The best new open source software on the planet

LXF HotPicks

Griffon » Puddletag » Mobik » PyHIDS » Potamus » Ioping » Dateutils » Berusky » UBee512 » Vinterm » S2png

Mike Saunders
Having visited every single nook and cranny of the internet, Mike knows where the best open source gems are hidden.

Text editor and IDE

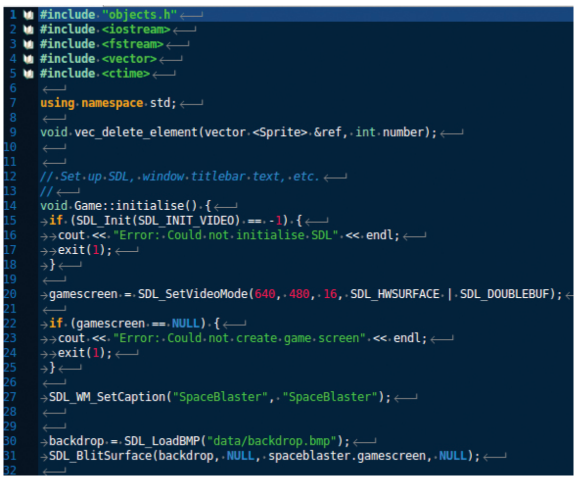
Griffon

Version 1.6 **Web** http://griffon.lasotel.fr/en

We Brits don't have a great reputation for learning foreign languages, so it always impresses us to see so much multilingualism in the open source world. We mention this because *Griffon*'s lead coder, Philippe Muller, is a Frenchman and although there are a few problems with the English translation of his program, they're minor and we applaud his effort.

Griffon is a hugely ambitious text editor with elements of an IDE (Integrated Development Environment). It attempts to cram everything you could need into a single window, producing an interface that looks horribly cluttered on low-resolution displays, but if you have plenty of screen real estate, you'll appreciate having lots of useful panels and widgets surrounding the main editing area. *Griffon* is written in C with *GTK*, and along with the source tarball a Deb package is available for Debian and Ubuntu-based distros.

The core component of Griffon is the syntax highlighting, auto-completing



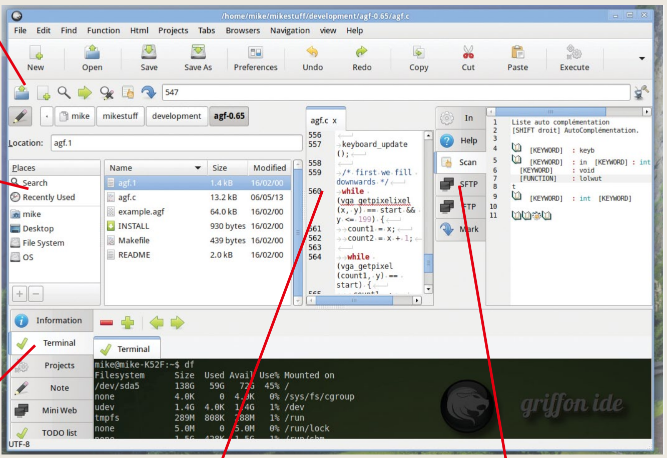
```
#include "objects.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <ctime>

using namespace std;

void vec_delete_element(vector <Sprite> &ref, int number);


// Set up SDL, window titlebar text, etc.
//
void Game::initialise() {
	if (SDL_Init(SDL_INIT_VIDEO) == -1) {
		cout << "Error: Could not initialise SDL" << endl;
		exit(1);
	}

	gamescreen = SDL_SetVideoMode(640, 480, 16, SDL_HWSURFACE | SDL_DOUBLEBUF);

	if (gamescreen == NULL) {
		cout << "Error: Could not create game screen" << endl;
		exit(1);
	}

	SDL_WM_SetCaption("SpaceBlaster", "SpaceBlaster");


	backdrop = SDL_LoadBMP("data/backdrop.bmp");
	SDL_BlitSurface(backdrop, NULL, spaceblaster.gamescreen, NULL);

```

› **Don't like the colour scheme? Go to Themes in the menu for an alternative selection of eye-pleasing photons.**

> **"A very customisable editor with a few interface rough spots."**

editor; the latter's implementation is a bit strange at first, making heavy use of the right-hand Shift key, but it becomes second nature after a while. Syntax highlighting is available for 13 languages, and *Griffon* has a special focus on HTML, with a dedicated menu for creating HTML templates, inserting Doctypes, and adding tags to your code. However, the editor can be used for virtually everything, thanks to a Projects feature which lets you associate build commands and working directories to source code files.

Exploring the Griffon interface



```
mike@mike-K52F:~$ df
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda5       138G   59G   72G  45% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            1.4G  4.0K  1.4G   1% /dev
tmpfs           289M  808K  288M   1% /run
none            5.0M     0  5.0M   0% /run/lock
```

Second toolbar
Use the blank space to enter a line number and click the blue arrow to jump to it.

File browser
A standard filesystem navigator – double-click on a source code file and it will open in the tabbed editor in the middle.

Extra tools
This area contains an embedded terminal, web page viewer, note-taker and to-do list, switchable via tabs.

Editing panel
The heart of *Griffon*. Hit the right-hand Shift key to bring up auto-complete options when typing.

Yet more tools!
Another switchable area, this time containing auto-complete keywords, language help and a place to mount SFTP shares.

Get things done

Griffon lets you mount SFTP servers in the right-hand panel, while the bottom section is home to a bunch of useful tools, such as a *WebKit*-based viewer (for previewing web page edits), a terminal and a to-do list. Ad-hoc user commands can be created by going to File > Manager configuration file > Configuring the user menu: this opens a new editor tab where you can add custom commands accessible via Function > User Menu.

In all, *Griffon* is a very customisable editor with a few interface rough spots, but plenty of neat features to pad it out.

Audio tag editor

Puddletag

Version 1.0.2 **Web** http://puddletag.sourceforge.net

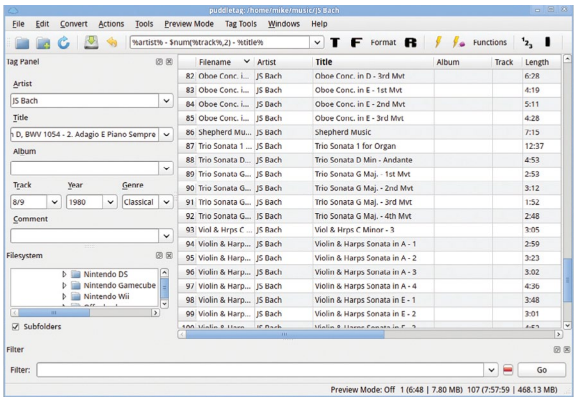
Most music players include a way to edit tags, the metadata associated with music files, but the implementation is often lacking. In many cases you have to open a new editing dialog box for each song, which is horribly tedious. *Puddletag* takes a different approach by presenting songs and their tags in a spreadsheet-like layout – so you can work on multiple files at the same time.

It's written in Python, and you'll need various modules, such as Mutagen and PyParsing (the full list is in the Readme file). Once you have these, you can run the program from its own directory with **./puddletag**. You'll be presented with a mostly blank display, so use the Filesystem panel in the bottom-left to navigate to a directory containing your songs. This will populate the main panel with tag data, and you can scroll around inside it like a typical spreadsheet – double-click on a cell to change its data.

If your music collection is lacking decent tags, you can create new tags based on filenames via the Convert menu. You can also do this the other way round, creating more descriptive file names based on the content of your tags. *Puddletag*'s Actions system is immensely versatile: you can define custom actions to modify tags based on regular expressions, merge tags together, remove whitespace and much more. These actions can be applied to batches of songs, providing an automated editing system.

With the python-musicbrainz2 module, *Puddletag* can retrieve song data from the net and tag files accordingly, and there's a life-saving

“One of the most versatile tag editors we've come across.”


puddletag: /home/mike/music/JS Bach
File Edit Convert Actions Tools Preview Mode Tag Tools Windows Help
%artist% - $num(%track%,2) - %title%
Format
Functions
Tag Panel
Artist
JS Bach
Title
1 D, BWV 1054 - 2. Adagio E Piano Sempre
Album
Track
8/9
Year
1980
Genre
Classical
Comment
Filesystem
Nintendo DS
Nintendo Gamecube
Nintendo Wii
Subfolders
Filter
Filter:
Go

	Filename	Artist	Title	Album	Track	Length
82	Oboe Conc. i...	JS Bach	Oboe Conc. in D - 3rd Mvt			6:28
83	Oboe Conc. i...	JS Bach	Oboe Conc. in E - 1st Mvt			4:19
84	Oboe Conc. i...	JS Bach	Oboe Conc. in E - 2nd Mvt			5:11
85	Oboe Conc. i...	JS Bach	Oboe Conc. in E - 3rd Mvt			4:28
86	Shepherd Mu...	JS Bach	Shepherd Music			7:15
87	Trio Sonata 1 ...	JS Bach	Trio Sonata 1 for Organ			12:37
88	Trio Sonata D...	JS Bach	Trio Sonata D Min - Andante			4:53
89	Trio Sonata G...	JS Bach	Trio Sonata G Maj. - 1st Mvt			2:53
90	Trio Sonata G...	JS Bach	Trio Sonata G Maj. - 2nd Mvt			3:12
91	Trio Sonata G...	JS Bach	Trio Sonata G Maj. - 3rd Mvt			1:52
92	Trio Sonata G...	JS Bach	Trio Sonata G Maj. - 4th Mvt			2:48
93	Viol & Hrps C ...	JS Bach	Viol & Hrps C Minor - 3			3:05
94	Violin & Harp...	JS Bach	Violin & Harps Sonata in A - 1			2:59
95	Violin & Harp...	JS Bach	Violin & Harps Sonata in A - 2			3:23
96	Violin & Harp...	JS Bach	Violin & Harps Sonata in A - 3			3:02
97	Violin & Harp...	JS Bach	Violin & Harps Sonata in A - 4			4:36
98	Violin & Harp...	JS Bach	Violin & Harps Sonata in E - 1			3:48
99	Violin & Harp...	JS Bach	Violin & Harps Sonata in E - 2			3:01

Preview Mode: Off 1 (6:48 | 7.80 MB) 107 (7:57:59 | 468.13 MB)

› **Scratching an itch: the author wrote *Puddletag* to escape *Mp3tag* on Windows and free up a partition.**

Undo button if you mess up. You can even play songs from inside the program, by default in *Amarok* (go to Edit > Preferences to change to a different player).

Puddletag is one of the most versatile tag editors we've come across, and its spreadsheet-based approach is way better than the tiresome dialog-box system normally employed by. If you've used *Mp3tag* on Windows then you'll feel right at home here; the interfaces are very similar, although there are some feature differences.

Wiki engine

Mobiki

Version 2013-07-23 **Web** http://bit.ly/1488jlw

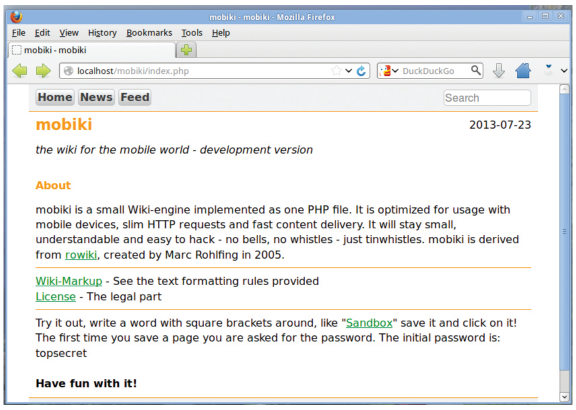
For quick document collaboration jobs, nothing beats a wiki. They're easy to use, anyone with a web browser can contribute, and admins can monitor changes to the content. The only problem is the setup: many wiki engines require the admin to create databases, fiddle around with permissions and other laborious tasks.

Mobiki is designed to be ultra simple to install – it's a single PHP file, and it writes wiki pages to individual text files, so no database is required. It's “optimised for usage with mobile devices”, hence the minimal interface when compared with the likes of *MediaWiki*, and the code is easy to hack thanks to an abundance of comments.

To get the wiki running, place its files in the document root of a web server with PHP 5.4 (or later) support (we used *Nginx* in our testing) and open **index.php** in your browser. To perform initial edits, you'll need to input a password. This can be modified in the configuration section of **index.php**. *Mobiki*'s markup isn't quite the same as *MediaWiki*'s, but it shares many similarities and a pre-built Wiki-Markup page is provided for experimentation.

The **template.html** and **template.css** files are used for customising the wiki's appearance; in the former you can modify the structure of pages, while the latter lets you fine-tune fonts and colours. *Mobiki* provides a simple News page which shows recent edits sorted by date, along with an RSS feed containing the same information. Here's where *Mobiki* feels weak in

“Mobiki is designed to be ultra simple to install – it's a single PHP file.”


mobiki - mobiki - Mozilla Firefox
File Edit View History Bookmarks Tools Help
mobiki - mobiki
localhost/mobiki/index.php
DuckDuckGo
Home
News
Feed
Search

mobiki 2013-07-23

the wiki for the mobile world - development version

About

mobiki is a small Wiki-engine implemented as one PHP file. It is optimized for usage with mobile devices, slim HTTP requests and fast content delivery. It will stay small, understandable and easy to hack - no bells, no whistles - just tinwhistles. mobiki is derived from rowiki, created by Marc Rohlfing in 2005.

Wiki-Markup - See the text formatting rules provided
License - The legal part

Try it out, write a word with square brackets around, like "Sandbox" save it and click on it! The first time you save a page you are asked for the password. The initial password is: topsecret

Have fun with it!

› **Mobiki's default theme is targeted at mobile devices.**

comparison to the larger engines: it's not possible to do comparisons with earlier page versions or revert changes.

With a quick five-line hack (saving extra copies of pages with a timestamp affixed) it's possible to add a basic versioning system, and manually copy older versions of pages over current ones if necessary. But *Mobiki* isn't designed to be a high-end engine, and for a simple, low-maintenance wiki for a small group of users, it does a solid job.

Intrusion detection system

PyHIDS 0.4

Version 0.4 **Web** http://bit.ly/14o4vr1

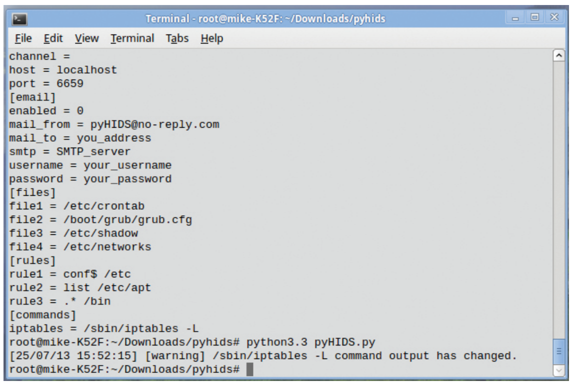
If you suspect that someone has broken into your home, what do you do? Before calling the old bill, you'll want to find some evidence that a no-gooder has been inside your property (and they are not still there too). For instance, a window that has been forcibly opened, or a broken door lock. You have a similar situation when your computer is compromised, whereby system and configuration files that have been changed without your involvement can indicate the work of an intruder.

The problem is, a typical Linux installation contains tens of thousands of files, and you can't manually check all the important ones every day. This is where an IDS comes in immensely useful: it's an Intrusion Detection System that monitors the content of your system-critical files and the output of certain commands, and lets you know if something changes. *PyHIDS* is a compact and easy to use example of an IDS, as its name suggests it's written in Python. To run it you'll need Python 3.3 and the RSA module, which are available in Debian/Ubuntu-based systems like so:

```
sudo apt-get install python3-pip
sudo pip-3.3 install rsa
```

Copy **conf.cfg-sample** to **conf.cfg** and open it in a text editor. You'll see various configuration sections marked in square brackets; first look at the [files] section. Here's where you specify the files that *PyHIDS* will keep track of, so if something changes in them, *PyHIDS* will let you know. Along with individual files you can specify directories in the [rules] section, and under [commands] you can tell *PyHIDS* to alert you when the output of certain commands changes.

Once you have *PyHIDS* configured to your liking, run **genKeys.py** and **genBase.py**. This will create a database of the files and commands that are described in **conf.cfg**. Then run *pyHIDS.py* to see if any of them has changed – try manually editing one of the files listed in **conf.cfg** and run **pyHIDS.py** again to see what the warning output looks like. *PyHIDS* can be run in a *Cron* job each day, and you'll be alerted by mail if any of the files have changed without your knowledge.


Terminal - root@mike-K52F: ~/Downloads/pyhids
File Edit View Terminal Tabs Help

```
channel =
host = localhost
port = 6659
[email]
enabled = 0
mail_from = pyHIDS@no-reply.com
mail_to = you_address
smtp = SMTP_server
username = your_username
password = your_password
[files]
file1 = /etc/crontab
file2 = /boot/grub/grub.cfg
file3 = /etc/shadow
file4 = /etc/networks
[rules]
rule1 = conf$ /etc
rule2 = list /etc/apt
rule3 = .* /bin
[commands]
iptables = /sbin/iptables -L
root@mike-K52F:~/Downloads/pyhids# python3.3 pyHIDS.py
[25/07/13 15:52:15] [warning] /sbin/iptables -L command output has changed.
root@mike-K52F:~/Downloads/pyhids# 
```

› Here we're looking at the config file, and then running *PyHIDS* – in this case the *Iptables* have changed.

> **"PyHIDS is a compact and easy to use example of an IDS."**

Audio player

Potamus

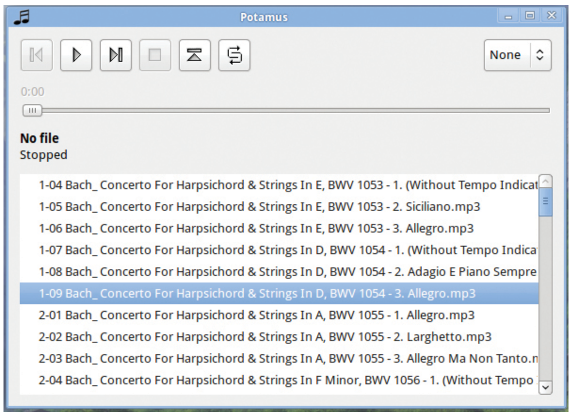
Version 14 **Web** http://offog.org/code/potamus

Some of the best open source projects were born out of frustration. It's a similar situation with *Potamus*, which is "a reaction to music players like *Rhythmbox* that use complex databases to keep track of music".

Now, you could argue that these databases are essential to give high-end players like *Rhythmbox* all the features we've come to expect. But every Linux system has a decent database for organising songs: the filesystem. So *Potamus* doesn't try to do anything fancy for sorting and managing your tunes – it doesn't extract ID3 tags or move things into categories etc. It just shows you your songs based on their filenames, but it lets you rearrange them with drag-and-drop, or drop in new songs from your file manager. There's also a Shuffle button to create a random playlist.

Potamus's interface is built with *GTK* so you'll need the development headers to compile from source, along with the development packages for **libao** and **libsamplerate**. On first start, the program shows a blank playlist – drag-and-drop songs into it or start the program with a directory containing songs as a parameter. From here, it's simple to operate: there are Play, Pause and Skip buttons along the top, and a dropdown list for switching between various mono and stereo modes.

The player supports MP3, Ogg Vorbis and FLAC files, along with 24-bit audio output and gapless playback. And that's pretty much it, feature-wise; you won't find visualisers or album artwork downloaders here. If your music collection is lumped together in a single directory and you rely on ID3 tags and pre-built playlists to manage it, then *Potamus* won't rock your world. But if you've carefully organised your songs into directories for artists and albums, and given songs descriptive filenames, it's a great little player – integrate it with your file manager, and you can right-click on a directory to play an album.



1-04 Bach_ Concerto For Harpsichord & Strings In E, BWV 1053 - 1. (Without Tempo Indicat
1-05 Bach_ Concerto For Harpsichord & Strings In E, BWV 1053 - 2. Siciliano.mp3
1-06 Bach_ Concerto For Harpsichord & Strings In E, BWV 1053 - 3. Allegro.mp3
1-07 Bach_ Concerto For Harpsichord & Strings In D, BWV 1054 - 1. (Without Tempo Indica
1-08 Bach_ Concerto For Harpsichord & Strings In D, BWV 1054 - 2. Adagio E Piano Sempre
1-09 Bach_ Concerto For Harpsichord & Strings In D, BWV 1054 - 3. Allegro.mp3
2-01 Bach_ Concerto For Harpsichord & Strings In A, BWV 1055 - 1. Allegro.mp3
2-02 Bach_ Concerto For Harpsichord & Strings In A, BWV 1055 - 2. Larghetto.mp3
2-03 Bach_ Concerto For Harpsichord & Strings In A, BWV 1055 - 3. Allegro Ma Non Tanto.n
2-04 Bach_ Concerto For Harpsichord & Strings In F Minor, BWV 1056 - 1. (Without Tempo

› *Potamus* won't win any beauty contests – it's all about the functionality.

> **"Drag-and-drop, or drop in new songs from your file manager."**

Disk I/O latency measuring tool

Ioping

Version 0.7 **Web** http://code.google.com/p/ioping

You've almost certainly heard of *ping* before, but if not: it's a small command line utility that sends short messages to other machines on a network, waits for the response, and then shows the total time taken. For example, if you run it like this:

```
ping google.com
```

Ping will send a request to **www.google.com** every second, and show how long it takes for the response to come back. Slow responses could indicate a problem with Google's servers, or some congestion on the network between your machine and Google's mammoth data centres. In any case, it's a valuable tool for system administrators, and now we have a disk I/O equivalent in the form of *Ioping*.

To compile it, extract **ioping-0.7.tar.gz**, jump into the resulting directory and enter *make*. You can now run the program in place, but you need to provide a filesystem location for testing the disk speeds. Try this:

```
./ioping .
```

Here, Ioping reads 4k of data from the current directory every second, and shows how long the reading process takes. With such a small amount of data, the results will probably be displayed in 'μs' on your machine – that is, microseconds, or millionths of a second. You can bump up the amount of data that Ioping reads like so:

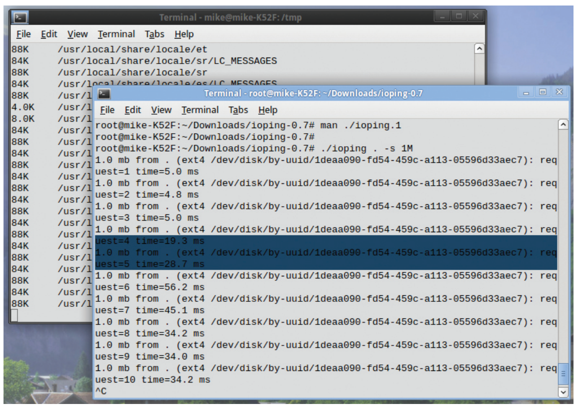
```
./ioping . -s 1M
```

This time it reads 1MB every second.

While you're running *Ioping*, try opening another terminal and doing something disk-intensive, like **du /**. You'll immediately see the effect this has on disk-read latency – the response time will jump right up. And this is where Ioping is most useful: you can run it on different storage devices at different times, and see exactly what is happening with your disk throughput. Using *top*, it's easy to see how a particular *Cron* job affects CPU usage for example, and with *Ioping* you can now see if it's slowing down disk drive accesses for other processes as well.

"It's easy to see how a Cron job affects CPU usage, for example."

Ioping has other options (enter **man ./ioping.1**) providing more detailed analysis of disk latency, and it can even generate raw data for feeding into another program for further processing.



```
88K     /usr/local/share/locale/et
84K     /usr/local/share/locale/sr/LC_MESSAGES
88K     /usr/local/share/locale/sr
```

```
root@mike-K52F:~/Downloads/ioping-0.7# man ./ioping.1
root@mike-K52F:~/Downloads/ioping-0.7#
root@mike-K52F:~/Downloads/ioping-0.7# ./ioping . -s 1M
1.0 mb from . (ext4 /dev/disk/by-uuid/1deaa090-fd54-459c-a113-05596d33aec7): req
uest=1 time=5.0 ms
1.0 mb from . (ext4 /dev/disk/by-uuid/1deaa090-fd54-459c-a113-05596d33aec7): req
uest=2 time=4.8 ms
1.0 mb from . (ext4 /dev/disk/by-uuid/1deaa090-fd54-459c-a113-05596d33aec7): req
uest=3 time=5.0 ms
1.0 mb from . (ext4 /dev/disk/by-uuid/1deaa090-fd54-459c-a113-05596d33aec7): req
uest=4 time=19.3 ms
1.0 mb from . (ext4 /dev/disk/by-uuid/1deaa090-fd54-459c-a113-05596d33aec7): req
uest=5 time=28.7 ms
1.0 mb from . (ext4 /dev/disk/by-uuid/1deaa090-fd54-459c-a113-05596d33aec7): req
uest=6 time=56.2 ms
1.0 mb from . (ext4 /dev/disk/by-uuid/1deaa090-fd54-459c-a113-05596d33aec7): req
uest=7 time=45.1 ms
1.0 mb from . (ext4 /dev/disk/by-uuid/1deaa090-fd54-459c-a113-05596d33aec7): req
uest=8 time=34.2 ms
1.0 mb from . (ext4 /dev/disk/by-uuid/1deaa090-fd54-459c-a113-05596d33aec7): req
uest=9 time=34.0 ms
1.0 mb from . (ext4 /dev/disk/by-uuid/1deaa090-fd54-459c-a113-05596d33aec7): req
uest=10 time=34.2 ms
^C
```

› **Here we can see the effect of du / in the background terminal – the latency jumps up from 5ms to over 30ms.**

Date calculating toolkit

Dateutils

Version 0.2.5 **Web** www.fresse.org/dateutils

Talk about inconsistency: there are 60 minutes in an hour, 24 hours in a day, seven days in a week, 52 weeks in a year… and then there are the 12 months, which have varying numbers of days. We're all used to this system, but it all gets very messy when you try to calculate durations between dates, or add periods of time to them. For instance, off the top of your head, what time and date is it if you add 6,000 hours to 17:00 on the 24th of February 2016?

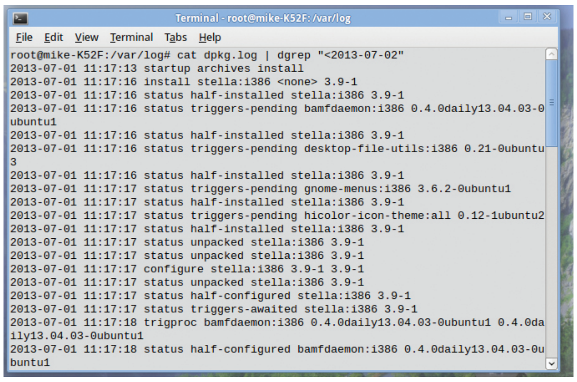
If you can answer that, well, you don't need **Dateutils**. But this toolkit is a godsend for the rest of us, and it features a small collection of programs with familiar Unixy names such as *ddiff* and *dgrep*. It has no dependencies beyond the standard C library, so it will compile virtually everywhere, and you can build it with **./configure** and **make**. If you want to install the programs system-wide, run **make install** as root; otherwise just jump into the **src** directory and run them from there.

The best way to learn how *Dateutils* works is with some examples, so here's how to solve the question posed earlier:

```
dadd 2016-02-24T17:00 +6000h
```

As you can guess from the name, **dadd** performs additions on dates and times, and in this case we're specifying the year, month and date. It's possible to work solely with dates, but by adding **T** we can use times as well, in 24-hour clock format. So after specifying the exact moment in time, we add the 6,000 hours – it's also possible to add or subtract days (eg **+3d**), minutes (**-2m**), months (**+6mo**) and so forth. Then there's **ddiff**, which shows the duration between two dates. If you enter **ddiff 2016-02-24 2016-08-15**, for example, you'll get 173 (the number of days between the dates). If you add exact times (eg **ddiff 2016-02-24T12:00 2016-08-15T13:00**) then you'll see the result in seconds, but you can format this to something more useful with the **-f** flag. Using **-f '%m %d'** shows the result split into months and days. *Dateutils* has other handy tools for searching and converting times. see **Readme.md** for the list.

"Work solely with dates or by adding T you can use times as well."



```
root@mike-K52F:/var/log# cat dpkg.log | dgrep "<2013-07-02"
2013-07-01 11:17:13 startup archives install
2013-07-01 11:17:16 install stella:i386 <none> 3.9-1
2013-07-01 11:17:16 status half-installed stella:i386 3.9-1
2013-07-01 11:17:16 status triggers-pending bamfdaemon:i386 0.4.0daily13.04.03-0
ubuntu1
2013-07-01 11:17:16 status half-installed stella:i386 3.9-1
2013-07-01 11:17:16 status triggers-pending desktop-file-utils:i386 0.21-0ubuntu
3
2013-07-01 11:17:16 status half-installed stella:i386 3.9-1
2013-07-01 11:17:17 status triggers-pending gnome-menus:i386 3.6.2-0ubuntu1
2013-07-01 11:17:17 status half-installed stella:i386 3.9-1
2013-07-01 11:17:17 status triggers-pending hicolor-icon-theme:all 0.12-1ubuntu2
2013-07-01 11:17:17 status half-installed stella:i386 3.9-1
2013-07-01 11:17:17 status unpacked stella:i386 3.9-1
2013-07-01 11:17:17 status unpacked stella:i386 3.9-1
2013-07-01 11:17:17 configure stella:i386 3.9-1 3.9-1
2013-07-01 11:17:17 status unpacked stella:i386 3.9-1
2013-07-01 11:17:17 status half-configured stella:i386 3.9-1
2013-07-01 11:17:17 status triggers-awaited stella:i386 3.9-1
2013-07-01 11:17:18 trigproc bamfdaemon:i386 0.4.0daily13.04.03-0ubuntu1 0.4.0da
ily13.04.03-0ubuntu1
2013-07-01 11:17:18 status half-configured bamfdaemon:i386 0.4.0daily13.04.03-0u
buntu1
```

› **Use dgrep to search log files for events that happened before or after a specific date/time.**

HotGames Entertainment apps

Bug-based puzzle game

Berusky

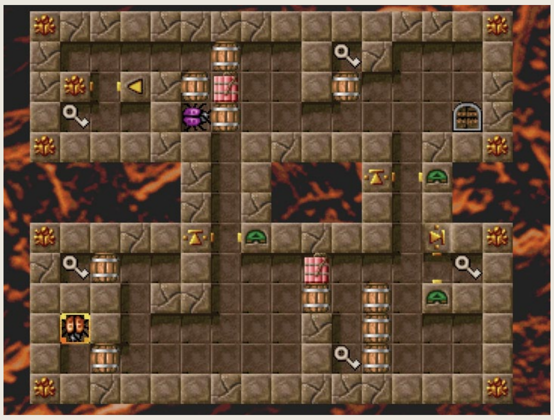
Version 1.7 **Web** http://anakreon.cz/?q=node/1

We thought our internet-sifting capabilities made the NSA look like amateurs, but somehow *Berusky* never came under our radar. It's a block-pushing puzzle game along the lines of *Sokoban*, but with heaps of extra features and fiendishly designed levels – oh, and its only major dependency is *SDL*, so it's easy to install. Extract **berusky-1.7.tar.gz** and build it with the usual **./configure**, **make** and **make install** (as root) procedure; then extract the data tarball and run **./install** inside.

Start the game by entering **berusky**, and click Play to be presented with a list of level categories. *Berusky* includes 120 levels, and helpfully there's a Training series which takes you through the basics. In summary: you control one or more bugs on a 2D playing field, and your job is to collect five keys scattered around before making your way to the exit. On levels where you can control more than one bug, hit the Tab key to switch between them.

Berusky's challenge comes from the various objects that stand between you and the keys. Stones block your path (collect a pick-axe to destroy them) and barrels hinder progress (push dynamite into them). Doors close behind you and trap you. Completing levels is rarely a straightforward affair – you need to plan your progress. The level design is excellent, and you often find yourself staring at a new level for a while to get your bearings, before even daring to make the first move.

Our only minor gripe is that the carefully crafted pixel art is tiny, so the game is almost impossible to play on hi-res screens. A quick fix for this is to enable High Resolution mode in the Settings – this doubles up each pixel, making the game look a bit blockier, but far easier to see.


› This is one of the easier levels, but it still gave our brain a good workout.

"A block-pushing puzzle game… with heaps of extra features."

MicroBee emulator

uBee512

Version 5.5.0 **Web** www.microbee-mspp.org.au

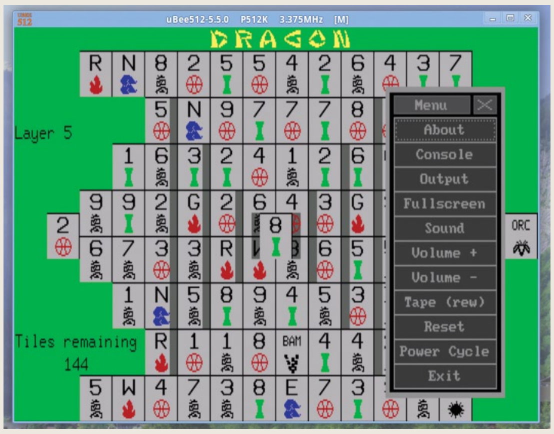
If you're one of our Australian readers, or you have a soft spot for the home computers of the 1980s (or best of all, both of these apply), then this is for you. The MicroBee was released down under in 1982, originally in kit format but later as a pre-built system, and sported a Z80 processor running at 2MHz along with 16k of RAM.

The MicroBee series was largely unknown outside of Australia and discontinued in 1990, but it still has fans. There's even the MicroBee Software Preservation Project, which aims to "preserve the old technology for future generations". Core to this effort is *UBee512*, a MicroBee emulator that runs on Linux, Windows and Amiga OS.

UBee512 is available in Deb and RPM formats, along with the source code, and to get it running you'll need MicroBee ROM images. These are provided by the MSPP at **www.microbee-mspp.org.au/technology_public/** – grab the **BASIC**, **charrom** and **rom1** files, and place them in **~/.ubee512/roms/**. Last, you'll need a **boot.disk** to start up the machine, examples of which are available at **www.microbee-mspp.org.au/repository**.

We tried **games_demo_ds80.dsk**, which includes various puzzlers such as Mah Jong, dominoes and *Tetris*. Other games can be found on the internet, such as the classic *Hunt the Wumpus*, with the usual complications about copyright (download at your own risk). *UBee512* includes its own mini toolkit and windowing system for configuration – right-click inside the emulator window to switch to full-screen mode, change the sound volume and power-cycle the system.

Given the MicroBee's obscurity when compared to the ZX Spectrum and C64, it's great to see an emulator for it being actively developed.



› The *MicroBee* made a minor comeback in 2011 with a limited edition (105 units) Premium Plus kit version.

"Available in Deb and RPM formats, along with the source code."

Vintage-style terminal emulator

Vinterm

Version 0.4.0 **Web** http://bit.ly/11Rm0Pn

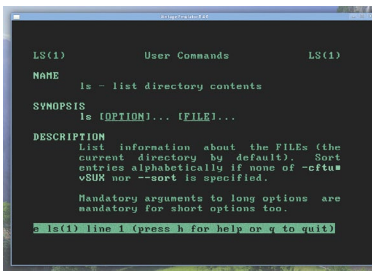
Remember *Terminology* [LXF174, p48], the super-snazzy terminal from the *Enlightenment* team? Well, here's pretty much the exact opposite. Whereas *Terminology* spruces up the command line with all sorts of forward-looking features like image previews and video playing, *Vinterm* takes you back to the 1980s, with fuzzy monitors and green-on-black displays. It's totally retro and we like it.

Vinterm's only major dependency is *SDL*, and to compile it you just issue a single **make** command. Then run it in place with **./vinterm**. Straight away you'll see how the program attempts to recreate the displays of yore, with a blurry effect added to the fading green text. To see the effect in more detail, use the **-s** flag to increase the scaling (zoom) level, eg:

```
./vinterm -s 2
```

Note that if your default shell is *Bash* (as is the case in most Linux distros), *Vinterm* won't source your **~/.bashrc** when you start it, leaving you with a very bare environment – you won't have your aliases for instance. Just enter **bash** (or your normal shell) to fix this. *Vinterm* doesn't support colours, but it can show bold and inverted text as used in **man** for instance. There's also a full-screen mode, accessible via Shift+Ctrl+F11.

And that's it for now, but the developer plans to introduce emulation of other monitor and TV types, and more configuration options to toy with.



```
LS(1)                 User Commands                 LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the
       current  directory  by  default).   Sort
       entries alphabetically if none of -cftu■
       vSUX nor --sort is specified.

       Mandatory arguments to long options  are
       mandatory for short options too.

e ls(1) line 1 (press h for help or q to quit)
```

› *Vinterm*'s retrotastic visual style is based on the IBM 5151, as shipped with the original PC.

Convert binary files to PNGs

S2png

Version 0.05 **Web** http://bit.ly/16zScGW

Steganography, the technique of embedding hidden messages inside images or other media, is a fascinating subject. It has been used throughout history for clandestine communication, and with modern digital images it's possible to hide large amounts of data inside a seemingly innocent file.

S2png (something to PNG) has a lot in common with steganography tools, but with one major difference: it doesn't try to hide anything. Instead, the resulting images that it generates scream 'Look at me! I contain a file.' You can store any kind of file inside a PNG image with *S2png*, but the picture states that it's a carrier for other data.

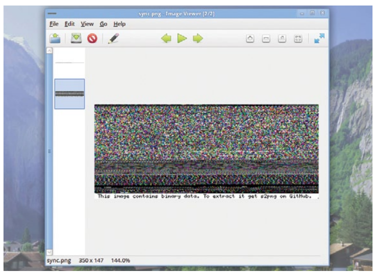
To compile it, just enter **make** – and if you receive a nasty warning about missing libraries, try adding **-lm** to the LIBS line of the Makefile:

```
LIBS = `libpng-config --libs` -lgd -lm
```

To use *S2png*, simply run its binary followed by a file. For instance:

```
./s2png myfile.txt
```

This will generate **myfile.txt.png** which you can now upload to an image gallery. (To decode the file use the **-d** switch). It's a useful way to share data on a message board which only allows image uploads, and by default each image contains a text message explaining how to get the software. For a thin layer of security, it's possible to password protect images as well. LXF



This image contains binary data. To extract it get s2png on GitHub.

› The resulting images look like random noise, but they can be viewed everywhere.

Also released

New and updated software that also deserves a look...

downtimed 0.6
Monitor operating system downtime, uptime, shutdowns and crashes.
http://dist.epipe.com/downtimed

storeBackup 3.4
Backup utility with compression that stores files on other disks.
www.storebackup.org

Giada 0.7.2
A minimal, hardcore loop machine for DJs and live performers.
www.giadamusic.com

› *Giaga* has lots of shiny knobs and buttons to play with.

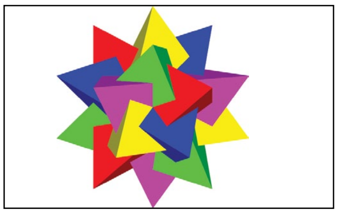
Converseen 0.6.4
Qt-based image converter and resizer, supporting over 100 formats.
http://converseen.sourceforge.net

DOMjudge 3.4.0
An automated judgement system to run programming contests.
http://domjudge.sourceforge.net

Aletheia 0.1.4
Debug web applications by sending raw HTTP requests.
http://aletheia.k42b3.com

Data Crow 3.11.0
Manage all your various collections from movie, video, book, image, software, and music.
www.datacrow.net

GeomSpace 0.12
Interactive geometry software for different geometric spaces.
http://sourceforge.net/projects/geomspace



› If Euclidian space isn't rocking your boat switch to Minkowski.