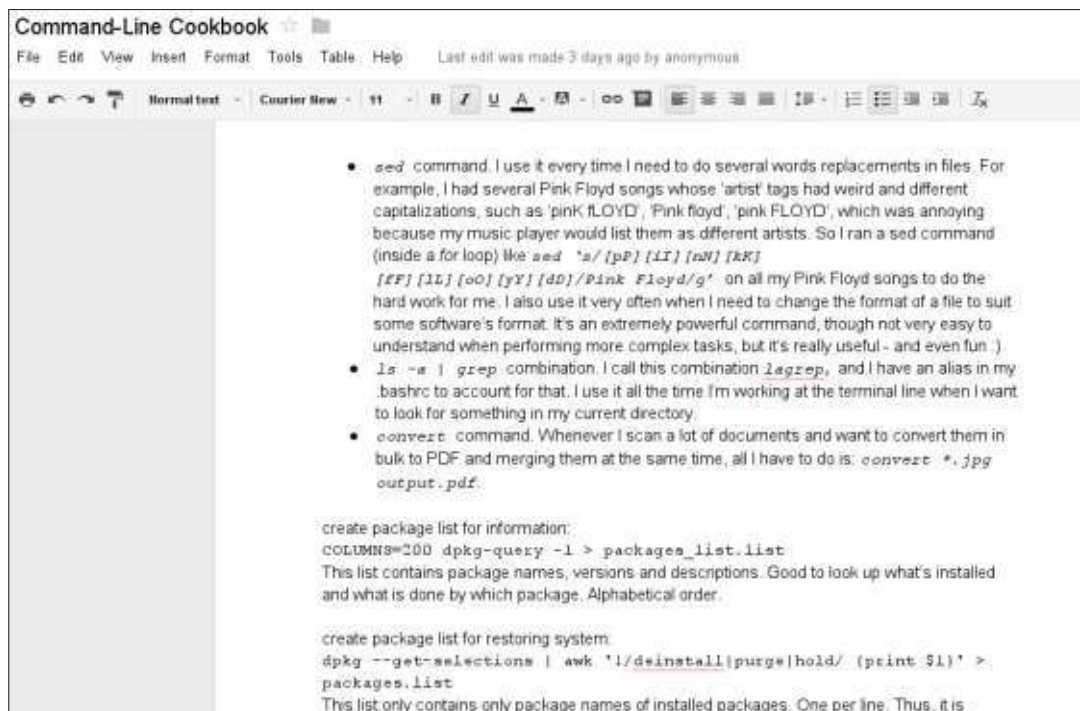
```
sudo cp text-extension.svg  
/usr/share/icons/gnome/scalable/mimetypes/
```

- Log out and back in for changes to take effect.

For our Python example, the command would look like this:

```
sudo cp text-x-python.svg  
/usr/share/icons/gnome/scalable/mimetypes/
```

Before I wrap up this article, I wanted to mention that the Command-line Cookbook document I started on Google Drive (located here: <http://goo.gl/fp09r>) has gotten quite a bit of activity. It's currently at 2.5 pages of commands. I'm going to leave it open for editing for another month, after which I



will begin to put it together. I'm currently thinking that I'll create a LaTeX document with the list, and stick it on github so that we can do our best to keep it up-to-date, as opposed to publishing it in a C&C style article. Let me know if you have any concerns with this (see the email address in the next paragraph). Also, if anyone wants me to document the process of creating the LaTeX document as a tutorial on using LaTeX, I'm more than happy to do so.

Hopefully this has successfully answered most questions you had

about assigning new defaults to files. If you have any questions, suggestions, or any other kind of feedback, feel free to email me at lswest34@gmail.com. If you do email me, please put 'C&C' or 'FCM' in the subject line, so it doesn't get lost.



Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.

HOW TO CONTRIBUTE

FULL CIRCLE NEEDS YOU!

A magazine isn't a magazine without articles and Full Circle is no exception. We need your opinions, desktops, stories, how-to's, reviews, and anything else you want to tell your fellow *buntu users. Send your articles to: articles@fullcirclemagazine.org

We are always looking for new articles to include in Full Circle. For help and advice please see the Official Full Circle Style Guide: <http://url.fullcirclemagazine.org/75d471>

Send your comments or Linux experiences to: letters@fullcirclemagazine.org
Hardware/software reviews should be sent to: reviews@fullcirclemagazine.org
Questions for Q&A should go to: questions@fullcirclemagazine.org
Desktop screens should be emailed to: misc@fullcirclemagazine.org
... or you can visit our site via: fullcirclemagazine.org

Please note:

Special editions are compiled from originals and may not work with current versions.

Full Circle Team

Editor - Ronnie Tucker
ronnie@fullcirclemagazine.org

Webmaster - Lucas Westermann
admin@fullcirclemagazine.org

Special Editions - Jonathan Hoskin

Editing & Proofreading
Mike Kennedy, Gord Campbell, Robert Orsino, Josh Hertel, Bert Jerred, Jim Dyer and Emily Gonyer

Our thanks go to Canonical, the many translation teams around the world and Thorsten Wilms for the FCM logo.

For the Full Circle Weekly News:

You can keep up to date with the Weekly News using the RSS feed: <http://fullcirclemagazine.org/feed/podcast>

Or, if your out and about, you can get the Weekly News via Stitcher Radio (Android/iOS/web):
<http://www.stitcher.com/s?fid=85347&refid=stpr>



and via TuneIn at: <http://tunein.com/radio/Full-Circle-Weekly-News-p855064/>

Getting Full Circle Magazine:

EPUB Format - Most editions have a link to the epub file on that issues download page. If you have any problems with the epub file, email: mobile@fullcirclemagazine.org

Issuu - You can read Full Circle online via Issuu: <http://issuu.com/fullcirclemagazine>. Please share and rate FCM as it helps to spread the word about FCM and Ubuntu.

Magzster - You can also read Full Circle online via Magzster: <http://www.magzter.com/publishers/Full-Circle>. Please share and rate FCM as it helps to spread the word about FCM and Ubuntu Linux.