

Logging | IRC | Plasma Active | Linode

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

FREE TO SUBSCRIBERS
Epub, Kindle, Android, iPhone & iPad editions

HOW TO: INSTALL LINUX ON AN EFI COMPUTER

MARCH 2012 | ISSUE 215 | www.linuxjournal.com

MOBILE

CODE IN THE CLOUD

with an iPad + Linode

GET MORE FROM YOUR TABLET

with KDE's New Desktop



Tips to Make IRC More Mobile-Friendly

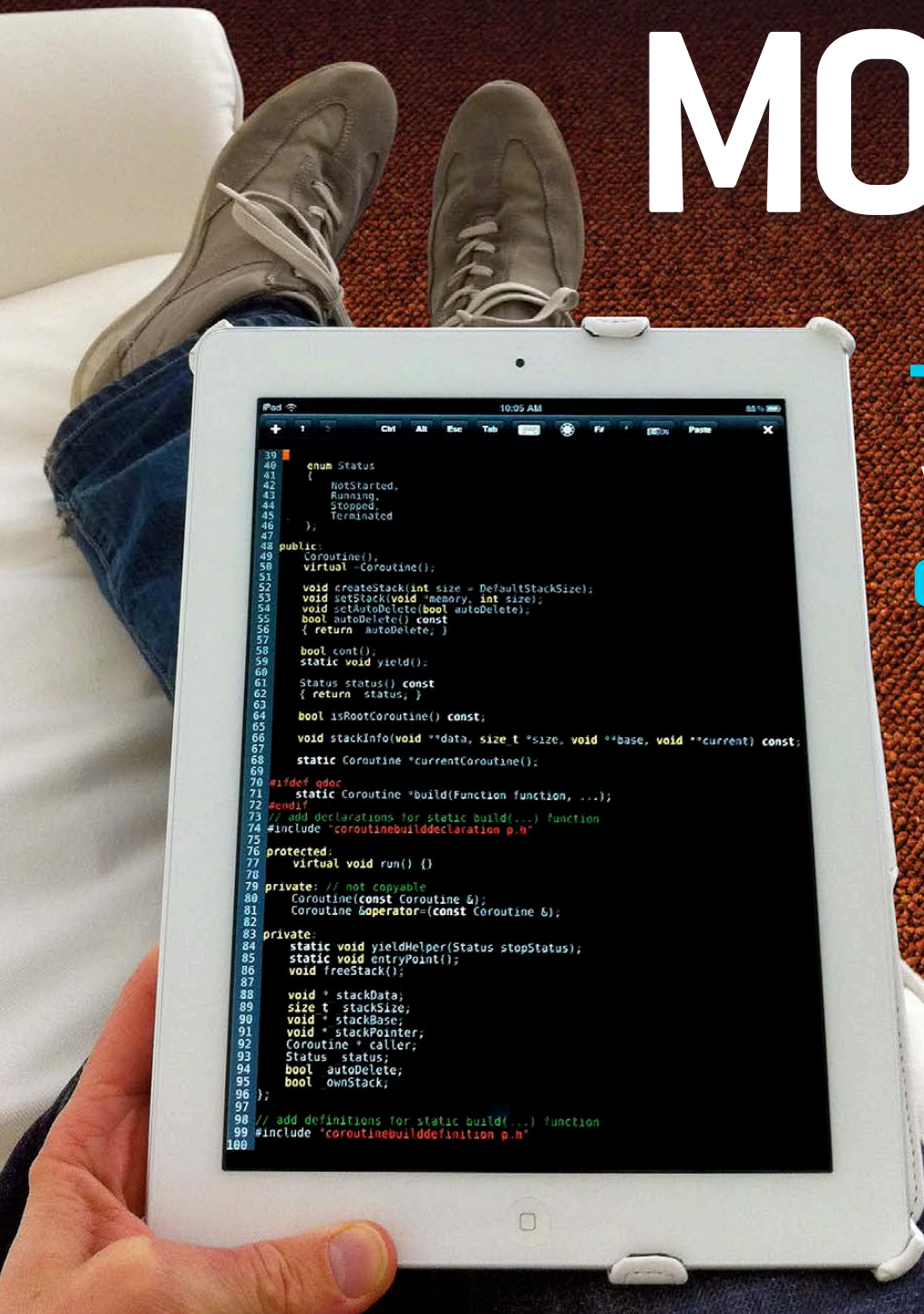
Logfile Skills for Web Developers

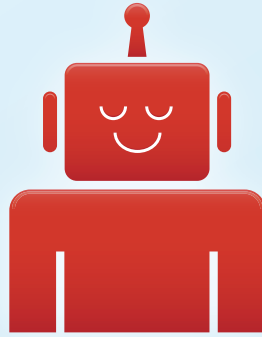
Explore the New Face of Thin Clients

```

39
40 enum Status
41 {
42     NotStarted,
43     Running,
44     Stopped,
45     Terminated
46 };
47
48 public:
49     Coroutine();
50     virtual ~Coroutine();
51
52     void createStack(int size = DefaultStackSize);
53     void setStack(void *memory, int size);
54     void setAutoDelete(bool autoDelete);
55     bool autoDelete() const
56     { return autoDelete; }
57
58     bool cont();
59     static void yield();
60
61     Status status() const
62     { return status; }
63
64     bool isRootCoroutine() const;
65
66     void stackInfo(void **data, size_t *size, void **base, void **current) const;
67
68     static Coroutine *currentCoroutine();
69
70 #ifdef qdoc
71     static Coroutine *build(Function function, ...);
72 #endif
73 // add declarations for static build(...) function
74 #include "coroutinebuilddeclaration.p.h"
75
76 protected:
77     virtual void run() {}
78
79 private: // not copyable
80     Coroutine(const Coroutine &);
81     Coroutine &operator=(const Coroutine &);
82
83 private:
84     static void yieldHelper(Status stopStatus);
85     static void entryPoint();
86     void freeStack();
87
88     void * stackData;
89     size_t stackSize;
90     void * stackBase;
91     void * stackPointer;
92     Coroutine * caller;
93     Status status;
94     bool autoDelete;
95     bool ownStack;
96 };
97
98 // add definitions for static build(...) function
99 #include "coroutinebuilddefinition.p.h"
100

```





Lullabot™

Learn Drupal & jQuery

FROM THE COMFORT OF
YOUR LIVING ROOM



The Lullabot Learning Series includes everything you need to become a Drupal & jQuery expert from the comfort of your living room! The videos are available in both DVD format and high-definition video download.

Purchase the videos at <http://store.lullabot.com>



visit us at www.siliconmechanics.com
or call us toll free at 866-352-1173

Silicon Mechanics Announces Research Cluster Grant

**Be sure to visit www.siliconmechanics.com on March 15
to find out who the grant recipient is!**



*We are pleased to announce our sponsorship
of a unique grant opportunity:
a complete high-performance
compute cluster using the latest
AMD Opteron™ processors and NVIDIA® GPUs.*

This grant program is open to all US and Canadian qualified post-secondary institutions, university-affiliated research institutions, non-profit research institutions, and researchers at federal labs with university affiliations.

To download the complete rules, application, and hardware specification, visit

www.siliconmechanics.com/research_cluster_grant
or email

research-grant@siliconmechanics.com

Silicon Mechanics would also like to thank the many hardware partners that have made this grant possible.



**When you partner with Silicon Mechanics,
you get more than affordable, high-quality
HPC — you get a team of Experts dedicated
to the advancement of scientific research.**

Expert included.

CONTENTS

MARCH 2012
ISSUE 215

MOBILE

FEATURES

62 Swap Your Laptop for an iPad + Linode

Coding in the cloud.

Mark O'Connor

74 Plasma Active—a New Approach to Tablet Computing

KDE takes on Android and Apple.

Stuart Jarvis

86 Seamlessly Extending IRC to Mobile Devices

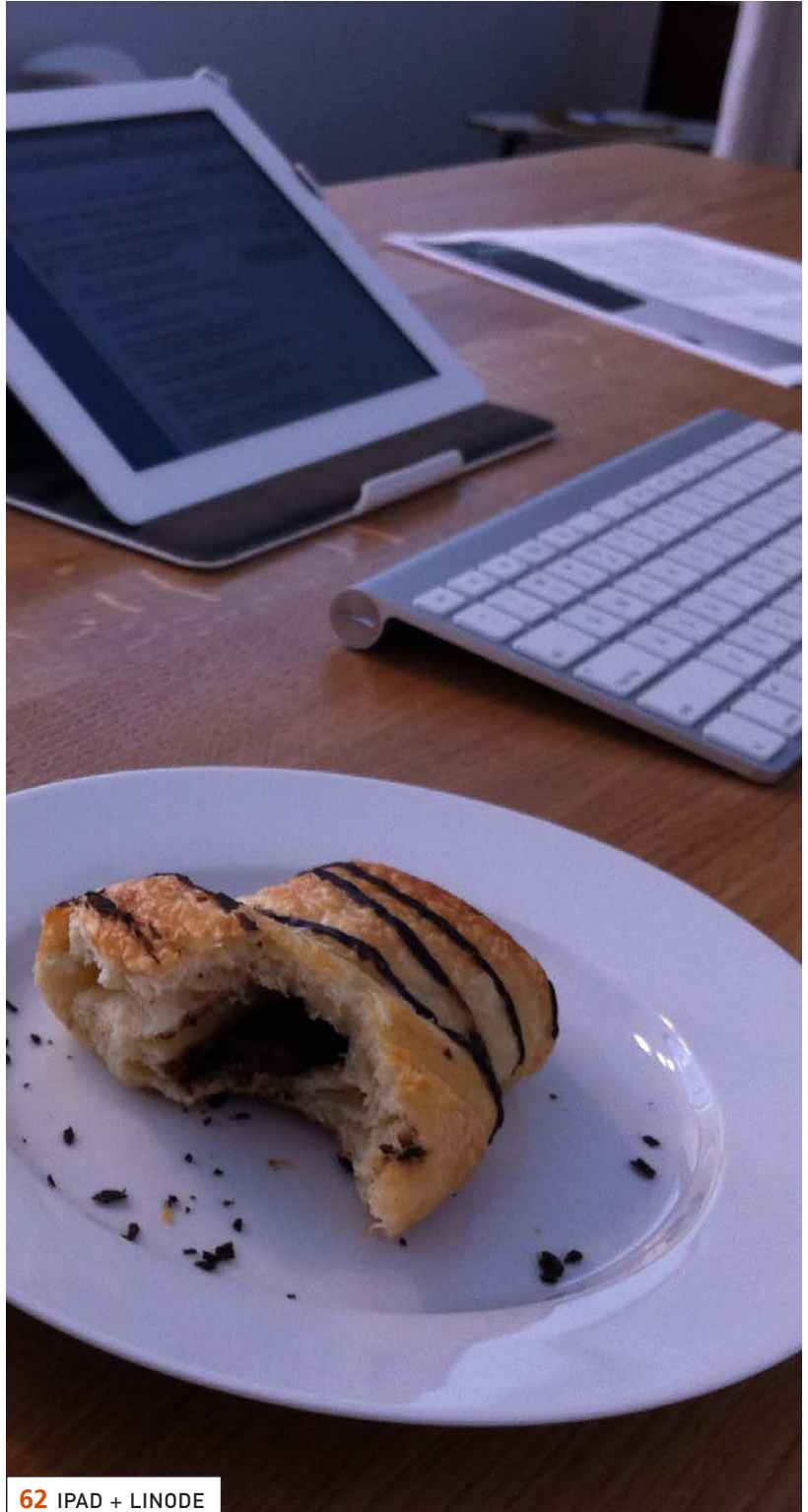
Love IRC? Want to make it more mobile-friendly? Here's how!

Bill Childers

ON THE COVER

- Fine-Tune GPU Password Cracking, p. 44
- How To: Install Linux on an EFI Computer, p. 94
- Code in the Cloud with an iPad + Linode, p. 62
- Get More from Your Tablet with KDE's New Desktop, p. 74
- Tips to Make IRC More Mobile-Friendly, p. 86
- Logfile Skills for Web Developers, p. 32
- Explore the New Face of Thin Clients, p. 50

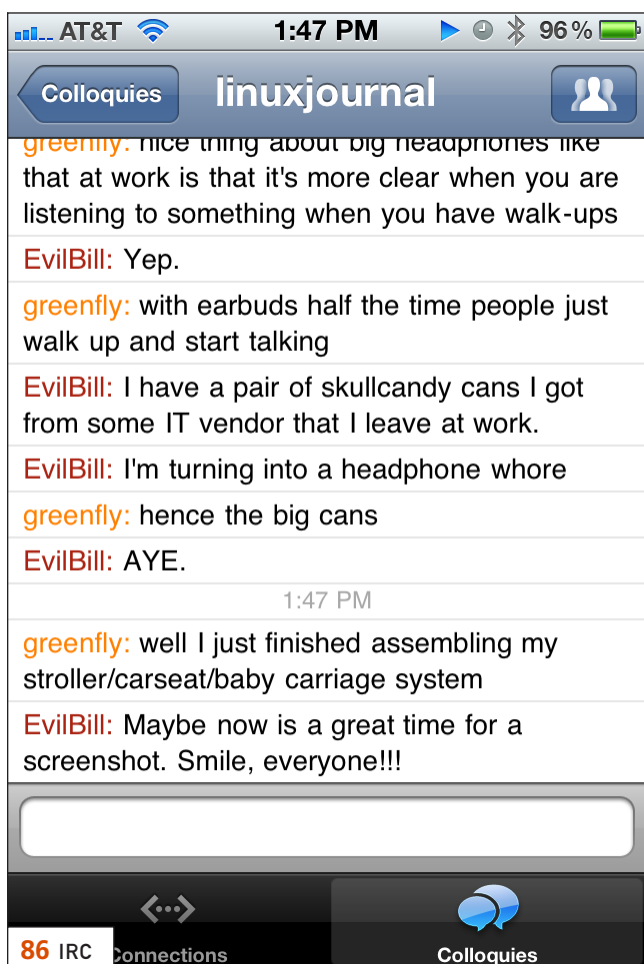
Cover photo courtesy of Mark O'Connor.



62 IPAD + LINODE

COLUMNS

- 32 Reuven M. Lerner's At the Forge**
Logging
- 40 Dave Taylor's Work the Shell**
A Word Finder for *Words With Friends*
- 44 Kyle Rankin's Hack and /**
Password Cracking with GPUs, Part III:
Tune Your Attack
- 50 Shawn Powers' The Open-Source Classroom**
LTSP, Part I: the Skinny on Thin Clients
- 116 Doc Searls' EOF**
Does Linux Have an Economy?

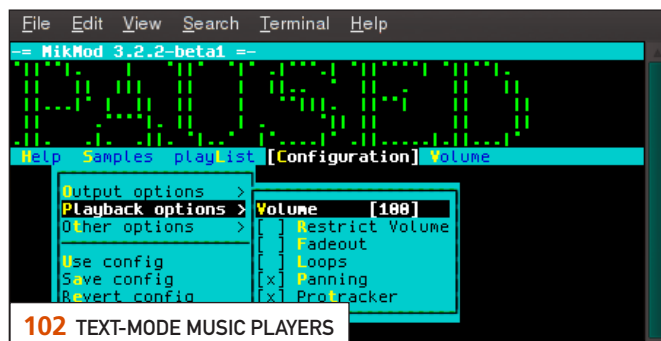
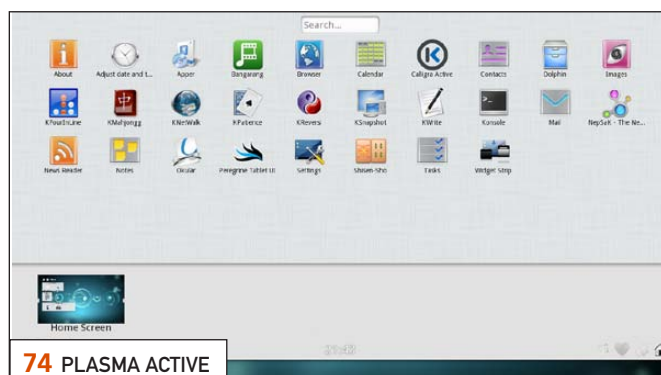


INDEPTH

- 94 Using Linux with EFI, Part III: Installing Linux on an EFI Computer**
The nuts and bolts of installing Linux on an EFI computer.
Roderick W. Smith
- 102 Rock Out with Your Console Out**
The many faces of text-mode music players.
Rebecca Ruji Chapnik

IN EVERY ISSUE

- 8 Current_Issue.tar.gz**
- 10 Letters**
- 20 UPFRONT**
- 58 New Products**
- 117 Advertisers Index**



LINUX JOURNAL™

Subscribe to
Linux Journal
Digital Edition
for only
\$2.45 an issue.



ENJOY:

- Timely delivery
- Off-line reading
- Easy navigation
- Phrase search and highlighting
- Ability to save, clip and share articles
- Embedded videos
- Android & iOS apps, desktop and e-Reader versions

SUBSCRIBE TODAY!

LINUX JOURNAL

Executive Editor	Jill Franklin jill@linuxjournal.com
Senior Editor	Doc Searls doc@linuxjournal.com
Associate Editor	Shawn Powers shawn@linuxjournal.com
Art Director	Garrick Antikajian garrick@linuxjournal.com
Products Editor	James Gray newproducts@linuxjournal.com
Editor Emeritus	Don Marti dmarti@linuxjournal.com
Technical Editor	Michael Baxter mab@cruzio.com
Senior Columnist	Reuven Lerner reuven@lerner.co.il
Security Editor	Mick Bauer mick@visi.com
Hack Editor	Kyle Rankin lj@greenfly.net
Virtual Editor	Bill Childers bill.childers@linuxjournal.com

Contributing Editors

Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti • Ludovic Marcotte
Paul Barry • Paul McKenney • Dave Taylor • Dirk Elmendorf • Justin Ryan

Proofreader Geri Gale

Publisher Carlie Fairchild
publisher@linuxjournal.com

Advertising Sales Manager Rebecca Cassity
rebecca@linuxjournal.com

Associate Publisher Mark Irgang
mark@linuxjournal.com

Webmistress Katherine Druckman
webmistress@linuxjournal.com

Accountant Candy Beauchamp
acct@linuxjournal.com

**Linux Journal is published by, and is a registered trade name of,
Belltown Media, Inc.**

PO Box 980985, Houston, TX 77098 USA

Editorial Advisory Panel

Brad Abram Baillio • Nick Baronian • Hari Boukis • Steve Case
Kalyana Krishna Chadalavada • Brian Conner • Caleb S. Cullen • Keir Davis
Michael Eager • Nick Faltys • Dennis Franklin Frey • Alicia Gibb
Victor Gregorio • Philip Jacob • Jay Kruizenga • David A. Lane
Steve Marquez • Dave McAllister • Carson McDonald • Craig Oda
Jeffrey D. Parent • Charnell Pugsley • Thomas Quinlan • Mike Roberts
Kristin Shoemaker • Chris D. Stark • Patrick Swartz • James Walker

Advertising

E-MAIL: ads@linuxjournal.com
URL: www.linuxjournal.com/advertising
PHONE: +1 713-344-1956 ext. 2

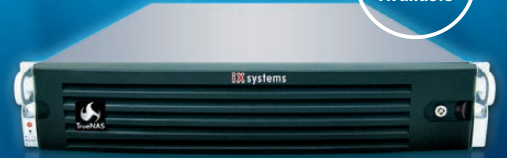
Subscriptions

E-MAIL: subs@linuxjournal.com
URL: www.linuxjournal.com/subscribe
MAIL: PO Box 16476, North Hollywood, CA 91615-9911 USA

LINUX is a registered trademark of Linus Torvalds.

TrueNAS™ 2U Appliance: You Are the Cloud

Expansion
Shelves
Available



Storage. Speed. Stability.

With a rock-solid FreeBSD® base, Zettabyte File System (ZFS) support, and a powerful Web GUI, TrueNAS™ pairs easy-to-manage FreeNAS™ software with world-class hardware and support for an unbeatable storage solution. In order to achieve maximum performance, the TrueNAS™ 2U System, equipped with the Intel® Xeon® Processor 5600 Series, supports Fusion-io's Flash Memory Cards and 10 GbE Network Cards. Titan TrueNAS™ 2U Appliances are an excellent storage solution for video streaming, file hosting, virtualization, and more. Paired with optional JBOD expansion units, the TrueNAS™ System offers excellent capacity at an affordable price.

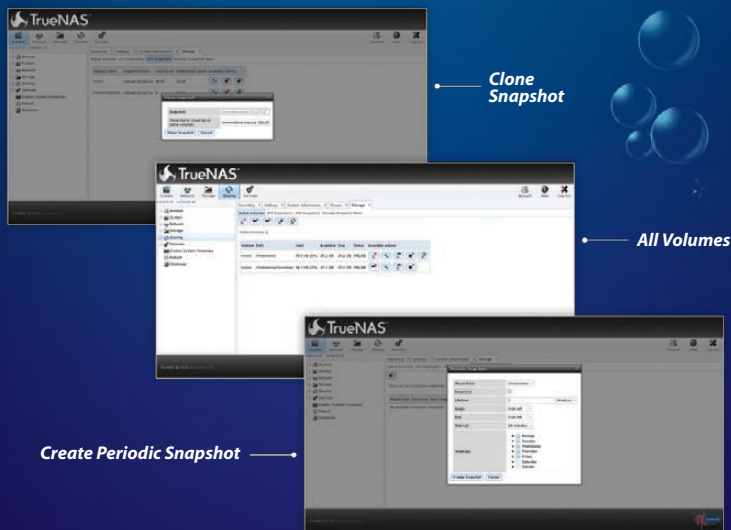
For more information on the **TrueNAS™ 2U System**, or to request a quote, visit: <http://www.iXsystems.com/TrueNAS>.

KEY FEATURES:

- Supports One or Two Quad-Core or Six-Core, Intel® Xeon® Processor 5600 Series
- 12 Hot-Swap Drive Bays - Up to 36TB of Data Storage Capacity*
- Periodic Snapshots Feature Allows You to Restore Data from a Previously Generated Snapshot
- Remote Replication Allows You to Copy a Snapshot to an Offsite Server, for Maximum Data Security
- Software RAID-Z with up to Triple Parity
- 2 x 1GbE Network interface (Onboard) + Up to 4 Additional 1GbE Ports or Single/Dual Port 10 GbE Network Cards

JBOD expansion is available on the 2U System

* 2.5" drive options available; please consult with your Account Manager



Call iXsystems toll free or visit our website today!
1-855-GREP-4-IX | www.iXsystems.com





SHAWN POWERS

Mobile Computing: When Size Matters

Technology is funny. Not too many years ago, the goal was to make a cell phone as tiny as technologically possible. Now, phones like the Galaxy Note are striving for aircraft-carrier size. This month, whether you want to embed a phone into your sneaker or play hopscotch in order to dial your buddy, we've got you covered. Mobile devices can do almost anything traditional computer systems can do, and oftentimes, they can do it better.

Reuven M. Lerner starts the issue off with logging. No, he doesn't show how to cut down trees with our Razer, but rather he talks about the importance of making applications that keep a log. Logs are really pointless, until you need them. Then, they're invaluable. If you need more convincing, listen to Reuven; you can trust him to lead you in the right direction. Dave Taylor, on the other hand, I don't recommend trusting—at least not in a game of

Scrabble. Dave continues his series on how to be a lying, cheating, filthy, jerk—for educational purposes only, of course! In all seriousness, Dave explores some really cool scripting using a very practical, if nefarious, object lesson.

Our king of nefarious, Kyle Rankin, finishes his series on password cracking in this issue. By this time, you've all learned how to do brute-force attacks with a GPU, so Kyle spends this month explaining how to tweak things so you can get the most hack for your buck. I follow Kyle's "educational" article with the second installment of my new column, The Open-Source Classroom. This month, I start a series on LTSP. Thin clients have evolved a lot since I started using them back in 2001 or so. I'll walk you through setting up a lab, and in the next few issues, I'll teach you how to tweak the system. Kyle probably will follow up with a tutorial on using the

As someone who constantly thinks tablet computers would be great if they had hinges and keyboards, I'm interested in alternative interfaces!

distributed CPU power of thin clients to break in to the local 7-11, but you'll have to wait and see.

Mark O'Connor shows how to use Linux on an iPad. No, probably not how you think, but rather, he explains how to use Linode on an iPad in order to do your work in the cloud. If you want the convenience of an iPad with the power and flexibility of Linux, Mark's solution is worth a look. Bill Childers does a similar feat with his article on IRC proxying to mobile devices. I've been using Irssi in a Screen session for all my instant messaging for a few months now, but I'll admit it's rough when I'm out and about. Logging in to Irssi on a software-keyboard over SSH isn't terribly fun on a phone. Bill describes how to get the best of both worlds, and at the same time!

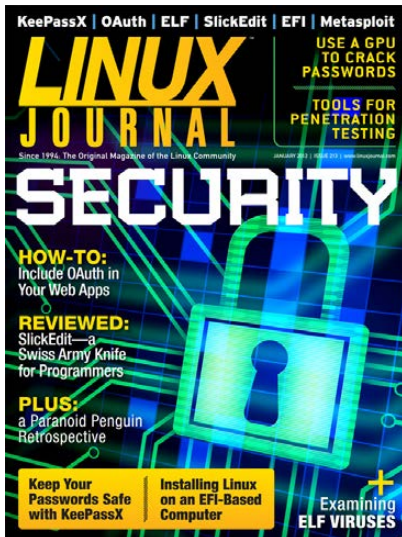
Rebecca "Ruji" Chapnik also delves into the command line, but instead of bridging IRC to a mobile device, she shows how to play music from the console. Many Linux users think Ncurses is as GUI as an application ever needs to get (ahem, Kyle Rankin), and Rebecca shows how to use the

command line to its fullest extent. Stuart Jarvis heads in the opposite direction and talks about Plasma Active. Tablet computing is still quite young, and the interfaces we use on touchscreen devices are far from perfect. Stuart describes what KDE is doing to address tablets and touchscreen devices. As someone who constantly thinks tablet computers would be great if they had hinges and keyboards, I'm interested in alternative interfaces!

Don't worry if you prefer your Linux more "desktoppy" than mobile. This is *Linux Journal*, and we always have a variety of articles that will tickle every geek's interest. Whether you want to continue the series on EFI with Roderick W. Smith, or explore the world of cryptocurrency with me, this issue has lots to offer. As always, we hope you enjoy this issue, we sure did. ■

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

letters



Kindle Edition of LJ

I love the .mobi version of LJ. It makes reading LJ on the Kindle absolutely fantastic. But, I've got

one request: every Kindle account has an @kindle.com e-mail to which you can send PDF/.mobi/... files, and they are added to your Kindle automatically when you switch on Wi-Fi. So instead of sending an e-mail with download links, can you make it an option to send the .mobi version straight to my @kindle.com address? Then, I'll just have to turn on the Wi-Fi on my Kindle, and I'll be able read the newest issue of LJ.

—Ward

Hmm, I'll make sure our distribution folks see your request. Another option might be a script that pulls links from your e-mail, downloads the .mobi, and then re-sends it to a specified account as an attachment. Perhaps it's a good challenge for Dave Taylor!—Ed.

Ultimate Linux Box

Browsing through the e-copies of my subscription, I found issue 148, the Ultimate Linux Box issue from 2006. Back then, dual-core processors were just becoming affordable for the common user, AMD launched its AM2 platform with the 939 socket, and DDR2 was emerging as the new standard in memory speed and performance. That was nearly six years ago. Maybe it's time to revisit the subject and make a comparison with today's technologies that make the Ultimate Linux Box? (And, (dis)prove the predictions of 2006 for the years ahead with the benefit of history.)

Having built a fair number of computers myself, mainly for myself, I could send you some proposed configurations if you'd like. Unfortunately, I didn't win the lottery, again, so I don't have the budget to build them myself and put them through their passes/benchmarks.

—Vincent Westein

We did stop with the Ultimate Linux Box a while back, for several reasons. The biggest one was that "Ultimate" became so relative. For our Christmas-wish list, we did throw a configuration together, but I suspect it wouldn't be ideal for everyone. I agree though. I miss the

issues highlighting rigs I'd never be able to afford. It was like looking at a geeky hot-rod magazine.—Ed.

Electronic vs. Print

Although I completely understand the economic necessity of publishing electronically—after all, you prepare, edit and produce the copy electronically, and it's easy enough to generate multiple formats for consumption on any number of devices—I have concerns, or perhaps observations might be the right word, about the longevity of the material.

For many years I have kept issues of magazines—*National Geographic*, *Scientific American*, *Creative Computing* (remember that one?), *Linux Journal* and a few others. I have also, at least in the case of *Linux Journal*, bought the CD-ROM archive. I did buy the *National Geographic* every-issue-since-volume 1, number 1. I have 9-track tapes (UNIX System V, World Data Bank II and a couple others). I have cassette tapes (music), reel-to-reel tapes (live performances), CD-ROMs, DVDs, Blue-ray discs and a few disk drives lying about along with a personal library of some 2,000 books.

One thing I've noticed: I still can

read the hard copies, but most of the magnetic media is gone. Many CD-ROMs are gone, but I still can play records from the 1940s through today, and they "sound just as good" as they did then. I was more than a little surprised to discover that CD-ROM media just five years old had become useless, and a small collection of laser discs are unwatchable. So-called archive media, supposed to last decades (decades!?) is turning out to be somewhat less long-lived.

I have stood in the Bodleian Library at Oxford with a 600-year-old book in my hands (with gloves, of course) and read every word. I have stood in the British Museum in London and read a letter written by Elizabeth Tudor to her half-brother, then king of England, in the 16th century. I have admired Egyptian scrolls dating from 4,000 years ago (not decades, mind, millenniums).

I have long been painfully aware that the attention span of a computer is only as long as its power cord. I am becoming aware that our attention spans are ever shorter as our gadgets display fleeting glimpses of our civilization, glances destined to go "poof" in the ether when the battery goes dead. Alas.

—**Thomas Ronayne**

[LETTERS]

Your concern is, unfortunately, very valid. I say this as a man responsible for keeping a microfiche system working, because archiving in microfiche is the “way of the future”. My only encouragement is that a digital format is generally easier to manipulate than physical media. Although a small script can convert many documents, I’ve yet to find an affordable scanner to read microfiche. Since I can still read my Usenet postings from the early 1990s, perhaps there is some hope for long-term viability.—Ed.

LJ Deserves the Benefit of the Doubt

I wasn’t planning to renew my subscription to the magazine, my instinctive reaction being against any form of digital format. But, today I got your 2012 January issue and with it your reader letters, and at least a couple mentioned how much better digital is than dead trees for the environment. Although the jury isn’t out on this one (a printed magazine nowadays should be done from renewable carbon-neutral sources, and digital media means burning fossil fuel to power all those servers and fancy gadgets), my gut feeling is digital is better in this regard. I also saw the table of contents with highly technical security topics. Finally, I have bought your magazine in the three countries where I have lived from quite early on, sometimes providing the only bit

of uncensored Western media I had access to.

In conclusion, I will give the magazine a spin in its new form, but please, keep it Linux. Sometimes you deviate so much from just Linux that it’s not funny. Now that you are charging for bytes displayed on screen, you have to focus on what the magazine is all about (or make explicit the wider aims of the magazine, but that is another can of worms I’m sure you don’t want to open at this time).

—**Jose Luis Martinez**

I’m starting to get used to the digital format myself, and although my home still might be full of paper books, I find I read more on my Sony e-reader than anything else. I suspect my kids will grow into adults who prefer to read on their mobile devices. I’m not sure how I feel about that!

As far as content goes, we try to focus on Linux and open source. We do have a “Non-Linux FOSS” blurb in our Upfront section every month, but it’s tiny. We try to choose articles that are interesting to Linux users. Usually, that means Linux-specific, but sometimes the scope is a little broader.—Ed.

Digital Format for Magazines

Although I think the format could still use some tweaking, I have been waiting for this for a couple years. I had pretty much

quit subscribing to any print publications a couple years ago because I have gotten tired of all the paper lying about. I think your short-lived sister publication *TUX* is what finally pushed me over.

—**Jim Gribbin**

TUX was published before I was on staff with Linux Journal, and I too was sad to see it go. I still have every issue lovingly preserved. I'm personally excited about the different distribution options we have electronically. The same content can be manipulated to work in multiple ways. It's pretty cool.—Ed.

YNAB, YNOT

In your January 2012 issue's Upfront section, YNAB is listed as a Linux-supported budget application. Unfortunately, a couple months ago, Adobe discontinued Linux support in AIR.

—**davbran**

Ugh, tell me about it. In fact, I mentioned YNAB for multiple reasons. One, I think it's a great program, and I figured other Linux users might like it as well. I also wanted to mention YNAB in our magazine specifically, so the company would see Linux users as a viable customer base, not just something supported because AIR happens to be cross-platform. Will YNAB see a mention in a prominent Linux magazine as a sign that it should continue Linux support? I don't know, but it's possible. Will Adobe read it and decide to

resume its Linux support? Less likely, but I guess there is some hope.

Nevertheless, YNAB works on the last-released Linux version of AIR, and since I personally use it and think it's a great program, I mentioned it. As far as AIR support goes, that really frustrates me, because I've given Adobe a lot of positive lip service during the past few years for supporting Linux. Grrrr.—Shawn

Advertising?

As I was reading the latest issue (January 2012), I couldn't help but wonder whether you are being paid for the articles such as the one on YNAB in the Upfront section. They almost come across as blatant advertising! To include YNAB is a bit of stretch—the only tie to Linux being the Android app! Don't get me wrong; it provided me with the motivation to start using it, but it would be nice to know whether you are receiving money for this advertisement.

—**Josh**

No, the journalistic side of Linux Journal generally has very little contact with the advertising side. If I do something where a company has paid, I make it very clear, saying "This XXXXX was sponsored by the fine folks at Springy Widgets dot-com", or something like that.

For the UpFront part of the magazine,

[LETTERS]

I try to find things that interest me as a Linux user and share them. Sometimes my interests line up with readers better than others, but rest assured, it's just me. (I tend to be absent-minded and forget to keep track of finances, thus my mention of YNAB.)

PS. I'm also hoping the folks at YNAB will keep their product Linux-compatible too, since Adobe dropped AIR support. Hopefully, mentioning them in a Linux magazine will show them (or Adobe) that we're customers too dag nabbit!

Switch to Electronic-Only

Regarding some comments on the recent switch to electronic-only publishing: I always liked *Linux Journal*, but I always thought it was too expensive to subscribe to. Then when *LJ* became electronic-only, it was only \$20 to subscribe, and I'm glad I did. I've been enjoying reading the archived back issues that you thankfully included in the subscription price, and the new issues seem to be maintaining the same quality. As far as print versus electronic, I have shelves full of my favorite magazines in my basement that I save because I can't bear to throw them out, thinking that "someday", say, when I retire, I'll go back and read them again, but who knows if that will ever happen. Having the issues on hand electronically makes them much more available to me on any device that can read them, which

is good. But, I'm surprised to see you still going with the two-column format like you did in print. This means when I'm reading the .pdf, I have to use the arrow keys to go up and down each page, then page down to get to the next page, then use the up and down arrows again. Since you are now electronic-only, a single-column format to eliminate the arrowing up and down would be better (lots of your readers are older like me and can't read at a resolution that would put one page on the screen). Keep up the good work, and I'm glad to be a subscriber finally.

—Frank Palmeri

Glad to have you as a subscriber, Frank ("One of Us. One of Us"). Seriously though, the two-column layout for the PDF is designed for folks who prefer the traditional magazine look. The mobile versions (.epub and .mobi) have flowing text, so the up/down-back/forth scrolling shouldn't be required.—Ed.

Electronic LJ

I bought my Xoom primarily to be able to read PDFs comfortably—including *Linux Journal* in PDF form. The new two-column format works beautifully on Xoom. Also, I get all the graphics as well as the sense that I'm reading a magazine.

I'm not a fan of .epub for anything but plain straight text (novels, for example). I've gotten several computer books from Kindle,

and on the Kindle reader, they're useless. I can get by reading them on the Xoom, but the code samples—and those are crucial, of course—have virtually unreadable layout. The same goes for trying to read poetry on a Kindle. So, please don't abandon the magazine-like PDF versions.

—Eric Beversluis

We don't have any plans to abandon the PDF, and I agree that PDFs look great on tablet computers. We do keep trying every month to improve the code layout on mobile reading devices, but it is a challenge.—Ed.

Good Work on the Digital Editions

I was a bit disappointed when you converted all subscriptions to digital, because I liked the printed edition. But after some months, I can say you guys rock. The PDFs are okay, but the native applications for iPad and Android *and* the .epub version *and* the .mobi versions are really worth the subscription.

Despite the really good work you already do, I dare suggest one more thing. Why not integrate to iOS Newsstand? I mean, don't abandon the native application for devices without



POWERPRO STORAGE SERVER

Move to The next era of Cloud Storage computing with PowerPRO

Our PowerPRO storage server transforms x86 IT infrastructure into the most efficient, shared, on-demand utility, with built-in availability, scalability, and security services for all applications and simple, proactive automated management.

Optimized with:

- Intel® Xeon® Processor X5690
- LSI 6Gb/s MegaRAID® SAS 9280 Controller



- With VMware VMFS5, you can create a 64TB datastore on a single extent. RDMs in physical compatibility mode with the size larger than 2TB can now be presented to a virtual machine.
- Storage DRS delivers resource aggregation, automated initial placement, and bottleneck avoidance to storage.
- Profile-driven storage allows you to have greater control and insight into characteristics of your storage resources.
- vStorage APIs for Storage Awareness; Storage APIs for Array Integration: Thin Provisioning
- iSCSI UI, Storage I/O Control NFS, 2TB+ LUN, Storage vMotion snapshot support.
- Swap to Host Cache, Software FCoE, Snapshot commitments and much more



Intel, the Intel logo, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries. LSI, the LSI logo, MegaRAID are trademarks or registered trademarks of LSI Corporation in the U.S. and other countries.



Yang Ming International Corp. (RackMountPro.com)

The Leading Server Builder in America. Enhancing Cloud Computing, The Optimized Technologies for Cloud

595 Yorbita Road, La Puente, CA 91744

Tel: (800) 526-8650

Fax: (626) 956-0098

sales@rackmountpro.com

[LETTERS]

iOS v.5, but it would be nice to have it integrated to the Newsstand. Thank you for the good job; I'm going to renew my subscription for sure.

—Lorenzo Lazzeri

Integrating with the various storefronts is something we're researching, but it's a complex endeavor. Hopefully, someday Linux Journal will be available seamlessly on any platform. Right now, the iOS app is the best way to get automatic delivery.—Ed.

ELF Virus, Part I—Program Bug

I read with interest Himanshu Arora's article, "ELF Virus, Part I", in the January 2012 issue of *Linux Journal*. However, I discovered a bug in his code that will cause the program to fail in certain cases. Near the end of the `infect()` routine, he uses the `rename()` function to rename the modified executable back to its original name. This call will fail if `/tmp` and `/home` are on different filesystems, as they are on my system.

My workaround was to change the `#define TMLPATE` directive to a temporary file on the same filesystem as `/home`. A more robust alternative would be to test for the potential error condition and use a `system()` call to move the file, or use `write()` to replace the original file. Thank you for the interesting article. I look forward to following this series.

—Mark Anthony

Himanshu Arora replies: *First, thanks for reading the article. Regarding the issue you pointed out, I don't think that it's a bug. It's one of those many scenarios in which the `rename()` function errors out. I have done proper error handling in the code for this.*

Your suggestion can well be addressed as an enhancement to the code in the sense that the logic won't error out if `rename()` fails. This comment could well apply to many of the other system calls used in my code. Because this code was more of a proof of concept rather than an actual virus, I refrained from adding such complexity to the code.

Anyway, I appreciate your hard work and would like to thank you for your review.

Product of the Year: GNOME 3???

I have long considered *Linux Journal* to be a reliable source of information, but declaring GNOME 3 as Product of the Year is really disappointing to me (see the Readers' Choice Awards in the December 2011 issue). I find it pretty hard to believe that this could be voted on as Product of the Year. It has been widely accepted in many, many forums that GNOME 3 is a *huge* step in the wrong direction. As a reader and longtime Linux user, I find it hard to believe that this is what voters said.

—Morocho Ni

THE OPEN WORLD SOURCE COMES TO



Keynote:
SCOTT MCNEALY
Co-founder, Sun Microsystems

COMES TO COLUMBIA, SC MARCH 28 AND 29



Other Confirmed Speakers

CHRIS ANISZCYK
Open Source Manager, Twitter

JIM JAGIELSKI
President, Apache Software Foundation

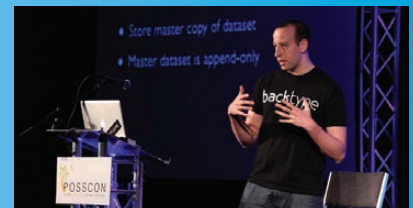
BDALE GARBEE
Chief Technologist for OS and Linux, HP

DR. DOUGLAS MAUGHAN
Director, DHS Cyber Security Division

JONATHAN LEBLANC
Technology Evangelist, PayPal

CAROL SMITH
OS Programs Manager, Google

DAVE ABRAHAMS
Founding Member, Boost.org



2012 POSSCON™

PALMETTO OPEN SOURCE SOFTWARE CONFERENCE

JOIN US FOR THE BEST SPEAKERS, TOPICS AND SOCIAL EVENTS IN THE COUNTRY

POSSCON.ORG

MARCH 28 & 29, 2012 | COLUMBIA, SC

presented by



Bringing Open Source to the Southeast

DEVELOPERS – EDUCATORS – IT LEADERS IN BUSINESS, GOVERNMENT, HEALTHCARE AND SECURITY

[LETTERS]

Readers voted; we counted the votes and reported. Perhaps those users who don't like GNOME 3 simply didn't vote. I don't know. Rest assured, however, we report what we see.—Ed.

Amazon Cloud Articles?

I am a subscriber and love this magazine. I have experimented with running Linux instances in the Amazon cloud, but I do not yet actually know enough about the process to accomplish any serious work. It would be extremely useful to me, and I am sure for others like me, to have a complete article or series of articles on spinning up Linux on AWS; creating an Amazon machine image (AMI) from scratch; converting a virtual box, Xen, or VMware virtual machine to an AMI; setting up a distributed computing task (such as data mining) on AWS; running special-purpose applications (like “Big Blue Button”) and so on. In other words, practical, in-depth treatment/tutorials on how to use the Amazon cloud for business and personal-computing solutions. I hope you will consider running such an article or set of articles.

—RAB

Thanks for your suggestions—we'll see what we can do!—Ed.

LJ Digital Format

I just purchased a Kindle to be able to

view the new LJ digital format, and so far, I think I am liking the change. I wasn't sure if this old dog could learn a new trick, but I think I have accepted the change. One request I would like to make is converting the old archives (1994–2011), which are currently available on a DVD, to the new digital format. If those were available in the .mobi file format for the Kindle, I would be first in line to place an order for such a product and probably many others would be interested as well.

—Gary Stout

Great minds think alike—we're actually working on converting some archives now. See <http://www.linuxjournal.com/ebook> for more information.

Frink Programming Language

May I suggest an article on the Frink programming language? I think it's really neat and deserves a lot of attention from the Linux community. Here are some links to whet your appetite: <http://futureboy.us/frinkdocs> and <http://confreaks.net/videos/120-elcamp2010-frink?player=flash> (a quick video presentation). It's also available as an Android install and has access to most, if not all, of the device's features (<http://futureboy.us/frinkdocs/android.html>).

—Leon

Thanks Leon, it looks cool!—Ed.

SUBSCRIPTIONS: *Linux Journal* is available in a variety of digital formats, including PDF, .epub, .mobi and an on-line digital edition, as well as apps for iOS and Android devices. Renewing your subscription, changing your e-mail address for issue delivery, paying your invoice, viewing your account details or other subscription inquiries can be done instantly on-line: <http://www.linuxjournal.com/subs>. E-mail us at subs@linuxjournal.com or reach us via postal mail at *Linux Journal*, PO Box 16476, North Hollywood, CA 91615-9911 USA. Please remember to include your complete name and address when contacting us.

ACCESSING THE DIGITAL ARCHIVE: Your monthly download notifications will have links to the various formats and to the digital archive. To access the digital archive at any time, log in at <http://www.linuxjournal.com/digital>.

LETTERS TO THE EDITOR: We welcome your letters and encourage you to submit them at <http://www.linuxjournal.com/contact> or mail them to *Linux Journal*, PO Box 980985, Houston, TX 77098 USA. Letters may be edited for space and clarity.

WRITING FOR US: We always are looking for contributed articles, tutorials and real-world stories for the magazine. An author's guide, a list of topics and due dates can be found on-line: <http://www.linuxjournal.com/author>.

FREE e-NEWSLETTERS: *Linux Journal* editors publish newsletters on both a weekly and monthly basis. Receive late-breaking news, technical tips and tricks, an inside look at upcoming issues and links to in-depth stories featured on <http://www.linuxjournal.com>. Subscribe for free today: <http://www.linuxjournal.com/enewsletters>.

ADVERTISING: *Linux Journal* is a great resource for readers and advertisers alike. Request a media kit, view our current editorial calendar and advertising due dates, or learn more about other advertising and marketing opportunities by visiting us on-line: <http://www.linuxjournal.com/advertising>. Contact us directly for further information: ads@linuxjournal.com or +1 713-344-1956 ext. 2.

Google Currents Edition?

If you guys haven't considered it already, you definitely should look into a Google Currents edition of *LJ*. Google Currents is an Android/iOS app that displays various publications and blogs. Even if you were unable to put the actual magazine into Currents (not sure how paid content works), the various reviews, blog posts and how-to articles on your Web site would be a welcome addition!

—Paul

We have started looking into Google Currents, but we do have a native Android app that will download the issue automatically every month. It pulls in articles from our Web site as well. If you haven't played with it, be sure to check it out. It's free in the Android Marketplace.—Ed.

You Won Me Over

Okay, I have to admit I was mad about the all-digital option at first. It took me a couple months before I sat down to configure my iPad to connect to my account. I can't believe how easy it was to get things going. Reading the content on my iPad has been a great experience. It's a new way of reading! I like getting the new monthly issue right away. I have to say, you won me over, and best of all, digital means *LJ* will stick around!

—Sean Humphrey

Hooray! I too enjoy seeing the full-color magazine on a tablet computer. It seems almost more vivid than the printed magazine was!—Ed.

WRITE LJ A LETTER We love hearing from our readers. Please send us your comments and feedback via <http://www.linuxjournal.com/contact>.

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

Tim Bird has started up the **Android Mainlining Project**, an attempt to create an organized flow of Android features back into the main kernel source. One of Linux's most fun qualities is that it'll compile and run unmodified on more devices than you can count. This is rarely by chance. In most cases, someone, or several people, are responsible for maintaining support on that architecture. Usually, that support is developed inside the official kernel from the start. In the case of the Android OS, **Google** forked off a copy of the kernel sources and had its teams make the extensive modifications necessary to run on Android devices. Given the high degree of "drift" that tends to come between code bases when that happens, Tim's project is pretty timely. Eventually, the entire Android OS may be reduced to a simple set of configuration variables and a lot of userspace software. Currently, it's configuration variables, userspace software and a whole lot of kernel patches.

Linus Torvalds once said that **Al Viro** had the uncanny ability to organize his kernel patches so that each one did some small thing that was obviously good. Although the great mass of his patches, taken as a whole, somehow also would manage to advance the kernel at large, advancing the broader goals of developing the **VFS** (Virtual Filesystem).

The VFS is not necessarily glamorous work, existing as it does almost entirely beneath anything the user can actually see. But, Al's work forms the underpinnings of every filesystem supported by Linux.

One little invisible thing he did lately was start cleaning up the **vfsmount data structure**, so that it wouldn't export a lot of VFS-specific data to the wider kernel world, but would take the cleaner approach of exporting only the data that non-VFS code might actually need. And, in keeping with Linus' comment of long ago, he was organizing the change so as to cause the smallest possible impact on the rest of the kernel, while

paving the way for further cleanups in the future.

What often happens with complex new technologies is that several people get different ideas about how to support them, and they start coding up those competing ideas, until one of them turns out to be the better way. **Virtualization** is one of those areas. Running other OSes under Linux used to be “virtually” impossible, or else there were frustrating partial solutions that tried to emulate a particular OS under Linux, with varying degrees of success. But, now there are things like **Xen** and **KVM**, so running another OS under Linux is trivial. As these projects continue to grow, however, their incompatibilities tend to stand out against each other. Some kernels run better under Xen than KVM, and vice versa. So, when **Stefano Stabellini** recently announced a port of Xen to the **Cortex-A15 ARM processor**, the resulting technical discussion on the mailing list tended to focus a bit on getting Xen and KVM to play nicely together.

Another area where multiple visions have been realized in the

official kernel tree is with display drivers. Between **OMAP**, **DRM** and **framebuffer projects**, no one can agree on which code base should be used to provide a more general display infrastructure that the others could be built on top of. Apparently, there’s plenty of bad blood to go around, and all the projects think that they are the most natural choice for the job.

Recently, the topic came up again when **Tomi Valkeinen**, author of the OMAP display driver, suggested using OMAP as the framework for all display drivers. Of course, the DRM and framebuffer folks thought that would be a bad idea, and a full-throated debate ensued.

Ultimately, through some mediation by **Alan Cox**, the decision seems to have been made just to focus on making all three of those systems more and more compatible with each other. This is a clever idea, because it’s hard to argue *against* greater compatibility. While at the same time, as the different implementations become more similar, it should become clearer and clearer which one truly would be the better choice to provide an underlying infrastructure for the others.—**ZACK BROWN**

Tikl Me, Elmo



Somewhere between the world of SMS messages and voice calling is the land of two-way push-to-talk technology. Some cell-phone providers have this feature as an

option for select phones, which makes your 2012-era cell phone act like a CB radio from the 1970s. Don't get me wrong, I understand there are situations when this is beneficial, but it still makes me laugh to see people using smartphones like walkie-talkies.

If you don't have the push-to-talk (PTT) feature from your cell-phone provider, you can download the free Tikl app from the Android Marketplace. Tikl allows you to use PTT technology with any other users that have Tikl installed on their phones. Because Tikl is available for both Android and iOS, it covers a wide variety of smartphones.

I don't use Tikl very often, but in my limited testing at a softball game, it worked as advertised. My daughter was able to give me her 10–20, and I was able to give her a big 10–4 on her request to play on the swings. Although using Tikl while driving probably is safer than texting, we still don't recommend it. It'd be tough to convince the Smokey that your Android smartphone is really a CB radio.—**SHAWN POWERS**

They Said It

Flying is learning how to throw yourself at the ground and miss.

—Douglas Adams

Everyone has a photographic memory, some just don't have film.

—Steven Wright

Duct tape is like the force. It has a light side, a dark side, and it holds the world together.

—Unknown (possibly Oprah Winfrey)

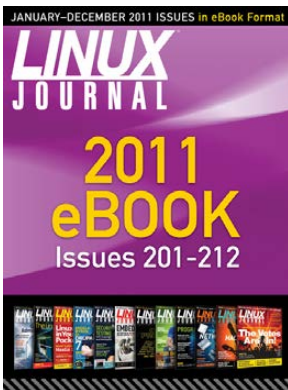
To err is human... to really foul up requires the root password.

—Unknown

Real men don't use backups, they post their stuff on a public ftp server and let the rest of the world make copies.

—Linus Torvalds

New *Linux Journal* Annual Archives for E-Readers



As this issue suggests, mobile technology is more a part of our lives than ever, and we are becoming accustomed to having a tremendous amount of information available to us at all

times, frequently on gadgets we keep in our pockets and bags. I am also a person who likes to keep a lot of documents, books and other reference materials on my phone, NOOK or tablet. You never know when you might need to look something up, right?

In order to make our articles even more available to you, we are introducing an

annual eBook of *Linux Journal* content in epub and mobi formats. I am a huge fan of these formats. In addition to the many benefits of reflowing text, I also enjoy the ability to search, highlight and take notes using my e-reader software. This new compilation of all 12 issues from 2011, is organized much like our popular Archive DVD and contains all of the articles from 2011 in one easily navigable file. This should be exciting to our readers who are fans of these e-reader formats, as it includes earlier issues that previously were not available for e-readers.

Please visit <http://www.linuxjournal.com/ebook> for all the information.

—KATHERINE DRUCKMAN

Non-Linux FOSS

If you want to record your Windows screen, but don't want to fork out the money for a commercial application like Camtasia, you might want to give CamStudio a try. CamStudio is an open-source program that captures your Windows desktop and encodes the video using an open-source video codec.

CamStudio has many features, including picture-in-picture support for folks with Webcams. If you're using Windows on one of your computers and want to try out some open-source screen capturing, give CamStudio a try. Download it at <http://www.camstudio.org> or from SourceForge at <http://www.sourceforge.net/projects/camstudio>. —SHAWN POWERS



GSL, Part II: Getting Work Done

My last article introduced the GNU Scientific Library and how to include it in your code, but I didn't really cover what you actually can do with this library. Here, I describe some of the available functionality, which hopefully will spark your interest in taking a deeper look.

A series of functions exist for handling polynomials. A polynomial is a function of different powers of a variable, with each element multiplied by a constant—for example:

$$P(x) = c[0] + c[1]*x + c[2]*x^2 + \dots$$

In the GSL, a polynomial is represented by an array containing all of the constants, with zeros for all of the missing powers. So, if your polynomial is $P(x) = 5 + x^3$, your polynomial would be represented by `c = [5, 0, 0, 1]`. Several functions are available for evaluating your polynomial at some particular value of x . And, there are separate functions for evaluating your function for real values of x (`gsl_poly_eval`), complex values of x (`gsl_poly_complex_eval`) and complex values of x with complex coefficients (`gsl_complex_poly_complex_eval`). This is because complex numbers are separate data types (`gsl_complex`) and need to be handled differently from simple doubles.

Aside from evaluating polynomials, you may want to solve the polynomial

and get the roots of your equation. The most basic example is finding the roots of a quadratic equation. These roots may be real or complex, which means there are two different functions: `gsl_poly_solve_quadratic` and `gsl_poly_complex_solve_quadratic`. You hand in the values for the three coefficients and pointers to two variables to hold the two possible roots:

```
gsl_poly_solve_quadratic(double a, double b,
    ↪double c, double *x0, double *x1)
```

If there are no real roots, x_0 and x_1 are unchanged. Otherwise, the roots are placed successively into x_0 , and then x_1 . There are equivalent functions to find the roots of a quadratic equation called `gsl_poly_solve_cubic` and `gsl_poly_solve_complex_cubic`.

Once you get beyond a quadratic equation, there is no analytical way to find the roots of a polynomial equation. The GSL provides an iterative method to find the approximate locations of the roots of a higher order polynomial. But, you need to set up some scratch memory that can be used for this purpose. For a polynomial with n coefficients, you would use `gsl_poly_complex_workspace_alloc(n)` to create this scratch space. Then, you can call `gsl_poly_complex_solve` to run this process. After you are

done, you would need to call `gsl_poly_complex_workspace_free` to free up this scratch space.

In science, lots of special functions are used in solving problems, and the GSL has support for dozens of functions. To use them, start by including `gsl_sf.h` in your source code. These functions can be called in two different ways. You can call them directly and simply get the computed value as a result. So, if you wanted to calculate the value of a Bessel function, you could use this:

```
double ans = gsl_sf_bessel_J0(x);
```

But, you will have no idea if there were any problems during this computation. To get that information, you would call a variant of this function:

```
gsl_sf_result result;
int status = gsl_sf_bessel_J0_e(x, &result);
```

The value of `status` lets you know if there were any error conditions, like overflow, underflow or loss of precision. If there were no errors, the function call returns `GSL_SUCCESS`. The result variable is actually a struct, with members `val` (the computed value of the function) and `err` (an estimate of the absolute error in `val`). All of the special functions default to evaluating with double precision, but in some cases, this is simply too costly time-wise. In order to save time in cases where a lower level of

accuracy is acceptable, the GSL special functions can accept a mode argument:

- `GSL_PREC_DOUBLE` — double precision, accuracy of 2×10^{-16} .
- `GSL_PREC_SINGLE` — single precision, accuracy of 10^{-7} .
- `GSL_PREC_APPROX` — approximate values, accuracy of 5×10^{-4} .

Some of the special functions supported by the GSL include Airy functions, Bessel functions, Debye functions, elliptic integrals, exponential functions, Fermi-Dirac functions, Legendre functions, spherical harmonics and many more. It's definitely worth taking a look at the manual before you even think about writing your own version of some function, because it's very likely already been done for you.

Vectors and matrices are used as data types in several scientific problems. The GSL has support for doing calculations with both vectors and matrices, treating them as new data types. They are both based on a data type called a block. A GSL block is a struct containing the size of the block, along with a pointer to the memory location where the block is actually stored. A vector is a struct defined as:

```
typedef struct {
    size_t size; /* number of elements in the vector */
    size_t stride; /* step size from one element to the next */
};
```

[UPFRONT]

```
double *data; /* location of the first element */
gsl_block *block; /* location of block if data
                  is stored in a block */
int owner; /* do I own this block */
} gsl_vector;
```

If `owner` equals 1, the associated block is freed when the vector is freed. Otherwise, the associated block is left alone when the vector is freed. Because of the complexity of the structure, there are special functions to handle vectors. The function `gsl_vector_alloc(n)` creates a vector of size `n`, with the data stored in the block member and the owner flag set to 1. The function `gsl_vector_free()` frees the previously created vector structure. To manipulate individual elements of your new vector, you need to use the functions `gsl_vector_get(const gsl_vector *v, size_t i)` and `gsl_vector_set(gsl_vector *v, size_t i, double x)`. If you instead want a pointer to an element, you can use `gsl_vector_ptr(gsl_vector *v, size_t i)`. Matrices are very similar, being defined as:

```
typedef struct {
    size_t size1; /* number of rows */
    size_t size2; /* number of columns */
    size_t tda; /* number of bytes for one row */
    double *data; /* location of matrix data */
    gsl_block *block; /* underlying storage block */
    int owner; /* do I own this block */
} gsl_matrix;
```

Matrices are stored in row-major order, which is the way it is done in C. Allocation and deallocation are handled by the functions `gsl_matrix_alloc()` and `gsl_matrix_free()`. Accessing elements are handled through the functions `gsl_matrix_get()` and `gsl_matrix_set()`.

Now that you have vectors and matrices, what can you do with them? The GSL has support for the BLAS library (Basic Linear Algebra Subprograms). There is a wrapped version, accessible through `gsl_blas.h`, where you can use GSL vectors and matrices in the functions. You also have access to the raw BLAS functions through the include file `gsl_cblas.h`. The GSL version treats all matrices as dense matrices, so if you want to use band-format or packed-format matrices, you need to use the raw functions. There are three levels of BLAS operations:

- Level 1: vector operations.
- Level 2: matrix-vector operations.
- Level 3: matrix-matrix operations.

BLAS has functions for things like dot products, vector sums and cross products. This provides the base for the linear algebra functions in the GSL. They are declared in the header `gsl_linalg.h` and are handled through level-1

and level-2 BLAS calls. There are functions for decomposition (LU, QR, singular value, Cholesky, tridiagonal and Hessenberg), Householder transformations and balancing. The header file `gsl_eigen.h` provides functions for calculating eigenvalues and eigenvectors of matrices. There are versions for real symmetric, real nonsymmetric, complex hermitian and real generalized nonsymmetric eigensystems, among others.

The last thing to look at is the functionality supporting calculus calculations. A whole group of functions handles numerical integration, and there are routines for both adaptive and non-adaptive integration for general functions. There also are specialized versions for special cases like infinite ranges, singular integrals and oscillatory integrals. The types of errors that may happen when you are trying to do a numerical integration are:

- `GSL_EMAXITER` — the maximum number of subdivisions was exceeded.
- `GSL_EROUND` — cannot reach tolerance because of round-off error.
- `GSL_ESING` — a non-integrable singularity or bad integrand behavior.
- `GSL_EDIVERGE` — integral is divergent or doesn't converge

quickly enough.

Numerical differentiation also can be done, using finite differencing. These functions are adaptive, trying to find the most accurate result. The three versions are:

- `gsl_deriv_central()` — central difference algorithm.
- `gsl_deriv_forward()` — adaptive forward difference algorithm.
- `gsl_deriv_backward()` — adaptive backward difference algorithm.

In all of these, you hand in a pointer to a function, the value of x where you want to calculate the derivative and a step-size, h , for the algorithm. You also hand in pointers to variables to store the values of the result and the absolute error.

I have barely scratched the surface of what is available in the GSL. Hopefully, you now have a better idea of some of the functions available. Although lots of scientific packages are available, sometimes there really is no option except writing your own. With the GSL, you should be able to do this with a bit less work and get to the actual computational science more quickly.

—**JOEY BERNARD**

Cryptocurrency: Your Total Cost Is 01001010010



Most people have heard of gold. Most people are familiar with dollars. For a handful of geeky folks, however, the currency they

hope will become a global standard is digital. Whether it's a problem or not, the currency you use on a day-to-day basis is tied to the government. The global value of the money in your pocket can vary widely, and if you're a conspiracy theorist, your concern might be that it could be worth nothing in the blink of an eye.

Surely gold and silver are wise investments if you're concerned your paper dollars will drop in value, but using gold as a means to buy a gallon of milk is a bit difficult. Perhaps cryptocurrencies are the solution. The most popular form of cryptocurrency is the Bitcoin. A very simple explanation of how it works is as follows:

- Users download the bitcoin client

and use their computer to solve complex math problems, which create a cryptographic record of any transactions on the Bitcoin P2P network.

- Users are rewarded Bitcoins for successfully "hashing" the cryptographic record of transactions, and that reward system is how Bitcoins are created.
- Users then securely transfer Bitcoins back and forth to purchase items, and those transactions are recorded in the cryptographic history for the entire P2P network to see.

The process is, of course, a little more complicated than that, but that's basically how it works. The computers creating the cryptographic history of transactions are called miners, and "Bitcoin Mining" is simply the act of solving complex math problems in order to make a cryptographic record of transactions. Because mining Bitcoins is how the currency is created, lots of people are connected to the network, racing

each other to get a solution that will generate a reward. In fact, it's so competitive, that unless you have a high-end GPU that can process the equations extremely fast, there is no point in trying for the rewards.

Are Bitcoins the future of global currencies? Will one of the alternative cryptocurrencies like Litecoin or Solidcoin become commonplace? The number of places that accept cryptocurrencies are extremely limited, so it's not any easier to buy a gallon of milk with a Bitcoin than it is with a lump of

gold, but many think that day is coming. What about you? Do you think cryptocurrency has a future, or do you think it's a geeky fad that will fade away? Send an e-mail with "CRYPTOCURRENCY" in the subject line to shawn@linuxjournal.com, and I'll follow up with a Web article based on your feedback. For more information on cryptocurrencies, check out these Web sites:

<http://www.bitcoin.org>,
<http://www.litecoin.org> and
<http://www.solidcoin.org>.

—**SHAWN POWERS**

Powerful: Rhino



Rhino M6500/E6510

- Dell Precision M6500 w/ Core i7 Quad (8 core)
- Dell Latitude E6510 w/ 2.53-2.8 GHz Core i5/i7
- Up to 17" WUXGA LCD w/ X@1920x1200
- NVidia Quadro FX 3800M
- 250-750 GB hard drive
- Up to 32 GB RAM (1333 MHz)
- DVD±RW or Blu-ray
- 802.11a/b/g/n
- Starts at \$1385

- High performance NVidia 3-D on a WUXGA RGB/LED
- High performance Core i7 Quad CPUs, 32 GB RAM
- Ultimate configurability — choose your laptop's features
- One year Linux tech support — phone and email
- Three year manufacturer's on-site warranty
- Choice of pre-installed Linux distribution:



Tablet: Raven



Raven X201 Tablet

- ThinkPad X201 tablet by Lenovo
- 12.1" WXGA w/ X@1280x800
- 2.0-2.13 GHz Core i7
- Up to 8 GB RAM
- 250-500 GB hard drive / 160 GB SSD
- Pen/stylus input to screen
- Dynamic screen rotation
- Starts at \$1940

Rugged: Tarantula



Tarantula CF-31

- Panasonic Toughbook CF-31
- Fully rugged MIL-SPEC-810G tested: drops, dust, moisture & more
- 13.1" XGA TouchScreen
- 2.4-2.53 GHz Core i5
- Up to 8 GB RAM
- 160-750 GB hard drive / 256 GB SSD
- Call for quote

EmperorLinux

...where Linux & laptops converge

www.EmperorLinux.com

1-888-651-6686



Calibre in the Cloud

I've mentioned before that I keep my entire e-book collection in my Dropbox folder, and I can access it anywhere I have a Web connection. I didn't come up with the idea myself; instead, I shamelessly stole the idea from Bill Childers. I suspect he stole it from someone else, so feel free to steal the idea from me.

Basically, it involves two programs, both free (well, three, if you count Dropbox). First, create a folder inside your Public folder that resides in your Dropbox

folder. You can name this folder anything you like, but because it will be hosting all your e-books, it's wise to name it something no one would guess.

Then, in Calibre, click on the bookshelf icon (upper left), and click "switch library". Then, select that new secret folder you made inside your

Public Dropbox folder. Calibre will move your entire library to that folder, so make sure you have enough free space in your Dropbox to handle your entire e-book collection. If you have too many e-books, you could create a separate library inside Calibre and just keep a select few books in that Public folder.

Now you should have a working install of Calibre that stores your e-books and database inside your Dropbox. You simply can open this library file with Calibre on

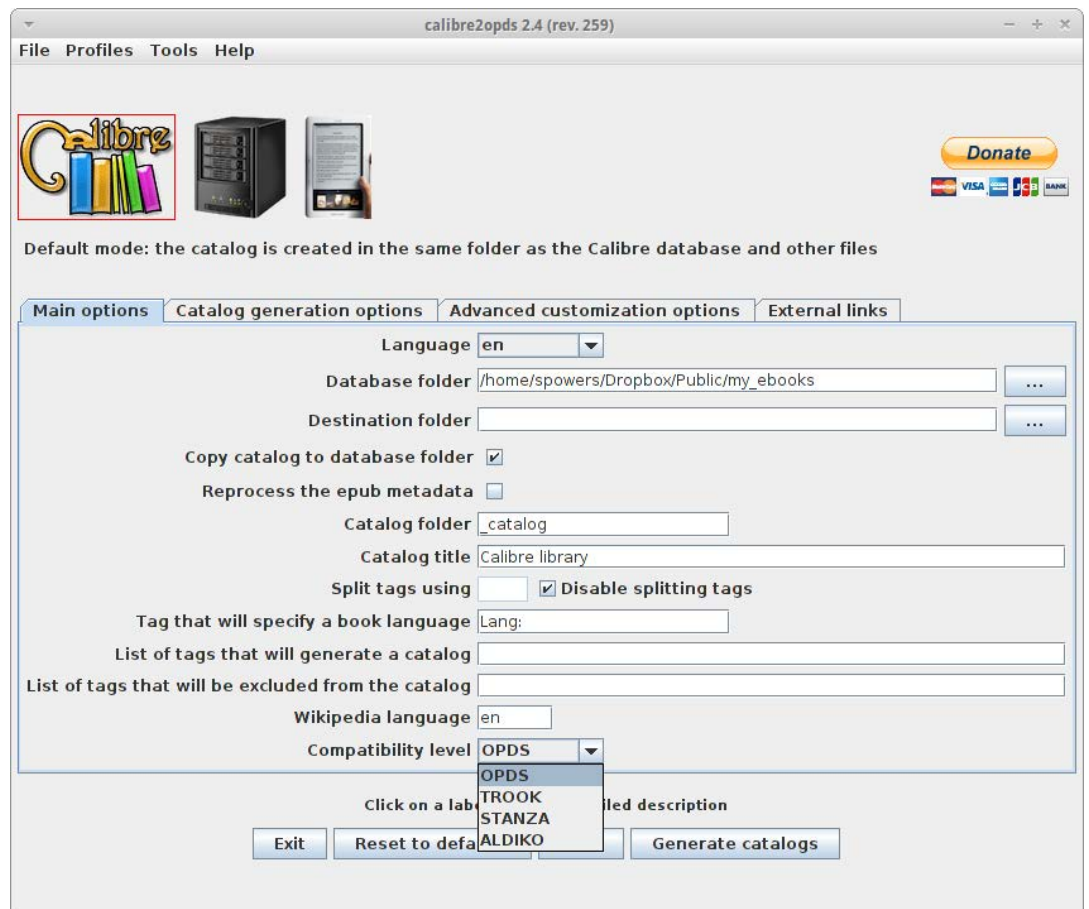


Figure 1. calibre2opds is a GUI Java application.

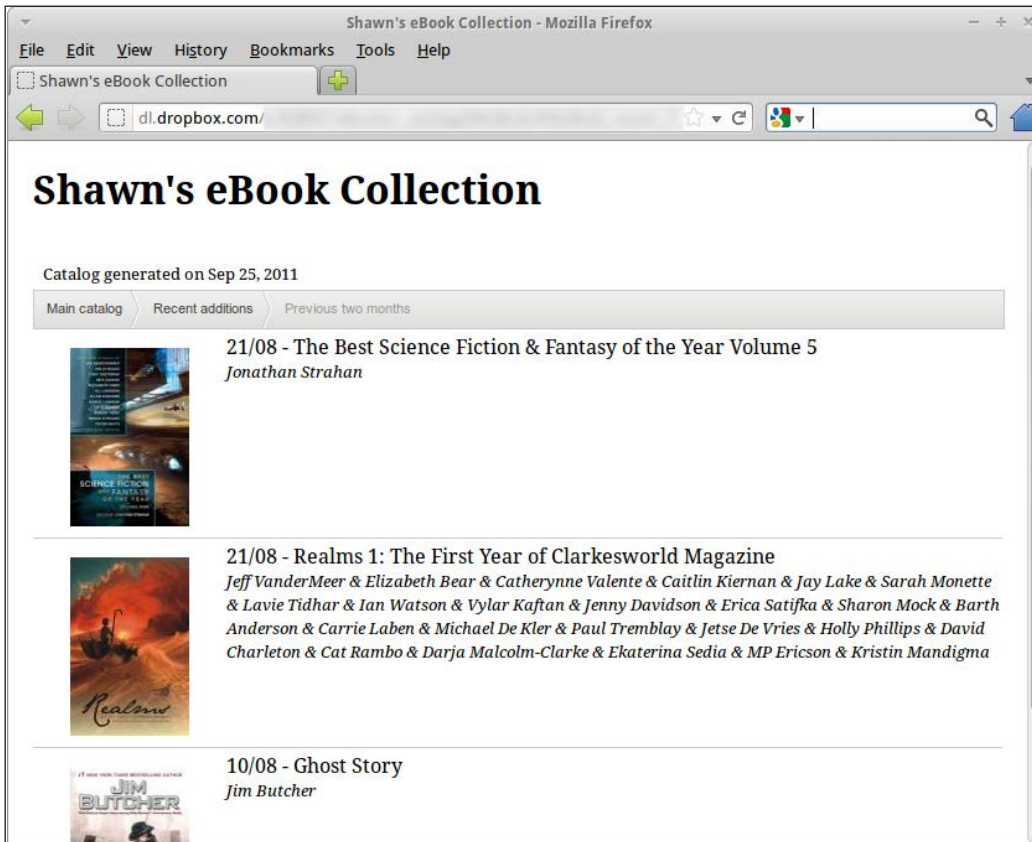


Figure 2. Here is my e-book collection, created by calibre2opds.

other computers that sync with Dropbox, or you can go one step further and create a cloud-based repository that you can browse from any computer. And, that's where calibre2opds comes into play.

calibre2opds is a Java application that creates a Web site from your Calibre library. Download the Java application from <http://www.calibre2opds.com>, and launch it with your favorite Java runtime environment. Once it's launched, you'll see *many* options for tweaking how your library will look. The first thing you need to do is make sure the Database Folder is pointed to the secret folder to which you moved your Calibre library. Then, you'll want to set the

Catalog Folder to something. It's okay to leave it set to `_catalog`, which is the default.

Next, you need to decide what sort of Web site you want to create. If you want to be able to browse it with any Web browser, leave the "Compatibility Level" at "OPDS". If you want to browse directly with your Android device, you can

choose either TROOK or ALDIKO, and calibre2opds will generate a catalog that those readers can access directly. Once you tweak any other settings to your liking, click the Generate Catalogs button on the bottom, and it will create all the files you need right inside your Calibre database folder.

Because you did all this inside your Public Dropbox folder, you can look for that `_catalog` folder and find the `index.html` file inside it. Right-click on `index.html`, get the Dropbox public link for it, and see the result. (Note: you may need to find the `index.xml` file if you're trying to browse with Aldiko or Trook.)

—SHAWN POWERS



REUVEN M.
LERNER

Logging

Knowing how to read logs and write to them are important skills for all Web developers.

When you're writing programs that operate on the command line, the error messages either appear in the same place as you're typing (that is, in the same terminal window) or are sent to a back-end logging system, such as the venerable "syslog" facility that Linux provides. But, one of the surprises and frustrations that Web developers encounter is that there is no such thing as a "terminal" in which their Web application is running. Rather, the Web server sends its error messages somewhere else entirely, to one or more files.

I often tell participants in my Web development courses that they should embrace their new best friend, the logfile. And, indeed, logfiles are a priceless tool in the hands of a developer, making it possible—like a doctor—to understand the source of a problem and then diagnose and fix it. Because there is no way to interact directly with a Web server, looking at the effects as displayed in the logfile is the best way to understand what is happening. Through the years, I've learned, however, that

developers often are unfamiliar with many of the issues having to do with logging and how they can make the best use of those logs to ensure that their programs run as smoothly as possible.

So in this article, I look at a number of issues having to do with logging, as well as some strategies and techniques you can use to make best use of the logs. Some of the things I mention here are specific to the Ruby on Rails framework, which I use in much of my day-to-day work, but several techniques are common to many packages or to programming in general.

Where Are You Logging?

A typical modern Web application consists of a database server, an HTTP server and an application. For example, I generally use PostgreSQL (database), Apache (HTTP) and Ruby on Rails (application), but it's not unusual for me to use other technologies, from MySQL and MongoDB to other languages, servers and systems. Each program has its own logfile, with its own configuration settings and output syntax.

The first thing you should do when working on a project, whether you are starting it or taking it over from someone else, is determine which logfiles are being written to. In a Web application that uses Apache, Rails and PostgreSQL, you'll have at least three different logfiles, one for each of these programs.

Sometimes these logfiles are placed under the `/var/log` directory in your filesystem, but not always. It's not unusual for a PostgreSQL logfile to be in the "data directory", which on my systems often is in `/usr/local/pgsql/data`. Note that the default location for a program's logfiles might not match the location that your Linux distribution has adopted, so unless you're careful, you can end up with two separate logfile directories, one of which is ignored by the system.

Some programs can create more than one logfile. Apache is able to act as an HTTP server for multiple sites, and it's often helpful to be able to separate out logs for those sites. And, Apache also separates HTTP request logs from errors and referrers, meaning that if you want to understand fully what is happening on your site, you might need to piece together several logs just for the HTTP server.

Apache Logs

The two most important logs that Apache produces are the request log and the error log. A typical line in the request

log (using the "common format", which combines referrer information with requests) represents a single HTTP transaction and looks like this:

```
84.108.219.125 - - [09/Jan/2012:14:13:52 +0200]
  "GET / HTTP/1.1" 200 764 "http://linuxjournal.com/"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/535.11
  (KHTML, like Gecko) Chrome/17.0.963.26 Safari/535.11"
```

The most important elements of this entry are as follows:

- The request was made from IP address 84.108.219.125.
- The request took place at 14:13:52, UTC + 2.
- The GET method was used to request `/`, using HTTP 1.1.
- The return code was 200.
- The response contained 764 bytes.
- The user came to this page from a link at LinuxJournal.com.
- The user was browsing with the Safari browser, on an Intel-based Mac.

The access log can be useful for a number of reasons. First, I'm often trying to debug the HTTP requests that a program makes to a server. This can be as mundane as when I click on a link in my

The test logfile generally is full of enough detail that if something goes wrong, you can reproduce it in the console and understand why it didn't work.

browser or when I'm writing a program that is trying to retrieve information from a server. The HTTP access log shows me precisely what URL was sent to the server; I frequently have been able to debug problems with the requesting program as a result.

Of course, only GET requests, in which name-value pairs are part of the URL, are debugged easily in the access log. POST requests send their name-value pairs on a separate channel, which means you can't rely on the server log to display them.

It's also easy to assemble basic statistics about a site by looking at the Apache logs. For example, from my server, the response code is column #9 of the logfile. By chaining together several command-line tools, I can find out what percentage of each request got each response:

```
cut --fields=9 --delimiter=' ' access.log | sort | uniq -c
```

First, I use `cut` to divide the logfile into fields, separated by spaces. Next, I grab the 9th column, which corresponds to the HTTP response code. I then sort those codes and use `uniq` to count the number of times each one appears.

The other important Apache logfile, and one that is crucial in debugging

Web applications, is the error log. Not only does this file mention many errors and warnings that Apache has encountered, but also anything that the Web application might have sent to the `STDERR` filehandle while executing. If something goes wrong on a Web application that I'm developing, my first instinct is to look at the error log. Often messages there will give me an indication of what is happening, including warnings of errors, incorrect HTTP headers or truncated requests.

Application Logs

For many years, the Apache error log was the best tool I had for debugging Web applications. Modern Web frameworks operate outside Apache (or any other HTTP server) though and often have their own logfiles. Ruby on Rails, for example, has a separate logfile for each of its environments. Thus, under the `logs` directory, you typically will find `development.log`, `production.log` and `test.log`.

The test environment is where you run your tests, and having a full-fledged logfile that I can open and peruse after tests have failed makes it relatively easy to figure out why they didn't work. The test logfile generally is full of enough

detail that if something goes wrong, you can reproduce it in the console and understand why it didn't work.

The development log also is full of useful information. Although it reflects whatever requests you make to the development server, it offers a chance for more insight. For example, the default setting in Ruby on Rails gives a huge amount of information about database access via ActiveRecord, including the SQL that ultimately is sent to the server and when objects are fetched from ActiveRecord's cache, rather than from the database itself. When I'm developing a Web app, it's very common for me to do a `tail -f` (more on that below) on the Web application log.

The production log, as its name implies, is written to the production server. On systems with more than one production server, it can become tricky to keep track of multiple logs, because you cannot know to which production server given users will be sending their HTTP requests. In such instances, using a system that allows you to aggregate logs in a single place, such as syslog or Graylog2 (which I haven't ever used, but it looks like a marvelous system), might well come in handy. When I have a small number of servers, I often take the easy way out, putting the production logs separately on each server and then just looking at both via `tail -f` to see where a problem might have cropped up.

Database Logs

Most modern Web applications use a database of some sort. These often are relational databases, such as MySQL and PostgreSQL. Each of these databases has a variety of switches and configuration variables that you can set to describe and customize the ways in which logfiles are written.

I tend to use PostgreSQL whenever possible and generally set up the logs the same way on all systems I run. I turn logging on and configure the logs to rotate once per day, such that I have the last seven days' worth of logs if I ever need to check on something.

One of the nicest things about PostgreSQL's logs is that you can decide what you want to log—connections, disconnections, query parse trees, statement durations, vacuum execution or even the output from the query optimizer. These settings are in the `postgresql.conf` configuration file, which normally sits in the data directory, alongside the security configuration file `pg_hba.conf`.

By setting the appropriate variables in `postgresql.conf`, you can set separate log levels for client connections (`client_min_messages`) and the logfile (`log_min_messages`), such that the logfile contains more detail than messages sent to database clients. You can ask for log messages to be displayed verbosely (`log_error_verbosity`), which gives a level of detail I've never

found useful, but which undoubtedly comes in handy when doing low-level database hacking.

The `log_statement` parameter allows you to log DDL (database definition statements), `mod` (modifications to the database) or everything.

One of the most useful log-related settings in the PostgreSQL configuration file is `log_min_duration_statement`. By default, it is set to `-1`, meaning that it is deactivated. Given a positive number as a parameter, this will log any time a statement on your PostgreSQL server takes longer than a certain number of milliseconds. I normally set this to 100ms and then can go through the logfile (using `grep`, searching for the word “duration”) to find which queries are taking the longest. That allows me to concentrate my efforts on the slowest queries, as identified by the database server itself.

How to Read Logs

One of the best ways to read through logfiles is also the easiest, namely the `less` command. `less`, which is a modern replacement for the traditional UNIX `more` command, lets you page through a file, backward and forward—and if the file grows while you’re viewing it, `less` still will let you read through it, including to the current end of the file.

`less` has a very large number of options that you can apply. For years,

my `.zshrc` profile has defined the `PAGER` environment variable to be:

```
less -RXs
```

This means it shows raw control characters (useful for reading colored logs and man pages), that it doesn’t reset the terminal (which I admit I use out of habit, rather than having demonstrated with certainty that this option is necessary), and that multiple blank lines will be squeezed into a second one. The fact that searching in `less` not only supports regular expressions, but also highlights any text that matches my search, makes it fairly easy to find my way around.

However, I often want to read the logs as they come through. One easy way to do this is with `tail -f`. The `tail` command shows you the final lines of a file on your filesystem. The `-f` flag tells Linux to keep reading from the file as it grows and to print it on the screen. When I’m developing, I often have one terminal window open on the application’s logfile with `tail -f`:

```
tail -f log/production.log
```

If you’re on a heavily loaded server, the output from `tail -f` not only will slow down your machine, but also will be unreadable by you. The solution is to filter the output of `tail -f` through

`grep`, looking for a particular piece of text that is of interest to you, such as the name of a variable, function or error message.

If you want to grab a bunch of lines, you can use `grep`'s `-A` (after), `-B` (before) or `-C` (context = before + after) flags. For example, if you want to see the list of parameters sent to your Web application server every time the `Home#index` method is called, you can do this:

```
tail -f log/production.log | grep -A10 Home#index
```

Do you really need ten lines after the method name is printed in the logs? Probably not, but this ensures that you'll get all of the parameters, without having to read through lots of other stuff.

Writing to Logs

It might sound odd, but it takes some practice to learn how to write to logfiles as well. I tend to write a lot of information to logs when I'm developing a Web application. I realize there are people for whom a debugger is their first tool of choice, but mine is the console that Rails provides, followed closely behind by the logfiles. From a Rails application, you can write to the log at any point using the "logger" object, which is defined in all models and controllers. The logger object supports a method for

each log level, from "crit" (critical) to "info", and everywhere in between. So to write "hello" into the logs, you would write:

```
logger.warn "hello"
```

Now, whatever you write to the logs is printed verbatim. You can, of course, write the value of a variable:

```
logger.warn some_variable
```

But, this is almost always a bad way to go. First, you want to be able to find what you've written to the logfile. Second, you're probably going to be writing multiple variable values, so you will want to indicate what you're printing here to distinguish it from other places. And finally, if the variable value is blank, or a newline, you'll want to know that—most easily by putting its value between delimiters. So when I want to print a variable value, I do something like this:

```
logger.warn "[ObjectName#method_name] some_variable  
↳ '#{some_variable}'"
```

Now I easily can find this value in the logfile; I can know in which method I was printing things, and I also can distinguish between different values—an empty string, a space or a newline.

If it's a complicated variable, I sometimes use the built-in `to_json` or `to_yaml`

methods that Rails provides:

```
logger.warn "[ObjectName#method_name] some_variable =
↳ '#{some_variable.to_yaml}'"
```

But watch out! I've learned the hard way that certain objects cannot be printed to YAML. What happens in those cases is that you get an error message instead of a printout, and that can cause more head-scratching, trying to figure out how and where you introduced a new bug. Printing things simply, in non-YAML format, is generally a good idea, when you can get away with it.

Logging Every Action

On some projects, I go all-out, logging every HTTP request to the database. This is a tremendous help when debugging applications, although it does tend to have the side effect of slowing things down, because each request needs to write to the database. (Using a non-relational database probably would be a faster way to go about this, although I enjoy using the powerful queries that SQL provides when I need to pull information out.) When I do this, I typically create a `logged_actions` table in the database, containing the following columns:

- `user_id`.
- controller name.
- action name.

- `logged_at` (timestamp).
- `message` (allows me to stick an arbitrary message into these logs).
- `ip_address` (of the person requesting).
- `browser_info` (from the user's browser).
- URL.
- `params` (all of the parameters sent, via GET and POST).
- `session` (the user's entire session, written in YAML if possible).
- `cookies` (the user's cookies, written in YAML if possible).
- `flash` (contents of the Rails "flash", a sort of temporary session).
- `referrer`.

I then create a "before filter" in the application controller, such that every single request to the Web application results in an entry in the `Logged_Actions` table. Moreover, whenever I want to add some debugging or other information, I put it into the logs in the usual way, or I can write it with much more information and context via my `Logged_Actions` table. This table, and the entries in it, have proved to be extremely valuable in many cases, allowing me to debug

problems that otherwise would have been difficult to revisit or understand.

When I'm not debugging, I often turn this feature off, given the combination of overhead that it causes and the potential privacy/security issues that it can sometimes raise. In some cases, I've configured this before filter based on a configuration parameter that I set elsewhere in the system, such that I can turn fine-grained logging on and off without having to modify and re-deploy the application.

Conclusion

If you are a Web developer, and if you aren't yet using logfiles as part of your development and debugging process, I strongly suggest you begin to do so. Logfiles are a gold mine of information in all cases, but when you start to track and write additional messages, you give yourself the ability to discover and fix problems even before your users are aware of them happening. Plus, if you keep track of users' parameters and request information, you can often discover, as I have, that sometimes the problem has to do with something in users' sessions or histories, or even the browsers they are using to visit your site. ■

Reuven M. Lerner is a longtime Web developer, architect and trainer. He is a PhD candidate in learning sciences at Northwestern University, researching the design and analysis of collaborative on-line communities. Reuven lives with his wife and three children in Modi'in, Israel.

Resources

Logging is a broad topic, but one of the best treatments and discussions I've recently encountered was on the *Ruby Rogues* podcast. Episode 25 was about logging and had an in-depth discussion of what to log, how to log and various tools that have to do with logging.

Apache has a very flexible logging system. If you are running the latest (2.2) version of Apache HTTPD, read the documentation at <http://httpd.apache.org/docs/2.2/logs.html>.

The Graylog2 Project, although I haven't used it myself, looks like a great way to both store large volumes of logging information, and then produce reports and analysis based on it. It's worth looking into, at <http://graylog2.org>.

A highly sophisticated and mature logging system for Java, known as Log4j, is run by the Apache Software Foundation at <http://logging.apache.org/log4j>. There are versions of log4j in many other languages, including Python, Ruby and Perl, giving you the ability to write different types of levels of messages to different destinations.

Finally, you can read more about PostgreSQL's logging capabilities in the documentation. Look for "Error Reporting and Logging" in the manual, which is always available at <http://postgresql.org>.



DAVE TAYLOR

A Word Finder for *Words With Friends*

Dave continues to explore how to create a *Scrabble* suggestion engine and learns that regular expressions sometimes aren't the best path to a solution.

In my last article, I looked at a simple crossword-puzzle word finder, something that requires a word list and a basic understanding of grep. Then, I switched to looking at *Scrabble* and the popular on-line equivalent *Words With Friends*.

In this latter case, the problem turns out to be quite a bit more difficult. Say you have seven tiles (one or more of which could be a blank or wild-card tile, but I'll address that later) that are a random set of letters, and from that, you want to combine them to create dictionary words. For example, I'm playing a game of *Words With Friends* with my sister, and the tiles I have to work with are T E C Y Z S X. But, of course, any good *Scrabble* player knows that you also need to work with the letters already on the board, so although I can make a word like "YET" or "SEX"

from these letters, I still have to interlock the word onto the board somehow. It's harder, but that's where big-point word play comes from.

Still, let's stick with the basics: enter a set of letters, and the script will offer a set of possible words using those letters. What about all these nuances? Yeah, they're going to make this way more complicated!

Words from Your Letters

Tapping the word list we downloaded in the last column, the most obvious search is for the letters as a regular expression:

```
$ grep '^t*e*c*y*z*s*x$' words.txt
ex
```

Ah, that doesn't work well because the order of the letters is important to grep although it's not important to us.

Instead, a complicated pipe offers an interesting alternative:

```
grep t words.txt | grep e | grep c | grep y
➔| grep z | grep s | grep x
```

But, that doesn't produce any results because it's looking for words that have all the letters in our hand, and that's basically impossible.

So, what about this:

```
grep 't*' words.txt | grep 'e*' | grep 'c*' | grep 'y*'
➔| grep 'z*' | grep 's*' | grep 'x*'
```

The `x*` notation is “zero or more occurrences of letter `x`”, and that is clearly not going to work because every word matches this complex search pattern if you think about it.

Let's flip this around and screen out all words that contain letters not in our set of letters instead:

```
$ grep -vE '[^tecyzsx]' words.txt
cee
cees
cess
```

There's another problem. The words match, except we're not taking into account the frequency of each letter. That is, although “cess” indeed comprises only letters in our set, we have one “s”, not two, so it's not actually a valid match.

Then again, at least it's a step in the right direction, which is more than the previous examples have demonstrated, so let's run with it.

Analyzing Length and Filtering Results

With seven letters, the first simple secondary filter is that any words longer than seven letters can be axed. How to test? The `wc` command works, but awkwardly. Still, I have a sense we're going to end up with each possible match going into a more complex test, so let's start building it:

```
#!/bin/sh
# Findword -- find matching dictionary words for
# Scrabble given a set of letters
datafile="words.txt"
maxlength=7
minlength=4
if [ -z "$1" ] ; then
    echo "Usage: $(basename $0) letters"; exit 1
fi
for possibility in $(grep -vE "[^$1]" words.txt)
do
    length=$(echo $possibility | wc -c)
    if [ $length -gt $maxlength ] ; then
        echo "$possibility is too long"
    elif [ $length -lt $minlength ] ; then
        echo "$possibility is too short"
    else
        echo $possibility is a possibility -- length = $length
    fi
done
```

You might find it faster simply to add one to each of the default settings, but because down the road, I am anticipating letting the user specify min/max length words, the compensatory code seems a better solution.

There's a built-in problem with this script that you'll realize if you're familiar with how `wc` counts letters. To demonstrate:

```
$ echo linux | wc -c
    6
```

Six? Shouldn't that be five?

The fix is to add the following:

```
# adjust lengths because our fast wc-c adds 1 char
maxLength=$(( $maxLength + 1 ))
minLength=$(( $minLength + 1 ))
```

You might find it faster simply to add one to each of the default settings, but because down the road, I am anticipating letting the user specify min/max length words, the compensatory code seems a better solution.

With that added code, we can find five-, six- or seven-letter words that are made up of only the letters in our hand by simply commenting out the "too long/too short" message:

```
$ findword.sh tecyzsx
cees is a possibility -- length = 5
cess is a possibility -- length = 5
cesse is a possibility -- length = 6
cesses is a possibility -- length = 7
cete is a possibility -- length = 5
cetes is a possibility -- length = 6
cyeses is a possibility -- length = 7
```

Now, we can't sidestep any longer; it's time to figure out how to test for the frequency of each letter to ensure that the words actually can be formed by the tiles we hold. Note that in the above example, none of the above words are actually a match when letter frequency is taken into account.

Counting Letter Occurrences

The first piece of this puzzle is to figure out how many times a letter occurs in a given word or sequence. Although there probably is a regular expression to do just that, I settled on the `-o` flag to `grep`, as demonstrated:

```
$ echo test | grep -o t
```

```
t
t
echo occurrences "t" "test"
occurrences "t" "test"
```

Append a `wc -l`, and you can write a simple function that returns the number of times a specified letter occurs in a given word or sequence:

```
occurrences()
{
    # how many times does 'letter' occur in 'word'?
    local count=$( echo $2 | grep -o $1 | wc -l )
    echo $count
}
```

Testing will demonstrate that the result is "2", as hoped for. This'll be the starting point for us in my next article when we continue this epic scripting journey into the world of *Scrabble*, *Words With Friends* and other word games. ■

Dave Taylor has been hacking shell scripts for more than 30 years. Really. He's the author of the popular *Wicked Cool Shell Scripts* and can be found on Twitter as @DaveTaylor and more generally at <http://www.DaveTaylorOnline.com>.

LINUX JOURNAL

now available
for the **iPad** and
iPhone at the
App Store.



Available on the
App Store

linuxjournal.com/ios



For more information about advertising opportunities within *Linux Journal* iPhone, iPad and Android apps, contact Rebecca Cassity at +1-713-344-1956 x2 or ads@linuxjournal.com.



KYLE RANKIN

Password Cracking with GPUs, Part III: Tune Your Attack

You've built the hardware, installed the software and cracked some passwords. Now find out how to fine-tune your attacks.

In the first two parts of this series, I explained what hardware to get and then described how to use the hashcat software suite to perform dictionary and brute-force attacks. If you have been following along, by this point, you should have had plenty of time to build your own password-cracking hardware and experiment with oclhashcat. As I mentioned in my last column, password cracking is a pretty dense subject. In this article, I finish the series by describing how to tune and refine your attacks further so they can be more effective.

Use More GPU Cycles

The first area where you can fine-tune your attacks is to put more or less load on your GPU. The `-n` option,

when passed to oclhashcat, changes how much of your GPU will be used for an attack. The documentation says that this value is set to 80 by default; however, on my computer, it seemed like the default was set closer to 40. When I first ran a brute-force attack, the output told me I was using around 70–80% of my GPU. Once I added `-n 80` to my oclhashcat command, I noticed I was using between 96–98% of my GPU and had added an extra 40,000 comparisons per second:

```
/path/to/mp32.bin -1 ?d?l?u ?1?1?1?1?1?1 | \  
/path/to/oclHashcat-plus32.bin -m 400 -n 80 \  
-o recovered_hashes phpass-hashes
```

Experiment with passing different

values to `-n`, and see whether your comparisons per second and the percentage of GPU used increases. Be careful though; the higher the number, the more power your GPU is going to use (and if it's not well-cooled, the hotter it will run). Also, if you plan to use the system for other things while you crack passwords, you may notice a greater impact on graphics performance.

Although it may seem like increasing the `-n` setting is a no-brainer, it turns out that a higher setting really only benefits brute-force attacks. The hashcat documentation recommends you try lower `-n` values when attempting dictionary attacks. Ultimately, the key is to experiment with both high and low values and see what gives you the best results.

Mask Attacks

In Part II of this series, I described two types of attacks: a dictionary attack and a brute-force attack. With a dictionary attack, you provide the cracking software with a dictionary full of possible passwords to try, such as all of the words in the English dictionary. A brute-force attack iterates through all possible combinations for a password of a certain length. Because a dictionary attack generally has way fewer passwords to try, it is

much faster than a brute-force attack. Although a brute-force attack takes a long time, it also ultimately will find the passwords you are looking for.

It turns out you aren't limited by either a fast, possibly ineffective, attack or a highly effective, but slow, attack. With mask attacks, you can combine the speed of dictionary attacks with some of the thoroughness of a brute-force attack. Mask attacks work by making some educated guesses about the characters that might be used in a password. With a mask attack, you perform a brute-force attack only with a far smaller list of combinations to try all based on a pattern.

Mask attacks make more sense once you see an example. Let's say that you are attempting to crack a password, and you know the password policy requires the user to select at least one uppercase letter and at least one number. As I mentioned in my previous article, you can calculate how many combinations are in a particular type of password by taking the number of characters in the character set, figuring out how long the password is going to be, then raising the first number to the power of the second. So, for instance, if you wanted to do a thorough brute-force attack against the above password policy, you would have 62 characters in your character

set (A–Za–z0–9) and with an eight-character password, the number of combinations would be: $62^8 = 218$ trillion combinations.

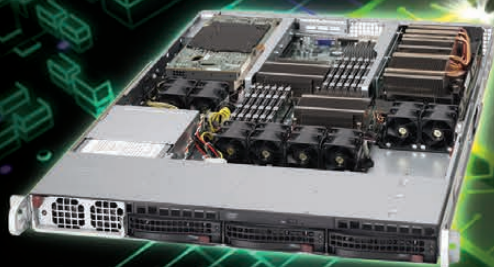
At 350,000 comparisons per second on my password-cracking hardware, it would take me approximately 7,200 days, or 19 years, to complete the attack.

The fact of the matter is, when you tell most users to create an eight-character password that has at least one uppercase character and at least one number, most users aren't going to generate a truly random password. Instead, they likely will make the first letter uppercase and then use lowercase

characters until they get to the end of the password, where they either will add a single number to the end of the password or they will put a four-digit year at the end—usually the year they were born, the year they graduated high school or the current year. A mask attack against the same password policy would build a brute-force pattern where you would just try an uppercase letter as the first character, lowercase for the next three, then either lowercase or numbers for the final four characters. In that case, the number of combinations would be: $(26) * (26^3) * (36^4) = \sim 767$ billion combinations.

Try Before You Buy!

Benchmark Your Code on Our GPU Cluster with AMBER, NAMD, or Custom CUDA Codes



NEW Microway MD SimCluster with 8 Tesla M2090 GPUs, 8 CPUs and InfiniBand 30% Improvement Over Previous Teslas

Configure your WhisperStation or Cluster today!
www.microway.com/tesla or 508-746-7341



GSA Schedule
 Contract Number:
 GS-35F-0431N

On my hardware, that would take a bit more than 600 hours, or 25 days. Although that's a long time to crack a password, it's still a lot better than 19 years and likely will be effective against a large number of weaker passwords.

To describe this pattern, I use the same custom pattern language with maskprocessor that I used in the previous column for regular brute-force attacks, only in this case, I combine a custom pattern that includes all lowercase characters and numbers with a regular set of character patterns. The final maskprocessor command would look like:

```
/path/to/mp32.bin -1 ?d?l ?u?l?l?l?l?l?l?l?l
```

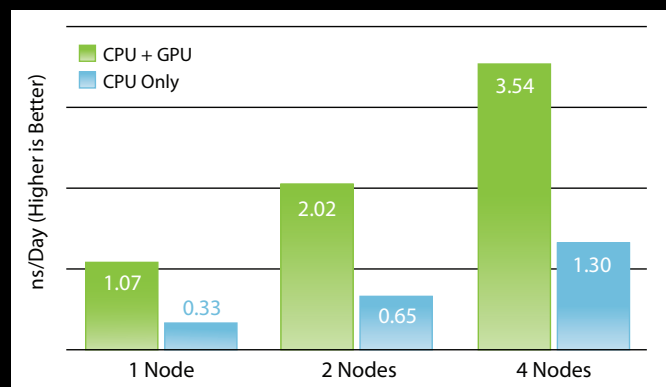
As you can see, I defined a special mask of ?d?l (0–9a–z) and assigned it to 1, then I created a password pattern where the first character was ?u (A–Z), the next three were ?l (a–z), and the final four were ?1 (0–9a–z). The complete command to attempt this mask attack against my phpass hashes with my new custom GPU tuning would be:

```
/path/to/mp32.bin -1 ?d?l ?u?l?l?l?l?l?l?l?l | \  
/path/to/oclHashcat-plus32.bin -m 400 -n 80 \  
-o recovered_hashes phpass-hashes
```

Microway's Proven GPU Expertise

Thousands of GPU cluster nodes installed.
Thousands of WhisperStations delivered.

- ▶ Award Winning BioStack – LS
- ▶ Award Winning WhisperStation Tesla – PSC with 3D



NAMD F1-ATP Performance Gain

Visit Microway at SC11 Booth 2606



Attack Rules

The final way to improve your attacks further is by applying rules to your dictionary attacks. A rule allows you to perform some sort of transformation against all the words in your dictionary. You might, for instance, not only try all your dictionary words, but also create a rule that adds a single digit to the end of the dictionary word. That will catch even more weak passwords and only increases the number of overall combinations by ten times.

Here's an even better example of how rules can help crack more tricky passwords. With the new requirement that users must have numbers in their password, a lot of users have resorted to "leet speak". For instance, instead of using "password" they might use "p455w0rd". The fact of the matter is, they still are using a dictionary word—they are just applying a basic transformation to it where a becomes 4, s becomes 5, o becomes 0, e becomes 3 and so on. When you want to crack such a password, all you have to do is add the `-r` option to hashcat and point it to a file that contains the rule you want to apply. Hashcat uses a custom language to define rules, but it's not too tricky to figure out, and the installation directory for oclhashcat has a rules directory that contains a number of rule files you can use as a reference. It even already includes a rule for leet

speak, so if you wanted to perform a dictionary attack that took leet speak into account, it would look something like this if you ran it from within the oclhashcat-plus directory:

```
/path/to/oclHashcat-plus32.bin -m 400 \  
-r ./rules/leetspeak.rule \  
-o recovered_hashes example400.hash example.dict
```

For more information about rules, check out the documentation on the Hashcat Wiki at http://hashcat.net/wiki/rule_based_attack.

You now should have everything you need to refine your (completely legitimate and white hat) password-cracking attacks. On the Hashcat Wiki, you will find even more examples of types of attacks and examples you can use to improve your odds of cracking a password hash. ■

Kyle Rankin is a Sr. Systems Administrator in the San Francisco Bay Area and the author of a number of books, including *The Official Ubuntu Server Book*, *Knoppix Hacks* and *Ubuntu Hacks*. He is currently the president of the North Bay Linux Users' Group.

Resources

Main Hashcat Site: <http://hashcat.net>

Hashcat Wiki: <http://hashcat.net/wiki>

Hashcat Rules Documentation:
http://hashcat.net/wiki/rule_based_attack

SAVE THESE DATES FOR 2012



9th Annual

2012 HIGH PERFORMANCE LINUX FINANCIAL MARKETS Show and Conference

APRIL 2, 2012 (MONDAY)

ROOSEVELT HOTEL, NYC

Madison Ave and 45th St, next to Grand Central Station

Plan now to attend, exhibit and to sponsor.

High Performance Computing systems including HPC, Big Data, Cloud, Linux, Low Latency, Data Centers, Networking Systems, Virtualization, Optimization, Grid, Blade, Cluster are the hot technologies for Wall Street. Traders are speeding their orders to market faster while lowering computer costs. This High Performance Computing marketplace will assemble 800 Wall Street IT professionals for a convenient networking and meeting place in New York.

HPC systems offer cost savings alternatives for Wall Street IT directors looking to replace aging systems.

Go online to register or get more information on exhibiting and sponsoring.

Visit: www.flaggmgt.com/linux



Show Management:
Flagg Management Inc
353 Lexington Ave, NY10016
(212) 286 0333
flaggmgt@msn.com



9th Annual

2012 HIGH PERFORMANCE COMPUTING FINANCIAL MARKETS Show and Conference

SEPTEMBER 19, 2012 (WEDNESDAY)

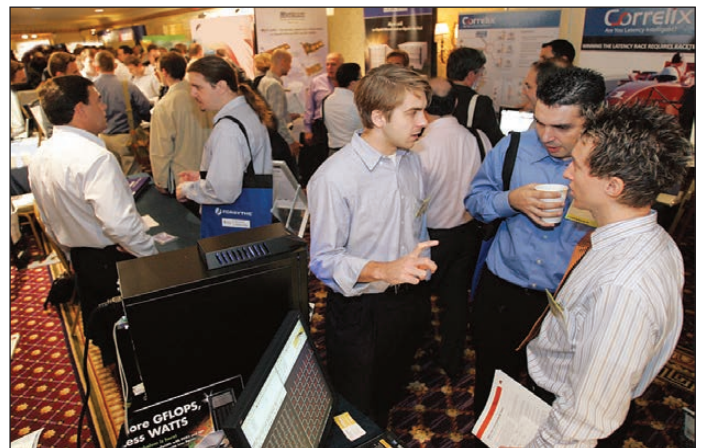
ROOSEVELT HOTEL, NYC

Madison Ave and 45th St, next to Grand Central Station

High Performance Computing, Big Data, Cloud, Linux, Low Latency, Data Centers, Networking Systems, Virtualization, Optimization, Linux, Grid, Blade, Cluster – the largest meeting of High Performance Computing in New York in 2012.

This major HPC network and marketplace will assemble 800 Wall Street IT professionals at one time and one place in New York. High performance computing, now cover all-in-one technologies to increase speed while reducing costs, space, heat, and energy consumption.

This show will focus on speed while lowering computer costs on Wall Street.



Go online to register or get more information on exhibiting and sponsoring.

Visit: www.flaggmgt.com/hpc

Show Management:
Flagg Management Inc
353 Lexington Ave, NY10016
(212) 286 0333
flaggmgt@msn.com



SHAWN POWERS

LTSP, Part I: the Skinny on Thin Clients

If you have a few older computers and a second Ethernet card, give the Linux Terminal Server Project a try.

One of my very first articles for *Linux Journal* was in August 2007 about the Linux Terminal Server Project (LTSP). The other article I wrote in that issue was about my MAME arcade system. Oddly enough, five years later, the most common questions I get from readers are about LTSP. And, the most common questions from my students are still about the arcade system! A lot has changed in the past half-decade, so in my next few articles, I explore the new face of thin clients.

MueKow, Because Geeks Have the Best Code Names

The term “thin client” often refers to a specific type of computer, but when it comes to LTSP, “thin client” means any computer that boots via the network and runs its operating system from a remotely mounted image.

The “thin” moniker is used because there is no need for the workstation to have a hard drive. This type of system offers several advantages over traditional workstations:

- All workstations boot from a single image, meaning updates and changes need to be done only once.
- The network “hard drive” is mounted in read-only mode, so it’s impossible to corrupt an individual computer.
- Hard drive failures are no longer an issue.
- Stolen workstations contain no data, because everything is stored on the server.

The LTSP process works like this:

Version 5 of LTSP, code-named MueKow, changed the way the network-mounted system was created and maintained.

1. Workstation boots via PXE.
2. DHCP server tells workstation where to get its kernel.
3. Workstation downloads kernel via TFTP.
4. Workstation mounts root directory in read-only mode over the network.
5. Workstation loads X11 locally and connects to the server.
6. All programs, except for X11 itself, run on the server, requiring minimal resources on the thin client.

In versions of LTSP before 5.0, the root directory was a specialized system mounted over NFS. It was stripped down to contain only the bits required to get X11 running, and then it pointed clients to the server via XDMCP. This had the advantage of requiring very minimal resources on the workstation (I'm talking requirements as low as Pentium I CPUs and 32MB of RAM or less). This also meant it was very difficult, or even impossible, to customize the

workstation's operating system.

Version 5 of LTSP, code-named MueKow, changed the way the network-mounted system was created and maintained. Instead of a specialized Linux system, it used a chroot environment containing a minimal install of the same operating system running on the server. Workstations still booted the same way, but now the chroot environment could be updated and customized. There also were other changes under the hood, like using SSH instead of XDMCP and creating a custom display manager (LDM). An NBD (network block device) server was used instead of NFS, increasing network efficiency as well.

If It Ain't Broke, Why Fix It?

A big motivation for changing the way LTSP managed its underlying operating system was that workstations, even outdated ones, were far too powerful to waste. Because traditionally, all applications ran on the powerful LTSP server, it would become overloaded quickly when users tried to run Adobe Flash or Java apps. With the new chroot environment, it became possible to run some apps locally and some apps on the



Figure 1. Old workstations make perfect LTSP thin clients. These machines were donated as junk, but they'll make excellent student workstations.

server. This meant servers could handle more thin clients connected to them, and that workstations shared the load. It also meant a runaway by an application like Firefox would use 100% of the *workstation* CPU, and not the server. I'll dive into local apps in Part II of this series, but I wanted to mention it now as it was a prime motivation for MueKow.

LTSP 5's new methodology does increase the system requirements for the thin clients themselves. Basically, whatever server system you're running (Ubuntu 11.10, for example) must be supported for the thin-client hardware. Because Ubuntu 11.10 requires at least a Pentium 4, so does LTSP 5 running on Ubuntu 11.10. (The Ubuntu kernel might actually boot on a Pentium 3, but if you're sticking with recommended CPUs, P4 is where

it's at.) Because very little actually is running on the thin client itself, it's possible to skimp on RAM. Although a minimum of 256MB is recommended, I've used 128MB systems successfully.

So what this all means is that LTSP has shifted the responsibility of defining "minimal configuration" to the server's operating system. In general though, it's good to have thin-client machines with the following:

- Pentium 4 or greater CPU.
- 256MB of RAM.
- PXE-bootable network card.

If the computer doesn't support PXE booting, it's possible to use gPXE to boot the computer from the network. Although not terribly difficult to configure, gPXE is beyond the scope of this article. For more information, check out <http://www.etherboot.org>.

We've Secretly Replaced Your Hard Drive with an Ethernet Port

LTSP requires a good network infrastructure. There's really no way to sugarcoat it; it just does. Because the operating system is mounted over the network, any time the thin client needs to access its "hard drive", it has to communicate over the network. Thankfully, LTSP 5 is more efficient at this than previous

versions, because instead of the traditional NFS-mounted filesystem, LTSP 5 uses NBD. The Network Block Device serves a single file, which is an image of the underlying file structure. This distinction means NBD is significantly faster and strains the network less than NFS. Even with that, however, LTSP requires a good network infrastructure. A gigabit-switched backbone with at least 100Mbit switched connections for each thin client is recommended. Anything less will really affect performance.

Thankfully, by default, LTSP runs in a split network environment. That means the server has two Ethernet cards. One card connects to the main network, and the other creates a NAT to which the thin clients can connect. This is a great way to isolate a thin-client lab, especially when a beefy network infrastructure isn't available. This method means the thin clients must be connected physically to the same switch as the NAT side of the server, but for smaller installations, that's usually not a problem. (I'll talk about larger thin-client installs in later articles.) When failover and high availability come into play, a good site-wide network infrastructure is really required.

It's Not the Size of Your Server, It's How You Use It

Part of the confusion behind LTSP is that it's very flexible, so a "standard"

install is a misnomer from the beginning. Like I just mentioned, the default installation method is to use a server with two Ethernet cards and create a private NAT'd network for the thin clients to live on. One huge advantage to this sort of install is that a modern workstation-class computer can act as a server for a small handful of thin clients. A dual-core workstation with 4GB of RAM easily could host 4–5 thin clients and still work as a desktop workstation itself. This setup is very attractive for teachers who want to provide terminals for their classrooms.

Every LTSP install is slightly different, so it's also difficult to judge how big a "server" needs to be to support X number of thin clients. You can make some educated guesses, but honestly, the best way is to test and see. If Ubuntu recommends 512MB of RAM, you can see the aforementioned workstation/server has eight times as much RAM as is required. Based on that rough figure, 4–5 thin clients should be able to share the resources of the server computer and still run efficiently. That's obviously a very rough figure, but you need to start the trial and error somewhere!

Because LTSP depends so much on the network in order to function, your server, whatever size, really should have gigabit Ethernet. Thin clients can run just fine with 100Mbit connections, but the server should have gigabit. Once

If you have a server with two Ethernet cards, Ubuntu's Alternate CD can set up everything you need automatically.



Figure 2. Ubuntu's Alternate CD makes installing an LTSP server simple.

you have your server ready to install, and your thin clients (whether they are old workstations or fancy new thin-client devices) ready to boot from the network, it's time to install LTSP.

Ubuntu Makes Questions Easier to Ask

I recommend Ubuntu for your first LTSP experience. The simple reason is that most LTSP folks use Ubuntu, so it's easier to find support. Before version 5, LTSP was pretty closely tied to Red Hat-based operating systems. Now, with the MueKow concept, LTSP no longer is tied to a specific distribution. For the purpose of this article, however, I'm assuming Ubuntu is the distribution used. (It should be fairly easy to adapt

to other distributions.)

If you have a server with two Ethernet cards, Ubuntu's Alternate CD can set up everything you need automatically. Boot up your server computer from the Alternate CD, and press F4. You'll see "Install an LTSP server" as one of the options (Figure 2). If you select that option, Ubuntu installs like normal, and at the end of the install, you'll see it build the chroot environment for LTSP.

Once the installation is complete, any thin clients connected to your second Ethernet card should be able to boot via PXE directly into an Ubuntu session. While I'm a big fan of magic, I also like to know how it works. So in a nutshell, here's what's going on behind the scenes during the install:

- A TFTP server is installed and activated on the second Ethernet port.
- A DHCP server is installed and activated on the second Ethernet port. (Note: it's important to keep the second Ethernet port off your main network, because your DHCP server could mess up the rest of your network—keep the second Ethernet port separate!)

- The LTSP-specific software is installed on the server. This includes things like LDM, which is the login screen thin clients display when they boot up.
- A minimal Ubuntu install is put into a chroot in your /opt/ltsp directory. This is a complete Ubuntu system with X11 support, but it has minimal applications installed, because by default, it launches only X11 then connects to the server.
- The chroot folder is compressed into an NBD (Network Block Device) image.
- An NBD server is installed and activated on the second Ethernet port, which serves the NBD image just created.
- The kernel is copied from the chroot environment to the TFTP server.

You're Done—Now Get Started

I mentioned the abstract process thin clients use when booting from the server. Now that you know how LTSP 5 is set up, let me elaborate a bit. When you power up your thin clients, this is what happens:

1. The thin client, connected to the same network switch as the server's second Ethernet port sends out a PXE request.
2. The DHCP server responds, telling the thin client the address of the

server (192.168.1.254 by default) and the name of the kernel image to download.

3. The thin client downloads the kernel via TFTP from the server's TFTP server.
4. Once the thin client loads the kernel, it mounts the NBD image of the root filesystem via NBD.
5. The thin client starts X11 and connects to LDM on the server.

System on Module

New - SoM-3517

- TI ARM Cortex-A8 600 MHZ Fanless Processor
- Up to 512 MB of DDR2 SDRAM
- UP TO 1gb of NAND Flash
- 2 High Speed USB 1.1/2.0 Host ports
- 1 High Speed USB OTG port
- 4 Serial Ports, 2 I2C and 2 SPI ports
- Processor Bus Expansion
- 10/100 BaseT Fast Ethernet
- CAN 2.0 B Controller
- Neon Vector Floating Point Unit
- 16-bit DSTN/TFT LCD Interface
- 2D/3D Accelerated Video w/ Resistive Touch
- Small, 200 pin SODIMM form factor (2.66 x 2.375")






The SoM-3517 uses the same small SODIMM form-factor utilized by other EMAC SoM modules, and is the ideal processor engine for your next design. All of the ARM processor core is included on this tiny board including: Flash, Memory, Serial Ports, Ethernet, SPI, I2C, I2S Audio, CAN 2.0B, PWMs, Timer/Counters, A/D, Digital I/O lines, Video, Clock/Calendar, and more. The SoM-3517M additionally provides a math coprocessor, and 2D/3D accelerated video with image scaling/rotation. Like other modules in EMAC's SoM product line, the SoM-3517 is designed to plug into a custom or off-the-shelf Carrier board containing all the connectors and any additional I/O components that may be required. The SoM approach provides the flexibility of a fully customized product at a greatly reduced cost. Contact EMAC for pricing & further information.

<http://www.emacinc.com/som/som3517.htm>

Since 1985
OVER
25
YEARS OF
SINGLE BOARD
SOLUTIONS

EMAC, inc.

EQUIPMENT MONITOR AND CONTROL

Phone: (618) 529-4525 • Fax: (618) 457-0110 • Web: www.emacinc.com

6. The thin client is ready to log in!

With this basic install of LTSP, all applications are executed on the server. This is a confusing concept for many folks, but I explain to my users that a thin client is basically a remote keyboard/mouse/monitor for the server. When a user starts Firefox, for example, the application starts on the server—you just control it remotely. If you're familiar with X11 forwarding over SSH, the concept should be easier to wrap your brain around.

Because everything is done on the server machine, any users or applications added to the server computer are available on the thin clients. This means LTSP users are simply users on the Ubuntu box, and they can be added or deleted using the standard Ubuntu tools.

Even with my explanation of how thin clients boot and what the server does in the background, you'll notice there are still some mysterious things going on. Sound probably "just works" on the thin client, although that's usually not the case with remote X11 apps. A few other hurdles have been conquered with LTSP 5 that historically were a problem. Things get much more complex when you start running some applications locally on the thin client and some applications remotely—but that's for next month's article.

This Month, Try to Break Things!

Now that you have a fully running Ubuntu system on all your thin clients, see if you can find some of the limitations of such a system. If you have a classroom of kids to use as guinea pigs, have them use Adobe Flash-based Web sites, and see if you can notice your server slowing down. Install a printer on your server, and notice how all the thin clients automatically have access to it. Notice how LibreOffice loads lightning fast after it's been opened on one machine (it gets loaded into memory).

LTSP is a powerful way to utilize older hardware. It also can make system maintenance minimal, because there is only a single install of Ubuntu to keep updated. To be honest, I've barely scratched the surface in this article—you can tweak LTSP to do some amazing things. In my next few articles, I'll cover local apps, print servers, network tweaks, load balancing and more. If you have a few older computers and a second Ethernet card, I urge you to give LTSP a try. By the time you're done, you'll be able to make your thin clients dance an Irish jig. (Or whatever the geeky network equivalent of Irish jigs might be!) ■

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

MySQL Conference & Expo 2012

- ▶ 60+ MySQL experts in 16 tutorials and 72 breakout sessions
- ▶ Keynote speakers from HP, Facebook, Percona and more

April 10-12
Santa Clara, CA

▶ Check it out

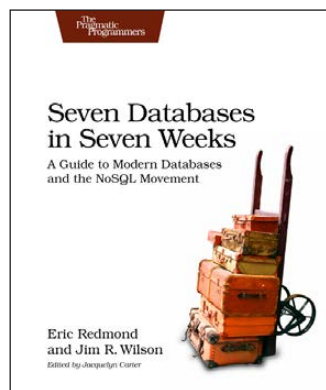
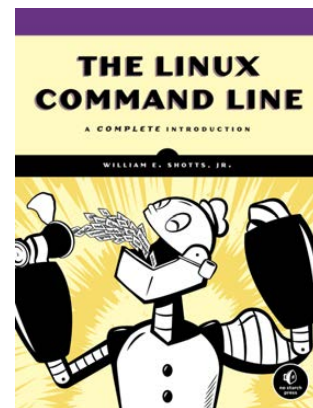


www.percona.com/live

William E. Shotts Jr.'s *The Linux Command Line* (No Starch)

Once you've dived into William E. Shotts Jr.'s *The Linux Command Line: A Complete Introduction*, you may find your mouse gathering dust, says the book's publisher No Starch Press. In order to appreciate Linux's premier advantages fully—its power, customizability and rich UNIX-supercomputer heritage—every new user should fire up the Bash shell and explore what's possible with the command line. Shotts takes readers from first keystrokes to writing full programs in Bash. Along the way, they will learn the timeless skills handed down by generations of gray-bearded, mouse-shunning gurus: file navigation, environment configuration, command chaining, pattern matching with regular expressions and more. In addition to that practical knowledge, Shotts reveals the philosophy behind these tools and the rich heritage that desktop Linux machines have inherited from their UNIX forebears. Shotts' hope is that his readers eventually will find that the command line is a natural and expressive way to communicate with a computer.

<http://www.nostarch.com>



Eric Redmond and Jim R. Wilson's *Seven Databases in Seven Weeks* (The Pragmatic Bookshelf)

Data is getting bigger and more complex by the day, and so are the choices in handling it. Cutting-edge solutions for managing this complexity range from traditional RDBMS to newer NoSQL approaches, seven of which are explored in Eric Redmond and Jim Wilson's new book *Seven Databases in Seven Weeks: A Guide to*

Modern Databases and the NoSQL Movement. The book is a tour of some of the hottest open-source databases today that goes beyond basic tutorials to explore the essential concepts at the core of each technology. With each database—Redis, Neo4J, Couch, Mongo, HBase, Riak and Postgres—readers tackle a real-world data problem that highlights the concepts and features that make it shine. Readers also explore the five data models employed by these databases: relational, key/value, columnar, document and graph to determine which kinds of problems are best suited to each, and when to use them.

<http://www.pragprog.com>

Xi3 Corporation's TAND3M Software



Go green and save green with Xi3 Corporation's TAND3M Software, an application that enables two people to share and use one (Linux or Windows) Xi3 Modular Computer for all computing functions simultaneously. Xi3's CEO says that by taking advantage of the underutilized resources of the Xi3 Modular Computer, the operating system and the other applications, computing costs are cut in half without a drop-off in performance. Each Xi3 Modular Computer needs only 20 Watts to operate, and the addition of TAND3M Software halves the electricity needs to a mere 10 Watts per seat. Such low power use is due in large part to Xi3's unique architecture that separates the traditional computer motherboard into three distinct pieces: one for processor and memory option, a second for display and power options, and a third for I/O or special connectivity options. Xi3 says that its patented design and architecture solves many of the problems blocking the progress of advanced computing.

<http://www.Xi3.com>

Wind River Solution Accelerators for Android

Intel subsidiary Wind River Software wants to help you jumpstart your Android device development with its new Wind River Solution Accelerators for Android. The three specialized software offerings—one each for user experience, connectivity and medical-specific devices—can accelerate Android device development and reduce engineering time and cost to help developers turn around high-quality devices faster than ever. Modular in nature, the offerings give developers flexibility to pick and choose software components to fill in gaps in expertise and instantly integrate complex, differentiating features. The user-experience module accelerates boot times and supports features like multi-windowing screen navigation, multimedia functionalities and advanced firmware management. The connectivity module supports multimedia interoperability capabilities via the DLNA standard, SyncML support and FM radio capabilities. The medical module helps medical device manufacturers leverage Android's platform richness and flexibility for innovation.

<http://www.windriver.com>

WIND RIVER

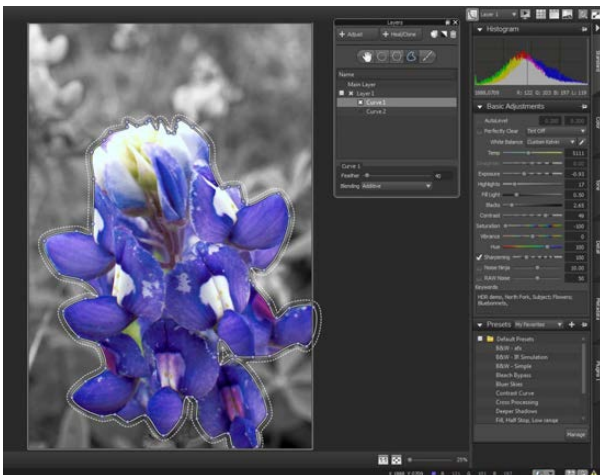
Apache Hadoop

Although the organizations that already use Apache Hadoop make up a Who's Who of the Net, this open-source framework for reliable, scalable, distributed computing officially has achieved the level of enterprise-readiness to earn a 1.0 designation. A foundation of cloud computing and at the epicenter of "big data" solutions, Apache Hadoop enables data-intensive distributed applications to work with thousands of nodes and exabytes of data. The framework enables organizations to store, process, manage and analyze the growing volumes of data being created and collected every day more efficiently and cost-effectively. It can connect thousands of servers to process and analyze data at supercomputing speed. Version 1.0 reflects six years of development, production experience, extensive testing and feedback. New features include support for HBase, strong authentication via Kerberos, Webhdfs, performance-enhanced access to local files for HBase and other features and fixes. The Apache Software Foundation directs the development of Apache Hadoop.

<http://www.apache.org>



Corel AfterShot Pro



Corel, whose support for Linux goes way back in interesting ways (remember Corel Linux OS?), has announced the release of its new AfterShot Pro, an application the company markets as "a total photographic workflow solution for professional and enthusiast photographers". Key product features include RAW workflow; flexible photo management; batch editing; robust metadata tools; easy integration with other image editors (such as Corel PaintShop Pro and Adobe Photoshop);

advanced, non-destructive editing; and "breakthrough performance". AfterShot Pro, which runs on Linux, Mac OS and Windows, is positioned as a powerful and affordable alternative to products like Adobe Photoshop Lightroom and ACD Systems ACDSee Pro.

<http://www.corel.com>



Opera TV Store

If you thought Opera was just a niche browser company, think again. The company recently released its newest offering, the Opera TV Store, a HTML5-based store solution for connected TVs. Opera says that its goal with the TV Store is “to bring apps

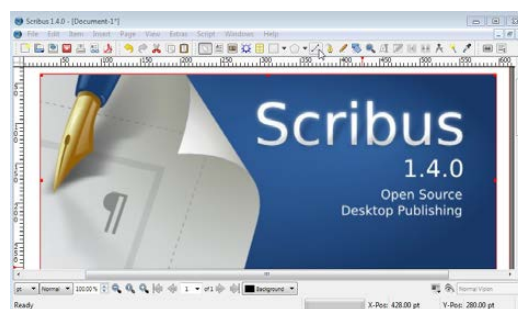
to the world of TV in a huge way” by providing users with a “lean-back Web experience” and to provide developers, content providers and manufacturers with “convenient, cross-platform technology”. The store is optimized for HD-Ready screens and standard remote controls, so users simply can fire up the cool apps, including video, games and news. The Opera TV Store can be installed by OEMs on any set-top box, Blu-ray player or HD-Ready TV, and manufacturers can harness the power of TV apps on any device running the Opera Devices SDK. Opera also says that “OEMs can rest assured that their users will get the best possible experience, without having to worry about the content themselves.”

<http://dev.opera.com/tv>

Scribus

After an intense, four-year-long slog, the developers of Scribus—the multiplatform, open-source, professional desktop publishing application—have released the new stable version 1.4. The upgrade integrates a whopping 2,000+ feature requests and bug resolutions. The most notable new feature is that Scribus 1.4 is now based on Qt4, which developers say enables it to run equally reliably on all supported platforms. Other feature highlights include improved object handling, advanced options for text and typography, undo/redo for nearly all text-related actions, usability improvements, new features for vector objects, better fill handling, additional color palettes, a rendering frame, more vector import filters and much more.

<http://www.scribus.net>



Please send information about releases of Linux-related products to newproducts@linuxjournal.com or New Products c/o Linux Journal, PO Box 980985, Houston, TX 77098. Submissions are edited for length and content.

Swap Your Laptop for an iPad + Linode

Ditch your laptop and code in the cloud—it's easier than you'd think.

MARK O'CONNOR

On September 19, 2011, I said goodbye to my trusty MacBook Pro and started developing exclusively on an iPad + Linode 512. This is the surprising story of three months spent working in the cloud.

It all started when I bought my first MacBook a couple years ago; despite a decade using Linux, I never really was satisfied with either GNOME or KDE. In OS X, I hoped to find a slick window manager on top of a solid BSD implementation. I enjoyed hardware with drivers that “just worked”, but I missed decent package management and the rest of the Linux ecosystem.

Although I like to use Python and GAE

for my own projects, at work, we write heavyweight C++/Qt code that runs on clusters, such as the 200,000 processor Jaguar machine, so most of my day was spent in Linux anyhow, a lot of it on remote systems. Typically, I'd develop in MacVim locally and run my code remotely or on Ubuntu under VMware Fusion.

One fateful day, VMware and OS X conspired to trash my shared filesystem. While dd was recovering as much as it could, I started toying with the idea of giving up on local filesystems altogether. To my surprise, it seemed possible—even plausible. The more I thought about it, the more attractive it seemed. I knew then, I just had to try.

Figure 1.
iPad on Sofa

```
iPad 10:05 AM 88%
+ 1 2 Ctrl Alt Esc Tab F/ Paste X
39
40
41     enum Status
42     {
43         NotStarted,
44         Running,
45         Stopped,
46         Terminated
47     };
48 public:
49     Coroutine(),
50     virtual ~Coroutine();
51
52     void createStack(int size = DefaultStackSize);
53     void setStack(void *memory, int size);
54     void setAutoDelete(bool autoDelete);
55     bool autoDelete() const
56     { return autoDelete; }
57
58     bool cont();
59     static void yield();
60
61     Status status() const
62     { return status; }
63
64     bool isRootCoroutine() const;
65
66     void stackInfo(void **data, size_t *size, void **base, void **current) const;
67
68     static Coroutine *currentCoroutine();
69
70 #ifdef qdoc
71     static Coroutine *build(Function function, ...);
72 #endif
73 // add declarations for static build(...) function
74 #include "coroutinebuilddeclaration.p.h"
75
76 protected:
77     virtual void run() {}
78
79 private: // not copyable
80     Coroutine(const Coroutine &);
81     Coroutine &operator=(const Coroutine &);
82
83 private:
84     static void yieldHelper(Status stopStatus);
85     static void entryPoint();
86     void freeStack();
87
88     void * stackData;
89     size_t stackSize;
90     void * stackBase;
91     void * stackPointer;
92     Coroutine * caller;
93     Status status;
94     bool autoDelete;
95     bool ownStack;
96 };
97
98 // add definitions for static build(...) function
99 #include "coroutinebuilddefinition.p.h"
100
```



Figure 2. iPad and Keyboard Box

The Setup

It turns out you need a little more than just an iPad and a dream, but not too much more:

- iPad 2 (16Gb, Wi-Fi).
- Apple wireless keyboard.
- Stilgut adjustable angle stand/case.
- iSSH (and optionally Jump).
- Linode 512 running Ubuntu 11.04.
- Apple VGA adapter.
- Total cost: around \$800 + \$20 per month.

I chose a Linode 512, which has been perfect for my needs. You get fantastic CPU power and a tiny little bit of RAM. Surprisingly, this is enough when you do most of your work at the command line, but it's nice knowing I always can

upgrade it later.

I also turned on the \$5-a-month backups. Zero-effort data safety for less than the price of a hot chocolate? Yes, please!

Linode's interface walks you through adding your new node. Pick a region close to yourself—you want to minimize the roundtrip time to the server. I spend most of my time in Munich, so I have mine in London and get a 30–40ms ping, which is great.

I run Ubuntu 11.04 in 32-bit mode on mine, but they offer a wide selection of pre-built Linux images along with the option to install your own.

For access to the server, you need a really good SSH client. On the iPad, I've tried both iSSH and Prompt, and of these, only iSSH is even feasible for serious use. The hardest part of setting up an SSH client on a tablet is getting your private SSH key on there without entrusting it to a third party. I split mine across multiple services and removed it after recombining it, but a better way would be to SSH in with a password first, then use copy and paste to copy the key inside iSSH itself.

I Wandered Lonely as a Cloud

I typically start my day by catching up on the bug-tracker chatter, mercurial diffs and other e-mails with the iPad in one hand while lying on the Combinat56 sofa.

I actually hate the mail app for this—the stupid animation when archiving

posts adds unnecessary delay, and the archive button is uncomfortably placed at the top of the screen. More recently, I've been scanning e-mails over IMAP with a Python script instead.

Next, I lazily swipe to Safari and review my tickets for the day in our Web-based bug tracker then return to the keyboard and fire off a couple e-mails before settling back into coding—the new four-finger swipe gestures in iOS5 have really improved my life.

But, I was talking about coding, which brings me back to the only reason this setup works for me at all: Vim.

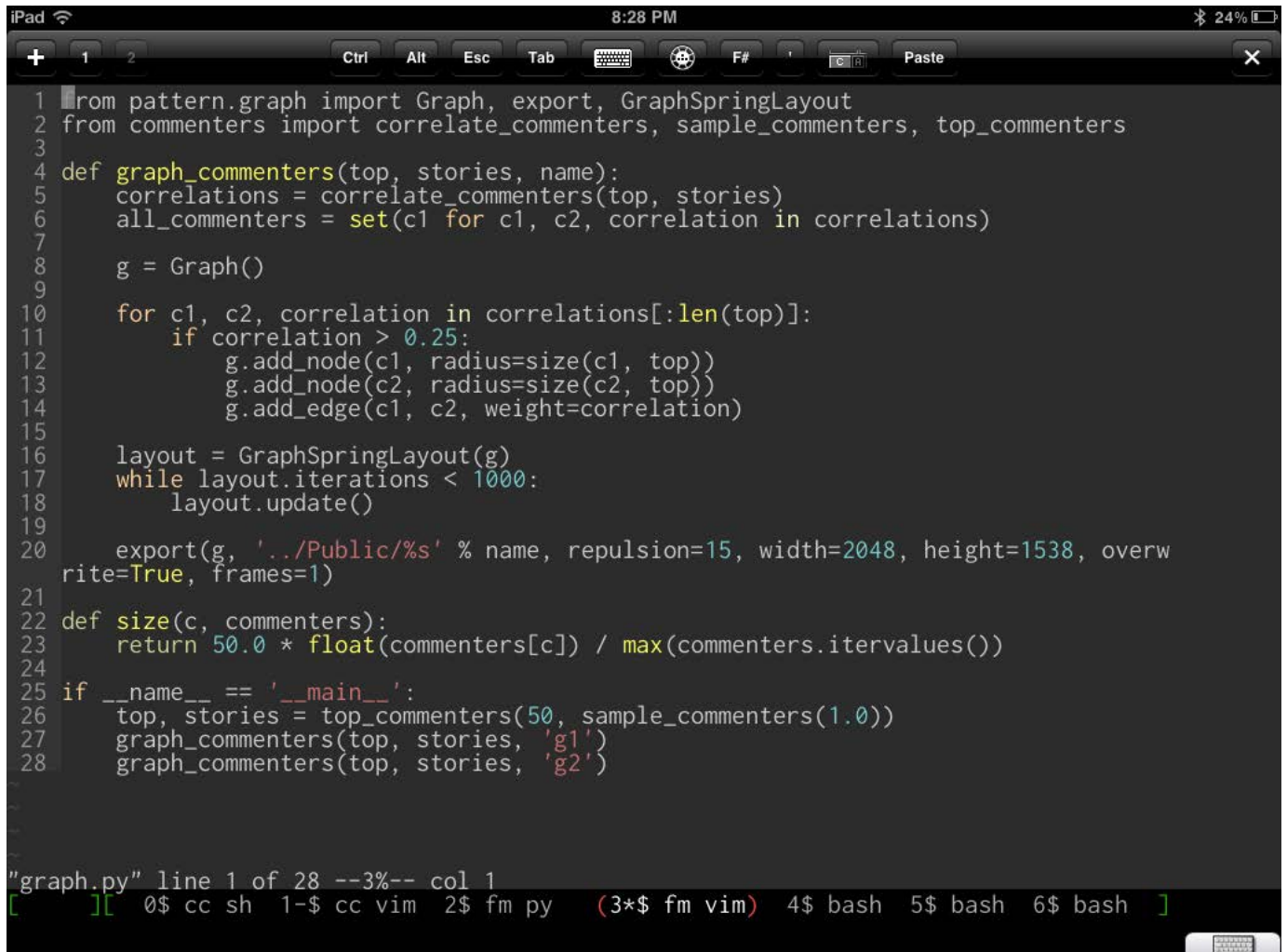
Vim: My Home from Home

Perhaps the only reason this transition has been so smooth is because my favourite editor/IDE looks and feels almost exactly the same running on an iSSH console as it did running locally on my MacBook. iSSH supports xterm-256color, which means you still can have pleasant color schemes despite working in a terminal. All my plugins are there, my code-completion, quick navigation and so on.

I found Ubuntu's default Vim didn't have everything I wanted, but don't worry! It's easy to download and build it yourself. See Listing 1 (also available at <https://gist.github.com/1357690>).

I honestly can't remember which plugins I installed and which ones I actually use. I just copied the existing .vim folder from my MacVim installation.

FEATURE Swap Your Laptop for an iPad + Linode



```
1 from pattern.graph import Graph, export, GraphSpringLayout
2 from commenters import correlate_commenters, sample_commenters, top_commenters
3
4 def graph_commenters(top, stories, name):
5     correlations = correlate_commenters(top, stories)
6     all_commenters = set(c1 for c1, c2, correlation in correlations)
7
8     g = Graph()
9
10    for c1, c2, correlation in correlations[:len(top)]:
11        if correlation > 0.25:
12            g.add_node(c1, radius=size(c1, top))
13            g.add_node(c2, radius=size(c2, top))
14            g.add_edge(c1, c2, weight=correlation)
15
16    layout = GraphSpringLayout(g)
17    while layout.iterations < 1000:
18        layout.update()
19
20    export(g, '../Public/%s' % name, repulsion=15, width=2048, height=1538, overw
rite=True, frames=1)
21
22 def size(c, commenters):
23     return 50.0 * float(commenters[c]) / max(commenters.itervalues())
24
25 if __name__ == '__main__':
26     top, stories = top_commenters(50, sample_commenters(1.0))
27     graph_commenters(top, stories, 'g1')
28     graph_commenters(top, stories, 'g2')
```

"graph.py" line 1 of 28 --3%-- col 1
[0\$ cc sh 1-\$ cc vim 2\$ fm py (3*\$ fm vim) 4\$ bash 5\$ bash 6\$ bash]

Figure 3. Vim on iPad

Listing 1. Building Vim

```
install-vim.sh #
apt-get install mercurial
apt-get build-dep vim
hg clone https://vim.googlecode.com/hg/ vim
cd vim
./configure --enable-rubyinterp --enable-pythoninterp
  --with-features=big
make
make install
```

I definitely use these every day though:

- a.vim
- cctree.vim
- clang_complete.vim
- color_sample_pack.vim
- command-t.vim
- cscope_maps.vim

- NERD_tree.vim
- scratch.vim
- searchfold.vim

You can get them all from <http://www.vim.org/scripts>, but you may want to install pathogen and get them that way instead. Note that command-t requires you to build a stub—follow the install instructions in command-t.vim, and it just works.

To use the clang_complete plugin, you'll need clang. Typing `apt - get`

`install clang` should do the trick. I had to play around with it a bit to get it working on my project, which included adding all the `-I` and `-D` command-line options to a `.clang-complete` file in the project directory.

Everybody configures Vim differently. Listing 2 shows my `.vimrc` (also available at <https://gist.github.com/1357590>). Some of the things here relate to our internal tracker system and won't be interesting for you, but it should be clear what most of these things do and which keys are bound to them.

In short, it's a seamless transition

Listing 2. `.vimrc`

```
"Set Mapleader
let mapleader = ","
let g:mapleader = ","

"Theme
colo zenburn

"NERDTree
map <Leader>, :NERDTreeToggle<cr>

"Taglist
let Tlist_Ctags_Cmd = '/usr/local/bin/ctags'
let Tlist_WinWidth = 50
map <Leader>. :TlistToggle<cr>

"Check python syntax
let g:pcs_hotkey = '<LocalLeader>x'

"Quick Open
let g:CommandTMaxFiles=30000
set wildignore+=*.o,.ddt-*,.treeserver-*
let g:CommandTMatchWindowAtTop = 1

"Scratch buffer
map <Leader>k :ScratchOpen<cr>

"RT
map <Leader>rt "zyiw:ScratchOpen<cr>:0r!~/Work/code/util/rt show -f
  ➤id,Subject,Queue,Status,Creator,Owner,LastUpdated,Priority
  ➤ticket/<C-R>z<cr>o<Esc>
map <Leader>tt "zyiw:r!~/Work/code/util/rt show -f Subject ticket/<C-R>z
  ➤\| grep Subject \| cut -d ' ' -f2-<cr>kA:<Esc>Jj

"ctags
set tags=tags;$HOME

"Rebuild cscope
map <Leader>cs !cscope -bqk<cr>:cs add cscope.out<cr>

"Buffer shortcuts
map <Leader>f :b#<cr>

"ddt log files
map <Leader>gdb :!cdata-from-log % >/tmp/cdata.log<cr>:e
  ➤/tmp/cdata.log<cr>

"Disable virtual bell
set vb t_vb="

"Make backspace work
set backspace=2

"Code navigation
map <Leader>gd ?^<cr>kf:ll

"Misc stuff
set autoread
set hidden
set hlsearch
set incsearch
set ignorecase
set smartcase
set smartindent
map <Leader>c :let @/ = ""<cr>
syntax enable
set nu
set textwidth=0 " No annoying word wrapping
set tabstop=4
set shiftwidth=4
set expandtab
set guifont=Menlo:h14
filetype on
filetype plugin on
set nocomp
autocmd FileType python set omnifunc=pythoncomplete#Complete
autocmd FileType javascript set omnifunc=javascriptcomplete#CompleteJS
autocmd FileType html set omnifunc=htmlcomplete#CompleteTags
autocmd FileType css set omnifunc=csscomplete#CompleteCSS
autocmd FileType xml set omnifunc=xmlcomplete#CompleteTags
autocmd FileType php set omnifunc=phpcomplete#CompletePHP
" Replaced by clang_complete for now
" autocmd FileType c set omnifunc=ccomplete#CompleteCpp
autocmd FileType ChangeLog set tw=80
```

FEATURE Swap Your Laptop for an iPad + Linode

from my MacVim environment. If I were developing OS X apps with Xcode or used Eclipse or Visual Studio regularly, this change probably would have killed me.

As it happens, working in the terminal on a remote Linode is even better than working locally, thanks to the magic of GNU Screen.

GNU Screen Is Magic

GNU Screen is like a window manager for your terminal sessions, giving you multiple tabs, searchable history, activity/idle notifications and—best of all—persistence.

So, I fire up iSSH, tap on my Linode connection and reconnect to the already-running Screen session. All my terminal tabs are exactly where I left them. Other SSH tunnels still are set up. My cursor still is in the same position. The clipboard is as I left it. It's as if I never left, because for my side projects, I have a different Screen session with a different set of tabs and editor instances running—perfect separation.

It's hard to overstate how pleasant it is to be able to return to exactly the same session each day. On a MacBook, there'd usually be some other distracting programs left that I'd opened in the meantime, and of course, any remote connections would have

been dropped. At the very least, I'd have used MacVim for something else in the evenings. It might be a largely psychological benefit, but it feels as if I can drop back into the flow almost as easily as I left it. (Listing 3 shows an example .screenrc file, also available at <https://gist.github.com/1357707>.)

The Good, the Bad and VNC

At work, we develop a graphical parallel debugger, so I can't spend all my time in the terminal. For hands-on tests and GUI work, I need X. iSSH has a workable, if not perfect, solution, but for a few extra dollars, I find Jump far superior.

Although it's still not as quick and accurate as using a mouse to interact with a traditional GUI program, both iSSH's on-screen "touchpad" and particularly Jump's tap circle work better than I'd expect. And as it happens, being limited isn't all that bad:

One good way to evaluate the usability of a program or dialog is to try to use the mouse with just one finger.—Joel Spolsky

VNC on the iPad isn't nearly as bad as pushing the mouse around with one finger, but it does make you consider users with lower screen resolutions,

```
py" line 1 of 28 --3%-- col 1  
[ 0$ cc sh 1-$ cc vim 2$ fm py (3*$ fm vim) 4$ bash 5$ bash 6$ bash ]
```

Figure 4. GNU Screen Bar

Listing 3. .screenrc

```
#
# Example of a user's .screenrc file
#

# This is how one can set a reattach password:
# password ODSJQf.4IJN7E # "1234"

# no annoying audible bell, please
vbell on

# detach on hangup
autodetach on

# don't display the copyright page
startup_message off

# emulate .logout message
pow_detach_msg "Screen session of \${LOGNAME} \${cr}:\${nl}ended."

# advertise hardstatus support to $TERMCAP
# termcapinfo * ' ' 'hs:ts=\E_:fs=\E\\:ds=\E_\E\'

# make the shell in every window a login shell
#shell -$SHELL

# autoaka testing
# shellaka '> |tcsH'
# shellaka '$ |sh'

# set every new windows hardstatus line to something descriptive
# defhstatus "screen: ^En (^Et)"

defscrollback 1000

# don't kill window after the process died
# zombie ""

#####
#
# xterm tweaks
#

#xterm understands both im/ic and doesn't have a status line.
#Note: Do not specify im and ic in the real termcap/info file as
#some programs (e.g. vi) will not work anymore.
termcap xterm hs@:cs=\E[%i%d;%dr:im=\E[4h:ei=\E[4l
terminfo xterm hs@:cs=\E[%i%p1%d;%p2%dr:im=\E[4h:ei=\E[4l

#80/132 column switching must be enabled for ^AW to work
#change init sequence to not switch width
termcapinfo xterm Z0=\E[?3h:Z1=\E[?3l:is=\E[r\E[m\E[2J\E[H\
\E[?7h\E[?1;4;6l

# Make the output buffer large for (fast) xterms.
termcapinfo xterm* 0L=10000

# tell screen that xterm can switch to dark background and has
# function keys.
termcapinfo xterm 'VR=\E[?5h:VN=\E[?5l'
termcapinfo xterm 'k1=\E[11~:k2=\E[12~:k3=\E[13~:k4=\E[14~'
termcapinfo xterm 'kh=\E[1~:kI=\E[2~:kD=\E[3~:kH=\E[4~:kP=
\E[H:kN=\E[6~'

# special xterm hardstatus: use the window title.
termcapinfo xterm 'hs:ts=\E];:fs=\007:ds=\E]2;screen\007'

#terminfo xterm 'vb=\E[?5h<200/>\E[?5l'
termcapinfo xterm 'vi=\E[?25l:ve=\E[34h\E[?25h:vs=\E[34l'

# emulate part of the 'K' charset
termcapinfo xterm
'XC=K,%\E(B,|\304,\\|\326,|\334,|\344,|\366,|\374,~|\337'

# xterm-52 tweaks:
# - uses background color for delete operations
termcapinfo xterm be

#####
#
# wyse terminals
#

#wyse-75-42 must have flow control (xo = "terminal uses xon/xoff")
#essential to have it here, as this is a slow terminal.
termcapinfo wy75-42 xo:hs@

# New termcap sequences for cursor application mode.
termcapinfo wy*
CS=\E[?1h:CE=\E[?1l:vi=\E[?25l:ve=\E[?25h:VR=\E[?5h:VN=
\E[?5l:cb=\E[1K:CD=\E[1J

#####
#
# other terminals
#

#make hp700 termcap/info better
termcapinfo hp700
'Z0=\E[?3h:Z1=\E[?3l:hs:ts=\E[62]p\E[0$~\E[2$~\E[1$]:fs=
\E[0]\E[61]p:ds=\E[62]p\E[1$~\E[61]p:ic@'

# Extend the vt100 description by some sequences.
termcap vt100*
ms:AL=\E[%dL:DL=\E[%dM:UP=\E[%dA:DO=\E[%dB:LE=\E[%dD:RI=\E[%dC
terminfo vt100*
ms:AL=\E[%p1%dL:DL=\E[%p1%dM:UP=\E[%p1%dA:DO=\E[%p1%dB:LE=
\E[%p1%dD:RI=\E[%p1%dC
# terminfo and termcap for nice 256 color terminal
# allow bold colors - necessary for some reason
attrcolor b ".I"
# tell screen how to set colors. AB = background, AF=foreground
termcapinfo xterm-256color 'Co#256:AB=\E[48;5;%dm:AF=\E[38;5;%dm'
# erase background with current bg color
defbce "on"

#####
#
# keybindings
#

#remove some stupid / dangerous key bindings
bind k
bind ^k
bind .
bind ^\
bind \
bind ^h
bind h
#make them better
bind 'K' kill
bind 'I' login on
bind 'O' login off
bind '}' history

# Yet another hack:
# Prepend/append register [/] to the paste if ^a^ is pressed.
# This lets me have autoindent mode in vi.
register [ "\033:se noai\015a"
register ] "\033:se ai\015a"
bind ^] paste [.]

#####
#
# default windows
#

# screen -t local 0
# screen -t mail 1 elm
# screen -t 40 2 rlogin faui40

# caption always "%3n %t%? @%u%?%? [%h]%"
# hardstatus alwaysignore
# hardstatus alwayslastline "%w"

# bind = resize =
# bind + resize +1
# bind - resize -1
# bind _ resize max
#
# attrcolor u "-u b"
# attrcolor b "R"

hardstatus alwayslastline
hardstatus string '%{gk}[ %{G}%H %{g}][%= %{wk}%?%-Lw%?%{=b kR}
\E[%{W}%n*%f%t%?(%u)%?%{=b kR}]{= kw}%?%+Lw%?%?%=%
\E[%{g}][%{Y}%l%{g}] %{=b C}[ %m/%d %c%l%{W}']
```

FEATURE Swap Your Laptop for an iPad + Linode

larger font sizes and mouse control that hasn't been honed by countless years playing *Quake* and *Starcraft*. I'd be lying if I said I hadn't wished iOS had Bluetooth mouse support some days.

Maybe It's a Lifestyle

Today is not one of those days. I unwrap a chocolate croissant, make a fresh cup of tea and settle down to work: a quick `hg pull -u &&` make to start the recompile, then Ctrl-X to my editor tab and carry on coding while the rebuild

happens. Ctrl-X is my screen's "hot key"; it defaults to Ctrl-A, but on a wireless keyboard, that leaves Unicode characters in the terminal—I assume this is related to Apple's support for some common Emacs keybindings in iOS. It's strange, but easy enough to work around—unless you're an Emacs user, I imagine.

After a few minutes of coding, the bar at the bottom of the screen notifies me that my compile has finished. I Ctrl-X back, start a sanity test suite running just to be sure nothing horrible was broken



Figure 5. iPad and Croissant

overnight, then carry on coding again. Compiling on the quad-processor Linode is around twice as fast as inside VMware on my MacBook was. It's also completely, blissfully silent. The only sound is the raindrop-like patter of the keyboard as I work. It doesn't even get warm—a surprisingly refreshing change from the keyboard of a hard-working laptop!

I swipe to the DuckDuckGo app and look something up in the Qt APIs, then swipe back. It's becoming second nature, but I still miss a keyboard shortcut for task switching.

After an hour or two of uninterrupted development, the UK team wakes up and the first instant messages arrive. Thank heaven for the iOS5 update! I use the imo messenger app, which is fine, but of course, before the notification center, each pushed message would interrupt whatever I was doing with a frustrating pop-up. Now, the new notifications center behaves much more like a growl notification would—it lets me know, and it gets back out of the way again.

I finish the function I was working on, then four-finger swipe to the chat app; it's my boss reminding me about our 11am conference call.

Skype-based conference calls work fine on both the iPad and the iPhone; at least, as well as VoIP calls ever work—that is, fine after a few false starts while someone reconfigures the Linux audio drivers. I appreciate not having to think about that anymore—with a thin,

consumer client and a powerful Linux server, I'm really enjoying the best of both worlds.

During the Skype call, my iSSH session timed out in the background. It's only held for ten minutes or so. Fortunately, a single tap reconnects me, and through the magic of GNU Screen, I'm back at exactly the place I left off again.

As is always the case, while fixing one bug, I encounter another, apparently unrelated one.

Instead of messing around with some Web interface, I've switched to grabbing screen dumps (I recommend `vncsnapshot`) and any log files or stack traces with a command-line script run on the Linode itself. We use the Best Practical RT tracker (for our sins), which conveniently comes with a Perl script to interact with it from the command line. Doing this is actually quicker and easier than uploading via a browser used to be—one command with a bug description and the attachments, and I'm done.

The Cloud Is Always with You

And, it's lunchtime already! I close the iPad and head off with the others. The remote rebuild I left going will still be there—no need to worry about any uploads getting interrupted; it's all happening in on the Linode, after all.

During lunch, I pull out my iPhone and connect to the Linode again. Unfortunately, the build failed early on—out of disk space on one of the office

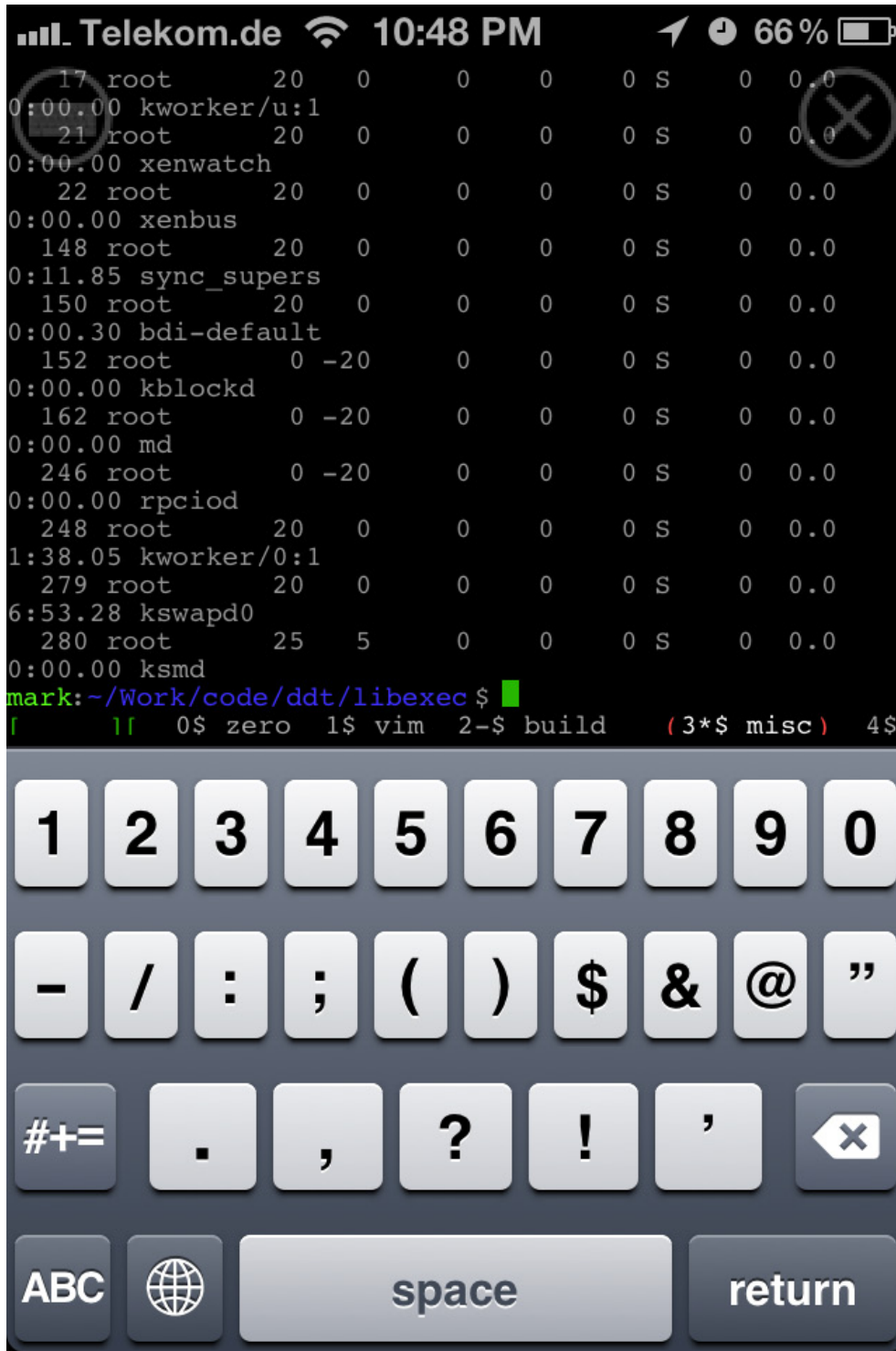


Figure 6. iPhone SSH

staging machines. The iPhone keyboard is somewhat painful to use, but for `rm -rf /tmp/build-2011-*` it suffices. I

spec/schedule changes my colleagues have made to an upcoming spec, stored in Google Docs.

kick off a new build while waiting for dessert to arrive.

Now I'm feeling nicely fed and just a little drowsy. To keep awake, I move to a standing workstation and set off some longer-running performance tests on a remote cluster. I find I move around between a lot of different working configurations with the iPad—much more frequently than when using a laptop. I'm not sure why—perhaps it's something about having a separate keyboard and screen that makes it easier to find a comfortable typing position, or perhaps it's just the novelty.

The tests will take an hour or so. I close iSSH and review the latest

Oh, Google Docs, Why Do You Hate Me So?

In a bitterly ironic twist, the only part of working in the cloud that doesn't work smoothly is using Google's cloud-based MS Office replacement. The mobile word processor is a joke, frequently losing the cursor position and sizable chunks of text. The spreadsheet version is better, but it's still a pathetic experience compared to the desktop version. There are no good native apps, so the only thing to do is to force Google to show the desktop versions of the sites and ignore the script errors. I wish I could convince my team to switch back to OpenOffice.org.

The tests have finally finished and the results are in. I move to the lounge area and connect to the big HD TV with the VGA cable—something about seeing console-mode Vim in giant HD makes me smile every time.

There are some oddities in the logs. I copy and paste a few snippets to a colleague and discuss them in chat. I'm glad copy and paste work smoothly, although having three different sets of buffers (iPad, Screen, Vim) does lead to confusion from time to time. Idly, I wonder if any of my colleagues have noticed I've been using an iPad instead of a laptop for the last month.

Six pm rolls around, and it's time to head home for the day. Despite a full day's intensive use, my iPad is still showing 15% battery life. I never bring a charger to work, and I've never needed one. That in itself is a taste of freedom.

At the End of the Day

Later that evening, I pull the iPad out and

open up Pages to finish a blog post. The house is quiet. Only the distant sounds of the city drift in from outside and mingle with the gentle patter of key presses on this delightful wireless keyboard.

I started this experiment because I fundamentally believe that most people don't want to rearrange windows, babysit their own general-purpose computers or back up their data. Sooner or later, almost everyone will work like this, and I wanted a taste of what that might feel like. I expected to find something that didn't work, but as the days turned into weeks and the weeks into months, I found I hadn't returned to my laptop even once.

I don't miss the weight. I don't miss the keyboard getting warm when I'm compiling. I don't miss its fragility, both physically and virtually. I don't miss running out of power. To my surprise, I find I am happy. Coding in the cloud isn't for everybody, but for my work flow, it's a perfect fit and I love it.

After a few minutes, the perfect peace is rudely disturbed by the twin jet-engine that is my MacBook fans spinning up to full power on the shelf behind me. I can't believe I used to put up with this all day and night.

Despite that, I leave the MacBook running. It's doing the only thing I still need it for on a regular basis. It's ripping DVDs. ■

Mark O'Connor is a Munich-based programmer, occasional writer and part-time startup founder. He believes in dynamic typing, first-class functions and the immortal essence of the human soul. He also likes tea. You can reach him at @yieldthought or <http://yieldthought.com>.

Plasma Active

a New Approach to Tablet Computing

KDE lines up a truly free tablet operating system.

STUART JARVIS

Why would you spend a few hundred dollars on a device that is little more than a smartphone (with a bigger screen, without the phone)?

Despite the success of Apple's iPad, that is a question that seems to have defeated most hardware and software vendors. MeeGo struggled to define a tablet user interface, never quite

managing more than a pre-release. It presented a few simple options, such as watching videos, playing music or browsing the Web—really no more than a modern phone with a larger screen. Even the iPad, an acknowledged success, is little more than an oversize iPhone. Its “wall of apps” approach has been largely copied by the Android-based tablets so far appearing on the market.

What Are Tablets For?

The tablet sitting in your hand (or unused in one of your drawers) is a real computer. Can it do more than browse the Web and play videos? Marco Martin, well-known KDE hacker and basysKom employee, thinks so: “the fact that people download and use thousands of apps shows that there is the desire to do something more”. He dislikes the way “most mobile applications feel quite disconnected with each other”. Marco believes this is where KDE’s new user interface and application set for touchscreen devices, Plasma Active, can shine.

Plasma Active takes a new approach

to touchscreen devices and tries to offer more than a set of applications for simple tasks. More than a desktop or even a notebook computer, a touchscreen device is likely to be carried around and used in different contexts, for different purposes. Plasma Active makes use of KDE’s Activities, something that has confused desktop users (see the Activities—*a Solution Looking for a Problem?* sidebar) but, the developers believe, makes sense on tablet devices.

Aaron Seigo, a founding member of the Plasma Active Project and one of the main drivers behind KDE’s Plasma family of user interfaces, hails Activities as a great step forward, claiming that

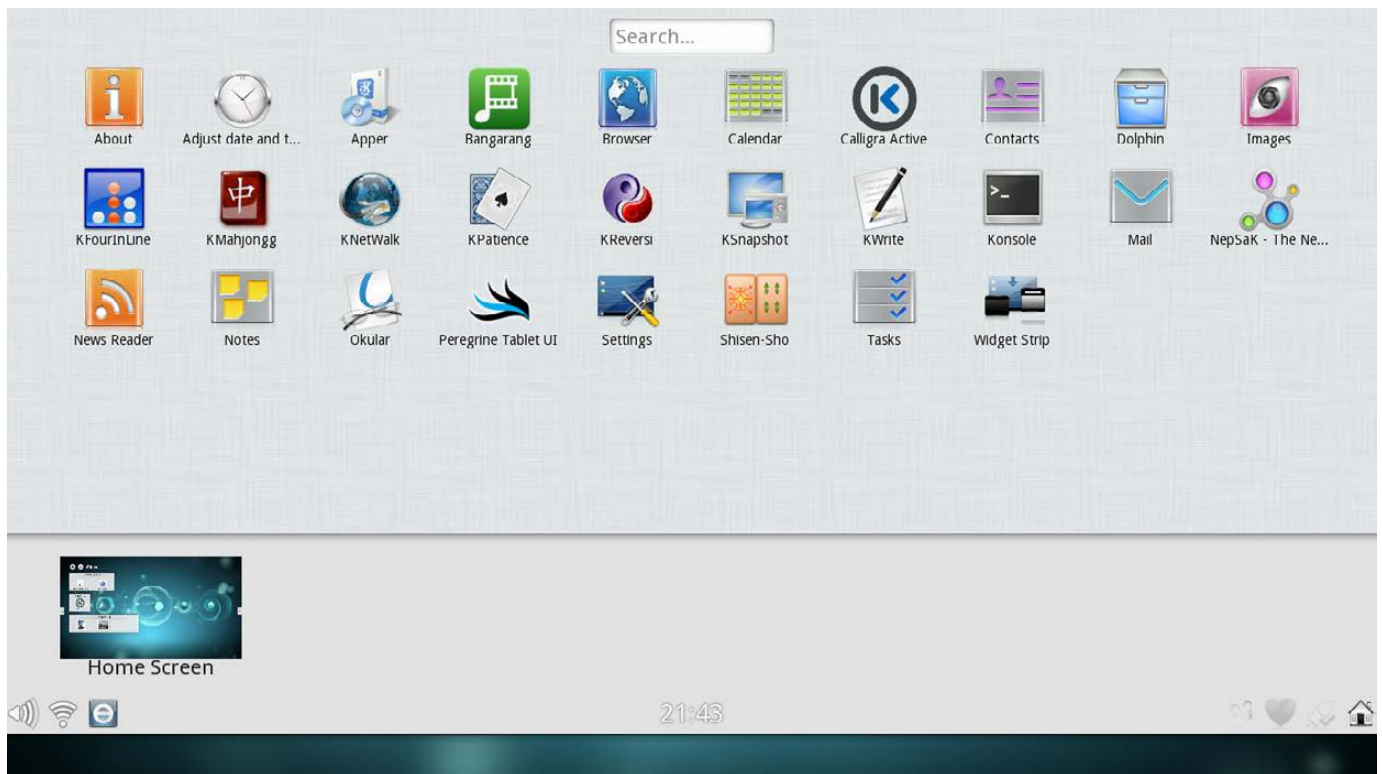


Figure 1. Plasma Active comes with a selection of applications, some more ready for touchscreens than others.

Activities—a Solution Looking for a Problem?

Since 2008, KDE has been pushing the concept of Activities, with mixed results. Many users have not been sure of the difference between Activities, designed to allow division of different types of tasks, and virtual desktops, which many people use to divide different types of tasks.

The idea is that although virtual desktops provide extra space and some grouping—for example, you might have different desktops for e-mail, Web, numerical work and graphics work—Activities provide customized interfaces for different tasks at different times.

Imagine you are a student at university. You might use virtual desktops as described above—with e-mail on one and lecture notes on another. But you might use Activities to differentiate between your courses, having one Activity for each set of lectures. So you can have a calculator widget on your desktops for your math lectures Activity, the periodic table for your chemistry lab Activity and quick access to your games in your free-time Activity. With its quick switcher and per-Activity recommendations, Plasma Active takes this concept further, making your tablet change its configuration completely at the scroll of the Activities wheel, so you can have it set up just how you want for every task you experience.

“many find the ability to sort their information and applications between different activities greatly increases the value of the device in their lives”. He uses a personal example: “while on a recent vacation I relied on Activities to keep track of our itineraries and plans, some work tasks and to keep up with things back home. I have a few Android tablets, and none of them would have been nearly as useful.”

First Impressions

When you launch Plasma Active for the first time (see the Try Plasma Active Two sidebar for how to try it), you are presented with what appears to be a fairly standard KDE desktop. The main things out of place are a panel at the top of the screen and the lack of an obvious application menu. Two small tabs halfway up either side of the screen also are not found in other KDE workspaces. Between them, these three items provide your

SELECTING AN ACTIVITY CHANGES THE DESKTOP BACKGROUND AND DESKTOP WIDGETS TO THOSE ASSOCIATED WITH THE ACTIVITY.

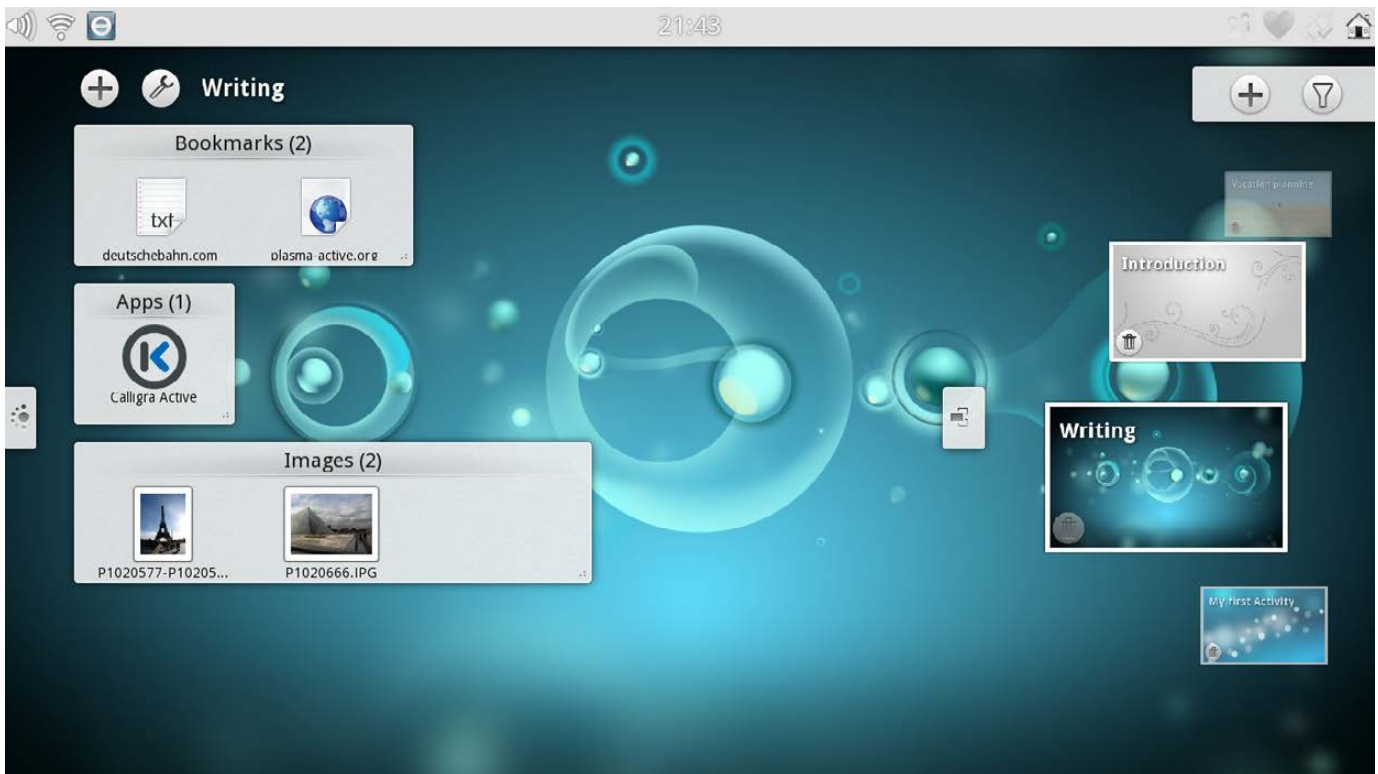


Figure 2. You can customize each Activity with widgets and switch between them easily using the Activities wheel.

control of Plasma Active. Drag the top panel down a little, and you will see a wide, touch-friendly task bar, with live previews of the running applications and the Home Screen that effectively minimizes all running programs. Drag a bit farther down, and you are presented with a wall of application icons, not unlike those provided on Android or Apple tablets, with a search box you can use to locate the correct application quickly. So far, not so very revolutionary.

Adapting to Different Tasks

It is the two tabs on the sides of the screen that are unique to Plasma Active. The tab on the right, when dragged out, reveals a wheel of Activities, each represented by a thumbnail image. A few selections are predefined, including "Introduction" that provides some information on getting started; "Vacation planning", set up for just that; and "My first activity", which invites you to make your own. You can delete, customize or

Try Plasma Active Two

The latest release of Plasma Active is easy to try out. If you already have a computer running MeeGo or OpenSUSE, you can install the needed packages. However, a safer and more convenient option is to try one of the ready-made live images—you always can install them if you decide you like Plasma Active. Live images on a MeeGo base are provided by basysKom, while open-slx provides an image built on its OpenSUSE-based Balsam Professional distribution.

You even can try Plasma Active on ARM devices (such as an Android tablet) using an image built on Mer, the port of a MeeGo-like system to the ARM architecture.

Details of all the installation and testing options can be found on the KDE Wiki (<http://community.KDE.org/Plasma/Active/Installation>).

The performance of the live images depends on your USB stick and your tablet device. You can install the software to get better performance, but that may, of course, overwrite your existing operating system.

As Plasma Active—and tablet devices—become more widespread, it is likely that many distributions will begin to offer Plasma Active as a user interface or provide special mobile-optimized distributions with Plasma Active.

As Aaron Seigo of KDE notes, “a solution only really matters if people can use it”. You also may one day see Plasma Active devices for sale—the Plasma Active team is “working quite hard on making this a reality in the near future”.

add activities using icons that are visible with the Activity wheel.

Selecting an Activity changes the desktop background and desktop widgets to those associated with the Activity. Choose the vacation planning Activity, and you are presented with a picture of a hay field as the desktop background and have the KDE weather forecast widget and bookmarks for OpenStreetMap, Wikitravel and a rail

operator on your desktop. You can open a browser and start booking your holiday, but if your child (or you) suddenly has an overpowering urge to play *Solitaire*, you can just switch to an Activity set up for games. If you check the task bar by pulling down the top panel, you will see that it shows only the applications from the current Activity, so your holiday booking in the vacation planning Activity is safe from little fingers accidentally

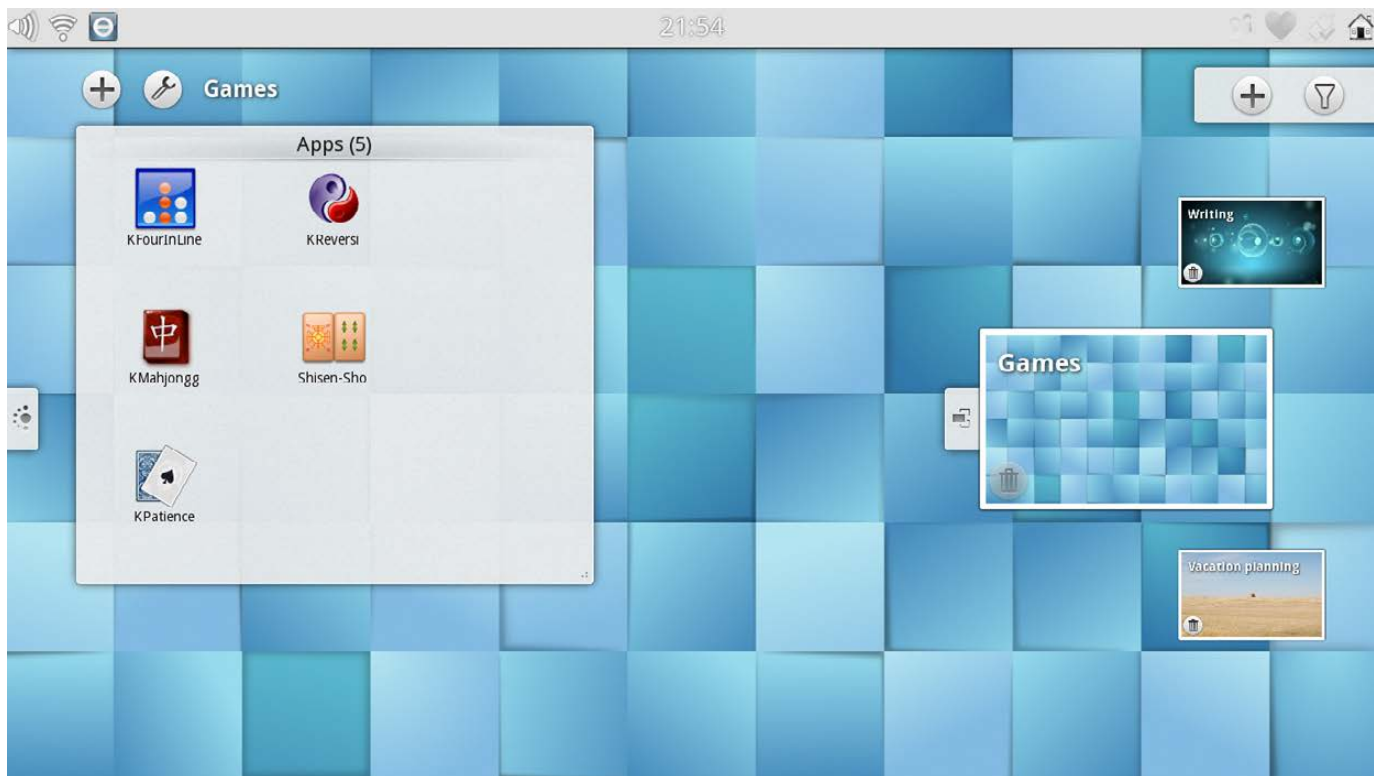


Figure 3. You can have an Activity for all the things you need to do—and for the things you want to do.

closing the browser or upgrading you all to first-class transatlantic travel. Once your child (or you) have had your game-playing fix, you can use the Activities wheel to get back to booking your holiday quickly.

Getting Smart with Nepomuk

The tab on the left of the screen reveals Plasma Active's "Recommendations": links to files, widgets and contacts that might be relevant to the current activity. This is based on Nepomuk, KDE's semantic storage technology, which draws links between items based on the context of their use. Marco explains that this enables "the information stored on the

device by users to be kept in a central place, allowing them to treat in the same way and display in a coherent way everything, regardless if it is a file, a contact, a bookmark or information about a location, linking them together with semantic information". What this means in practice is that the Recommendations are able to be more than a list of recently used or most-accessed files, suggesting documents that often are used at the same time as those presently open or often used within the current Activity (the recommendations are tailored to each Activity). While writing this article, it suggested some irrelevant items, but the system quickly learned to suggest

THE IDEA IS THAT THE SYSTEM LEARNS FROM ITS USER AND BECOMES EVER MORE USEFUL OVER TIME, AND BASED ON MY EXPERIENCE, I GIVE IT A CAUTIOUS THUMBS UP.

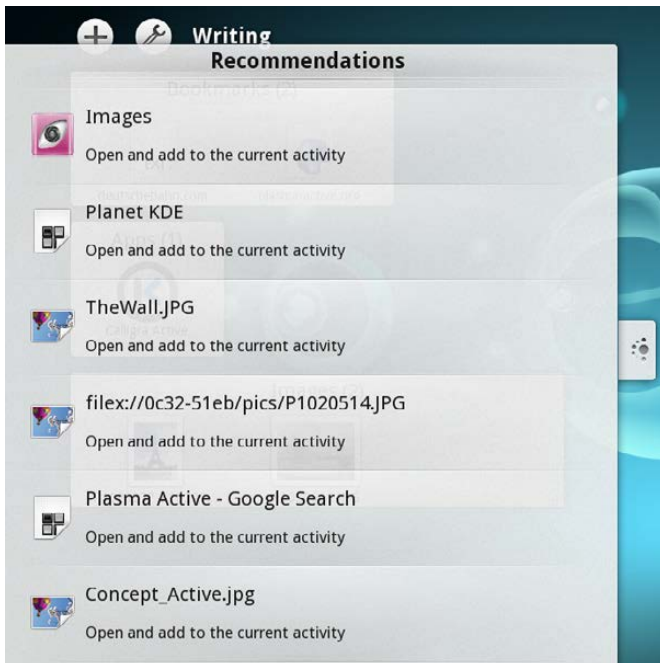


Figure 4. Plasma Active provides Recommendations of files and actions that are relevant to the task you are working on.

screenshots I had collected and suggested adding the *Linux Journal* author guidelines and Plasma Active Wiki pages to my bookmarks. The idea is that the system learns from its user and becomes ever more useful over time, and based on my experience, I give it a cautious thumbs-up.

Nepomuk has, since it was first introduced in KDE software in 2008, been the target of many complaints about resource usage, something that

is likely to be of even greater concern on a low-powered portable device. Marco, however, points out that “on a mobile device the stored data is very small compared to a desktop, and measurements have shown that with the limited number of items in it, the memory usage stays very small”, while, of course, using a central store also “avoids the necessity to build different storage/indexing for every application”. Aaron agrees: “the devices we currently target are all in the 600MHz to 1GHz range with 256MB or more of RAM. On these devices, it works acceptably”. Nevertheless, the developers have been working on “numerous optimizations and improvements”, and Aaron acknowledges there is always the possibility of “stripping out Nepomuk for some very low-end, in terms of hardware and user interaction, scenarios”. I did not come across any of the slowness that sometimes accompanies extensive indexing on the desktop.

It's All about the Apps

The basic user interface seems slick and well thought out, and through Activities,

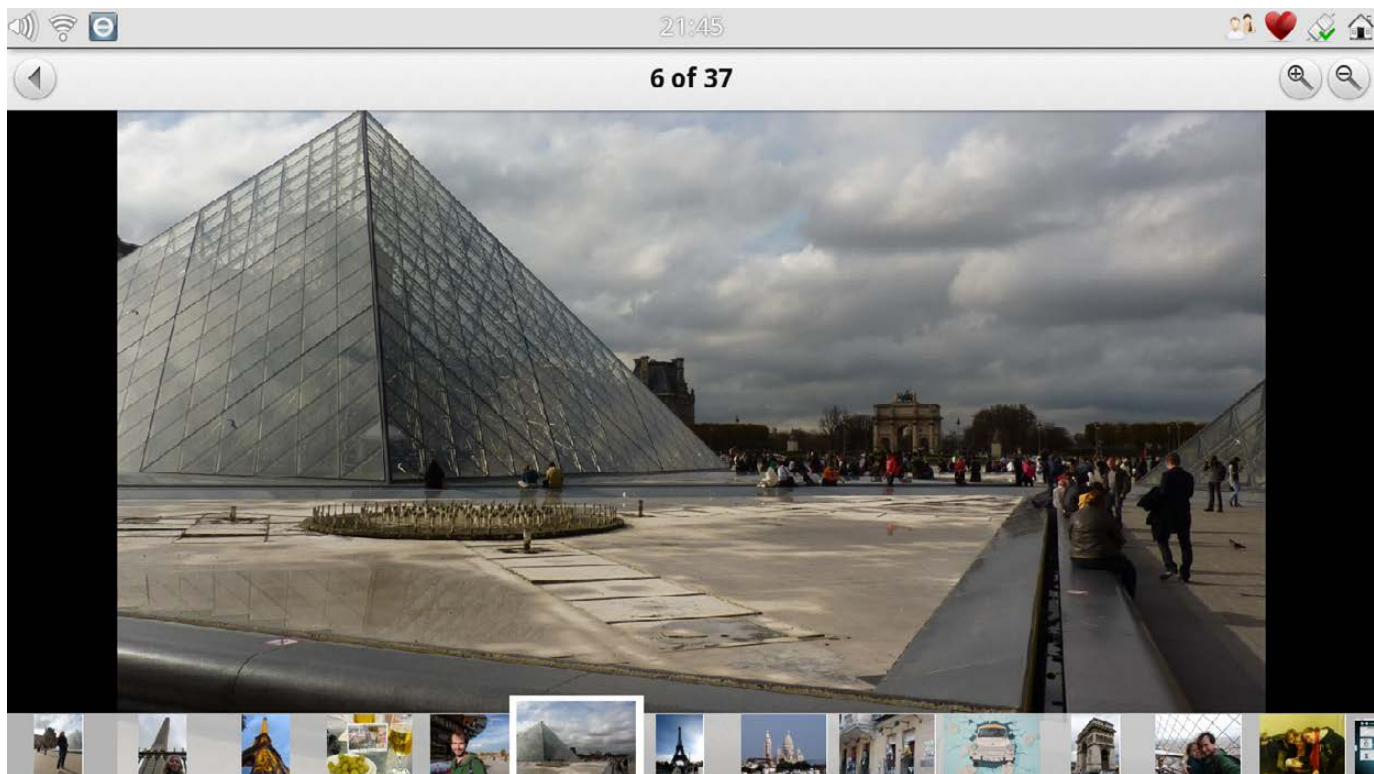


Figure 5. The applications that already have been adjusted for use with Plasma Active, such as the image viewer, work well.

it does offer something different from the competition. However, a computer is only as good as its applications, and if Plasma Active is to be a success, KDE must provide a compelling suite of touch-friendly applications.

A few "Active" variants of established KDE applications already are available. These include those that have been largely designed from the ground up for Plasma Active, such as the Web browser and image viewer, both of which were easy to use. Some other applications, such as the media player Bangarang, have received modifications to make them a little more touch-friendly.

There are dedicated Active versions of the Kontact suite of groupware applications. Each of these are easy to use with a stubby finger, but their interfaces are so different from their desktop counterparts that even if you are an experienced Kontact user, you will find they take a little getting used to. Calligra, KDE's productivity suite, also is available in an Active version, but it felt slow on the device I used. However, Calligra's underlying technology already has been used in the successful FreOffice viewers for Nokia's mobile phones, so it is likely the performance will improve.

Some other applications, such as

FEATURE Plasma Active—a New Approach to Tablet Computing

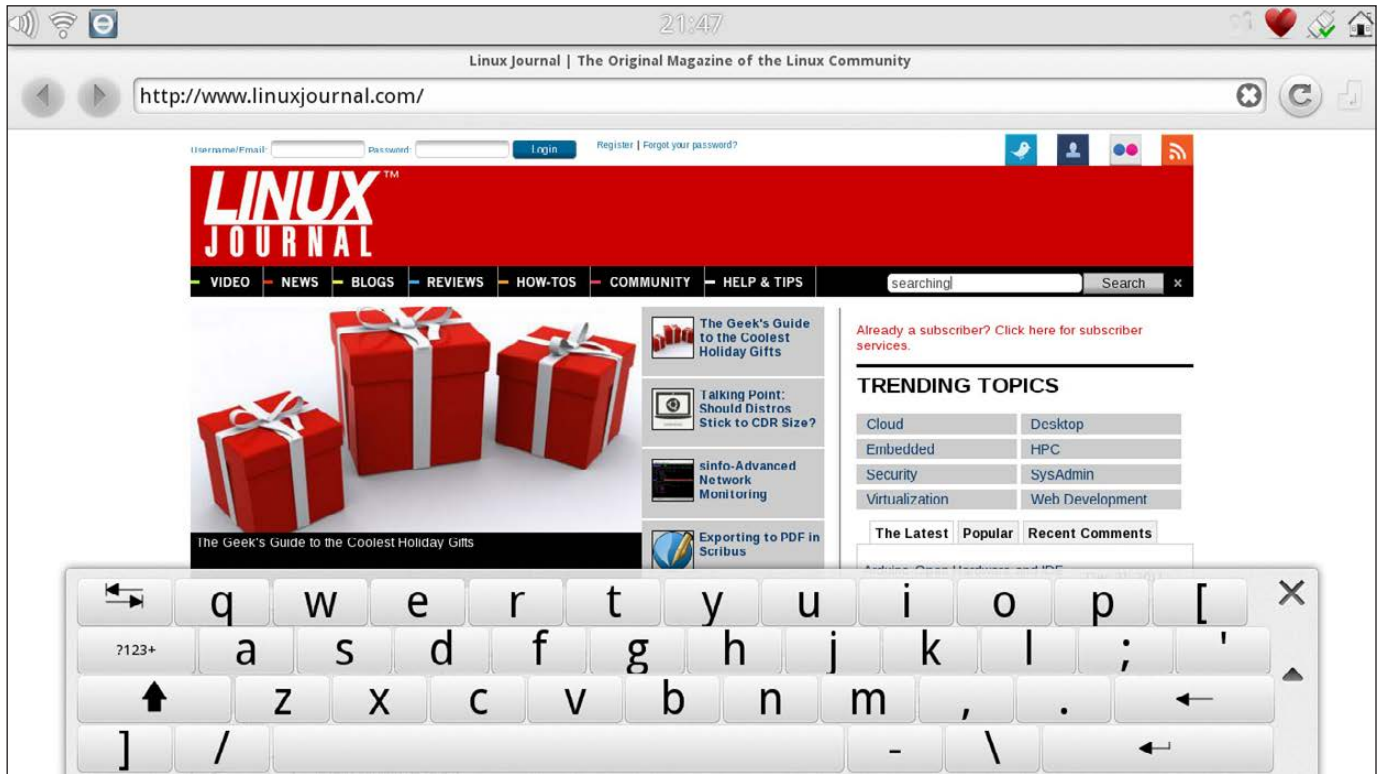


Figure 6. The on-screen keyboard is easy to use and can be moved out of the way when needed.

Dolphin (the KDE file manager), have not been adapted for touch-friendly use—and it shows. A similar interface is used in the Open and Save dialogs of most applications, but these will all be improved in future versions.

The most essential of applications on a tablet, the on-screen keyboard, works very well with easy-to-touch buttons and a sensible layout. It appears when needed, and it can be switched from the bottom to the top of the screen if desired.

Share, Like, Connect

Another new feature in Plasma Active is the presence of Share, Like and Connect buttons in the top panel. These

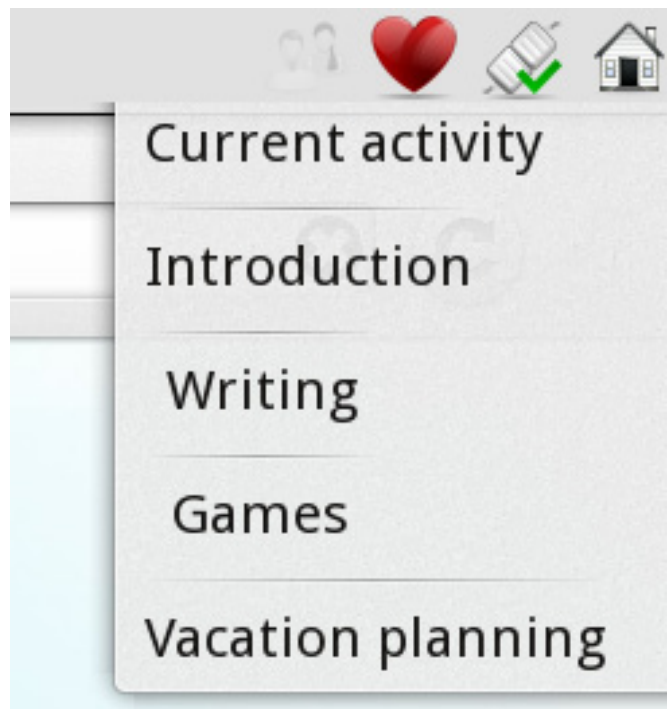


Figure 7. You easily can associate a file or Web page with an Activity by using the Connect button.

make it easy to share content instantly, such as images, to social networks or on-line storage; “like” things on either social networks or locally (for example, bookmarking a page); and connect things together, such as linking a document or a page to the current Activity so that it always will be readily available in the future. This way, if you want quick access to an image, just click the Connect icon to add it as a widget on the desktop of the present Activity.

Open for Business

Plasma Active has been unusual among KDE projects due to the heavy involvement of companies from the start. Among these, basysKom has employed developers to work on Contour, the combination of Activities and Recommendations at the center of the user experience. Marco states that “everyone from the community is welcome to join and contribute, just like any other KDE project—companies are members of the community as well and are helping in many tasks (also the less fun ones in order to give to it the level of quality needed for an actual product)”. Companies also are making it easy to try Plasma Active, with live images provided by basysKom and open-slx (see the Try Plasma Active Two sidebar).

Life after MeeGo

In the past, a large part of KDE’s

focus in the mobile space has been on MeeGo-powered devices, particularly those created by Nokia. However, the decision of Nokia to use Windows as the base for its smartphones and Intel’s subsequent dropping of MeeGo in favor of its collaboration with Samsung, Tizen, has changed things. These changes do not, however, unduly concern the KDE developers. Aaron points out that “Plasma Active is not welded to any one OS and is highly portable”, and indeed, there are “images with OpenSUSE, MeeGo and Mer kernels and userlands beneath the Plasma Active UI”.

ARM devices running Android have proven to be very popular, and Plasma Active also is targeting some of this hardware, with an ARM-ready image already available built on software from the Mer Project. But what about Android itself? It is not presently the most appealing alternative for Marco, who argues that “Android, while released with an open-source license is tightly controlled by Google and doesn’t leave much room for a developer community to grow and contribute”. He does, however, acknowledge that Android “is a good platform indeed and we don’t exclude some integration work with it in the future”.

There also are possibilities of Plasma Active or related technologies targeting much more than just tablets. Aaron notes that already

FEATURE Plasma Active—a New Approach to Tablet Computing

“some are running it on handset-style devices”, but that “the current user experience has been designed with a tablet in mind”. He plans, however, to start working on “interfaces that are designed specifically for other form factors, such as set-top boxes and handsets in the future”. A key enabler for this is Plasma’s design: “Plasma allows for multiple, and highly diverse, user interfaces without starting from scratch. Plasma Desktop, Netbook and now Active for tablets showcases this very nicely: they are all very different on the surface from each other, but share nearly all the implementation code beneath.”

The Future of Plasma Active

Plasma Active is still young software. Plasma Active Two was released shortly before this article was written and is the version discussed here. Plasma Active Three is expected in summer 2012 and “will be focusing on new major feature and application introductions”, according to Aaron. From a purely end-user perspective, the limited number of touch-friendly applications means Plasma Active is not ready yet. Nonetheless, it is well worth trying out and could become compelling by the time of its third release later this year. It already feels more polished and complete than any of the MeeGo tablet pre-releases.

There are other reasons to get

excited about Plasma Active. For Marco, the motivation for starting work on Plasma Active had “different reasons, both purely technological and social ones”. The social ones are perhaps best summed up by Aaron: “right now, there is too much focus on created devices that serve the owner of the application store and focus on consumption of new devices just for the sake of the newness of the device”, something he believes has “largely stalled progress”.

Aaron sees a different future for Plasma Active and those who choose to contribute to or use the software: “We should be looking at how to support people’s lives and in doing so make them better. This needs to be done in a socially responsible manner, which means free and open-source software as well as open processes must drive the development. This is the point and purpose of Plasma Active.” ■

Stuart Jarvis is a technology writer and member of the KDE community. He has had a tablet computer for the past six months and has been trying very hard to find a use for it.

Resources

Plasma Active: <http://plasma-active.org>

Plasma Active Installation:
<http://community.KDE.org/Plasma/Active/Installation>

O'REILLY

where CONFERENCE

THE BUSINESS OF LOCATION

APRIL 2-4
SAN FRANCISCO 12



Mobile changes everywhere.

The explosion of mobile technologies has put location-aware devices everywhere. The mobile landscape is constantly evolving and making smart decisions about buying or building the right tools isn't easy.

Get a handle on what's what with location at the O'Reilly Where Conference. Learn from leading-edge developers, business leaders, marketers, and entrepreneurs what's happening how and what's lurking just below the radar—and how you can leverage location for your business.

Program Tracks

Mobile Development, Location Development, Business & Strategy, Marketing—and the return of the Location Marketing Boot Camp.

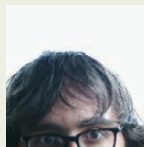
Featured Speakers



Thomas Goetz
Wired



Charlene Li
Altimeter Group



Jer Thorp
New York Times



Brian McClendon
Google



Noah Iliinsky
Complex Diagrams



Leah Busque
TaskRabbit

**Save 15% on registration
with code WHR12LJOU**

O'REILLY®

Seamlessly Extending **IRC** to **Mobile Devices**

IRC is still popular among techies, but it's not friendly to mobile devices. Here are some tips on making it more effective when you're on the run.

BILL CHILDERS

INTERNET RELAY CHAT (IRC)

is one of the older real-time communications methods still in active use on the Internet. Due to its popularity, flexibility and cross-platform nature, it still has a

very vibrant user base today. Personally, I've been on IRC since the late 1990s, and it's been very useful and lots of fun—particularly the #linuxjournal room on Freenode—stop in sometime!

Drawbacks to IRC

As great as IRC can be, there's one thing about it that's always bothered me, and it's something that Jabber got right—the concept of a resource and priority. In IRC, when you log in as whatever user you are, you can't log in as that user again from another machine. Jabber allows multiple user logins, so long as the "resource" is different, and it'll route your messages to the client with the lowest priority. IRC doesn't even have that concept, so once you log in on your desktop, you have to log in as another user if you want to log in on your laptop, which results in lots of logins like "WildBill|Laptop" and such. This causes a problem if people want to private-message (PM) you, as they never know what login you're on.

For years, I've used a fairly common workaround to bypass this problem to a certain degree. If you use a text-based client like Irssi, and run that client on a machine that's on all the time, you can run the client within a terminal multiplexer, such as GNU Screen. That gives you the ability to run the client, detach from the terminal session, then log in via SSH on another machine and reattach to your session, where you can catch up on what you may have missed. By and large, this is an acceptable workaround...except when I forget to log in to that Screen session. I've been

known to have a few days' worth of PMs piled up before I remember to log in and check. By that time, people who PM'd me probably forgot what they pinged me about.

The other major flaw with this scheme is that it doesn't work well with mobile devices like tablets and mobile phones. Yes, I could SSH from my iPhone to my Linux server and attach to that Screen session with Irssi running in it, but a terminal emulator on a 3.5" screen, where a good portion of the screen is taken up with an on-screen keyboard, is a recipe for a headache. There are excellent IRC clients for mobile phones, like Colloquy on the iPhone, Yaaic for Android and wIRC for WebOS, but if you run them while you have a Screen session going, you're back to the same problem mentioned above where you wind up with a "WildBill|Phone" login. There simply *had* to be a better way.

Is There a Better Way?

It turns out, after noodling around for a while one day, I did find a way. Irssi, the text-mode client I usually use via Screen and SSH, has a "proxy" (or "bouncer") mode where you can have it listen on a couple additional ports and then attach another IRC client to it. Enabling the proxy is relatively easy, and it can be done from within Irssi with only a few commands (assuming your Irssi is packaged and includes the

FEATURE Seamlessly Extending IRC to Mobile Devices

proxy module). This also assumes you have Irssi already configured to work properly with whichever IRC networks you wish. Of course, substitute your own password for “mypassword”, define whatever IRC networks you want to proxy with Irssi, and pick an arbitrary open port on which to have the proxy listen:

```
/LOAD proxy
/SET irssiproxy_password mypassword
/SET irssiproxy_ports linuxjournal=9000
/SAVE
```

Once you get the proxy module running successfully, connecting to it is as easy as pointing another IRC client at it. Just specify the fully qualified domain name of the host running your Irssi client in your second IRC client’s configuration, and give it the password and ports you called out in the Irssi commands above. This client can be a “regular” desktop or laptop, or it can be a mobile device, such as an Android phone or an iPad running an IRC client.

Although this works just fine, it bothers me a little bit, as the password that’s sent between the mobile device and your Irssi session that’s running in Screen is sent in the clear. I’d much prefer that to be SSL-encrypted, so no one can intercept that password. Unfortunately, the Irssi proxy module doesn’t support SSL, but there’s a way around that through the use of the

stunnel utility.

Stunnel is a generic encryption wrapper that’s designed to add SSL encryption to any non-encrypted service. It’s not hard to wrap the Irssi proxy service with stunnel. First, to prevent anyone on the outside from accessing the proxy without SSL, I bound the ports on the Irssi proxy to the loopback interface using the following Irssi command:

```
/SET irssiproxy_bind 127.0.0.1
/SAVE
```

Next, use your distribution’s method for installing stunnel. On Ubuntu 10.04, a simple `sudo apt-get install stunnel4` took care of that. I had to create a self-signed SSL certificate (see Resources for a how-to on this), and I put that cert in `/etc/stunnel`. Next, I had to create an `/etc/stunnel/stunnel.conf` that referred to the certificate I created and specified the Irssi proxy in the `stunnel.conf` file. An example follows below (adjust the file paths, “accept” IP address and port as necessary):

```
; stunnel.conf code snippet
; Certificate/key is needed in server mode and optional in client mode
cert = /etc/stunnel/cert.pem
key = /etc/stunnel/key.pem

[linuxjournal]
accept=123.123.123.123:9000
connect=127.0.0.1:9000
```


So, what the previous code snippet does is specify the cert/key file for the self-signed SSL cert, then binds the SSL side of stunnel to the external IP address 123.123.123.123 on port 9000, and it'll hand off packets to my Irssi proxy running in the clear on 127.0.0.1 (localhost) on port 9000. The last adjustment to this setup would be to go to the mobile device and specify that the IRC connection should be SSL, and now I have a secure proxy I can attach to with any client of my choice. Now I can log in to IRC from my regular Irssi client, but then when I'm on my phone, I can use the friendlier interface that Colloquy provides, all without having different nicknames on IRC.

Being "Pinged" and Getting Alerted

I still have a problem though. I still forget to log in to IRC and check my private messages—and some friends stack up many "pings" before they finally give up in frustration. I got to thinking and realized that since I'm on a mobile phone, it might be possible to send an e-mail to the SMS gateway my mobile provider runs when I get a PM. That'd give me a reminder to log in to IRC in the event that I forget. After searching the Internet for a bit, I discovered some other fellow had much the same idea. Michael Lustfield explains it on his blog in far more detail than I

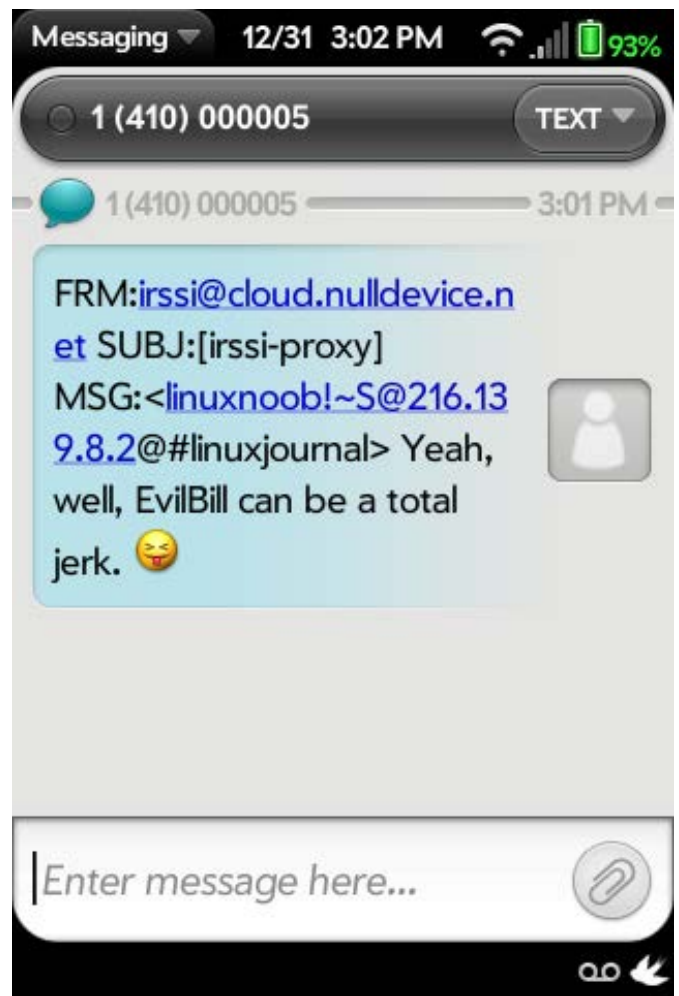


Figure 1. Getting an SMS on WebOS

can put in this article (there's a link in the Resources section), but I'll summarize his technique here. He's got two Irssi scripts he uses: `screen-away.pl` monitors if there's an active connection to the Irssi proxy, and `awayproxy.pl` sends any highlights off to a designated e-mail address if there isn't an active client connected to the Irssi proxy.

"But Bill...e-mail isn't an SMS!" You may think that, but just about every mobile carrier can accept SMS messages

FEATURE Seamlessly Extending IRC to Mobile Devices

for delivery to a mobile phone via e-mail. In my case, I use AT&T, so any e-mail sent to <myphonenumber>@txt.att.net will arrive as a text message on my phone. So, all I have to do is drop Michael's scripts in my Irssi scripts folder, adjust awayproxy.pl so that the \$config{mailto} variable is mymobilenumber@txt.att.net, and then activate both scripts from within Irssi by doing a /script load awayproxy.pl and /script

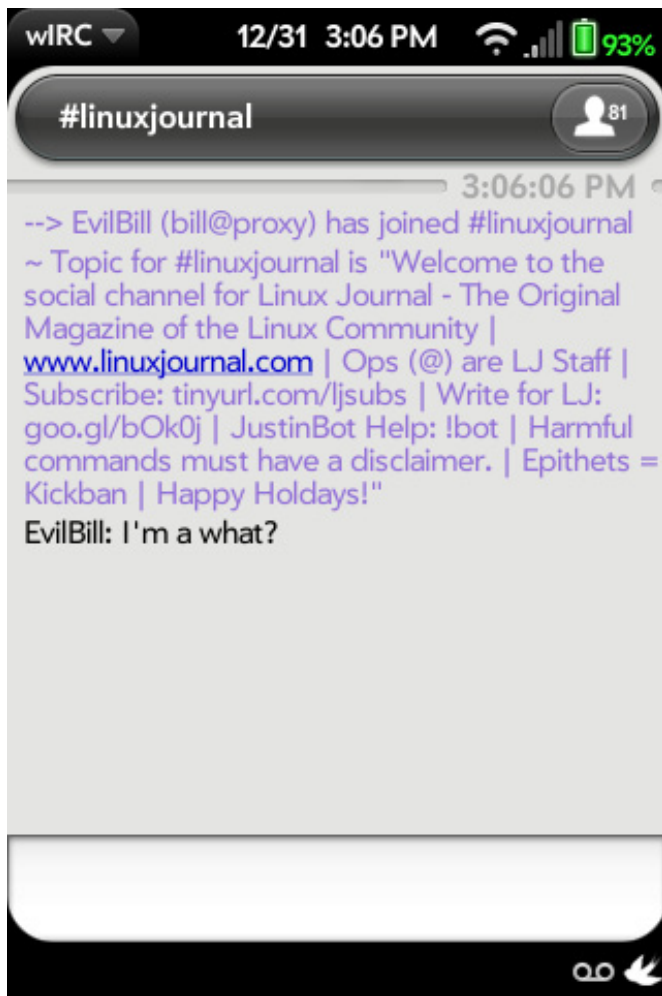


Figure 2. Jumping into the Conversation—Using wIRC on WebOS

load screen-away.pl. Now, any mention of my name in any of the channels I sit in will cause an SMS to be fired off to my phone, and I can log in from my phone or tablet and jump into the conversation.

We Can Rebuild It, We Have the Technology...

This solution, although totally usable, isn't quite as elegant as I'd like. My "main" tablet and phone are both iOS devices (iPad and iPhone, but let's not



Figure 3. Getting a Push Notify on an iPhone

discuss my choice of mobile devices here), and Apple has a very clean, integrated push notification system. I started poking around to see if there was any way to send push messages to my iDevices rather than rely on SMS—that way my iPad would receive IRC “pings” as well.

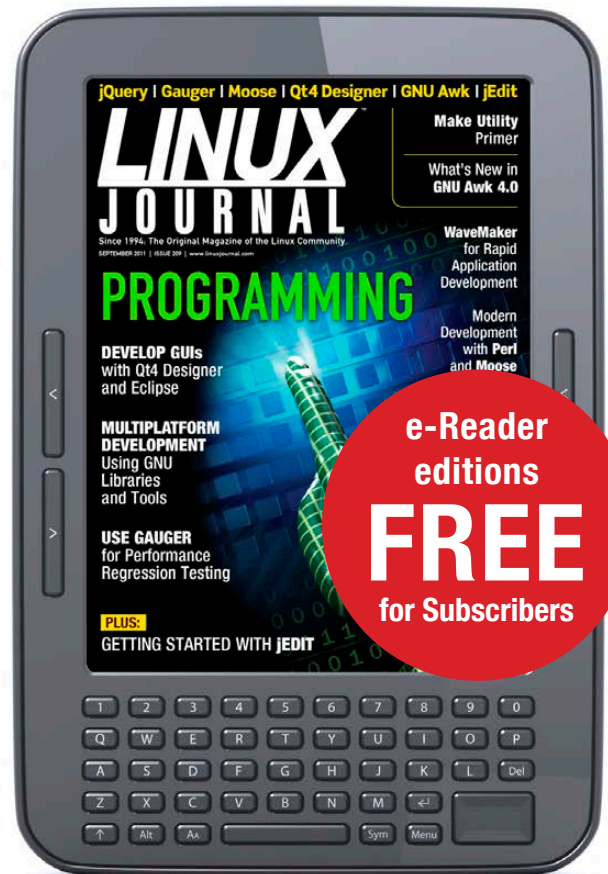
It turns out, the Internet already had an answer for this problem as well. A fellow named Chris Jones already put together a patched version of Irssi and an Irssi script that does



Figure 4. Looking at My Previous Notifications

LINUX JOURNAL

on your
e-Reader



Customized
Kindle and Nook
editions
now available

[LEARN MORE](#)

FEATURE Seamlessly Extending IRC to Mobile Devices

all the heavy lifting for this solution. His Web page (see Resources) talks all about the details on this and how to install it. He even mentions using Irssi with stunnel!

In closing, utilizing this set of scripts has really brought IRC into the 21st century for me. I love the idea of IRC, but it shows a lack of mobility features due to its age. With this set of scripts and the

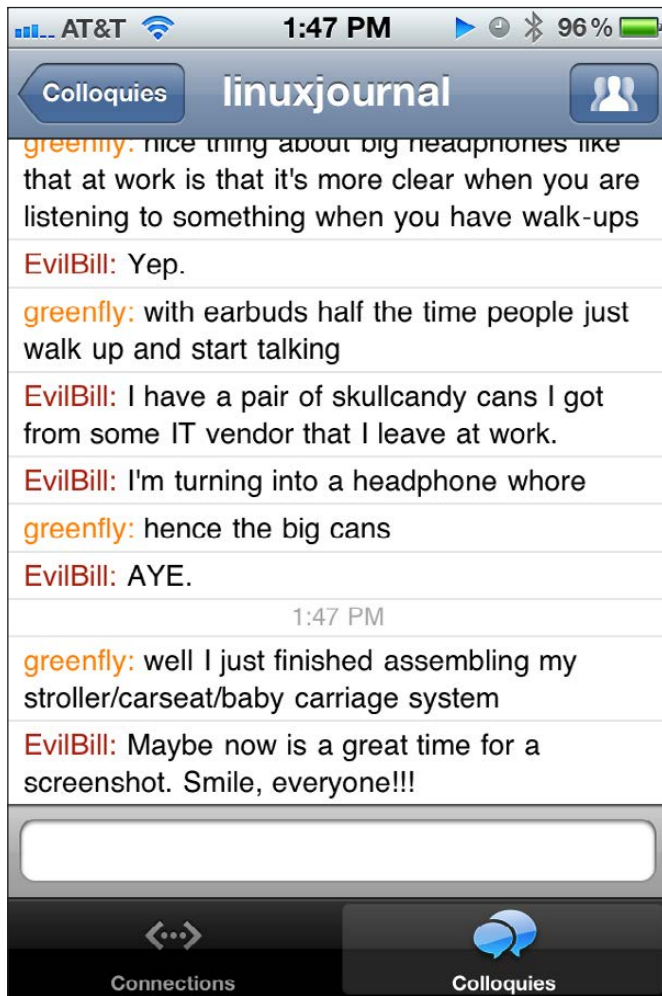


Figure 5. Firing Up Colloquy to Join the Conversation

right mobile tools, you can bring your IRC addiction up to date—and have it with you, anywhere you go! ■

Bill Childers is an IT Manager in Silicon Valley, where he lives with his wife and two children. He enjoys Linux far too much, and probably should get more sun from time to time.

Resources

Irssi Proxy Documentation:
<http://www.irssi.org/documentation/proxy>

Michael Lustfield's Irssi-to-SMS Plugin Use:
<http://michael.lustfield.net/content/irssi-sms>

Chris Jones' Irssi Proxy and iPhone:
<http://www.tenshu.net/2010/12/old-and-new-mixing-irssi-and-iphones.html>

Stunnel Home Page: <http://www.stunnel.org>

Generating a Self-Signed SSL Certificate: http://www.akadia.com/services/ssh_test_certificate.html

"Internet Relay Chat" by Jayson Broughton:
<http://www.linuxjournal.com/content/internet-relay-chat>








"IRC, Still the Best Support Around" by Shawn Powers: <http://www.linuxjournal.com/content/irc-still-best-support-around>

Linux Journal IRC Channel: #linuxjournal on Freenode



LinuxFest Northwest

Bellingham, WA
April 28th & 29th

- Grassroots Linux gathering 
- Exhibits of all flavors 
- Presentations of all levels 
- Prizes and after party 
- FREE admission & parking 
- FREE open source software 
- Bring the whole family! 

Hosted By



linuxfestnorthwest.org

Using Linux with EFI, Part III: Installing Linux on an EFI Computer

This installment covers booting in EFI mode, quirks of different distributions in EFI mode and converting a BIOS-based installation to boot in EFI mode.

RODERICK W. SMITH

In Part II of this series in the January 2012 issue, I described the prerequisites for installing Linux on a computer that uses an Extensible Firmware Interface (EFI) or Unified EFI (UEFI) as opposed to a more traditional Basic Input/Output System (BIOS) firmware. If you're lucky, you might be able to forego some of these preparatory steps, because your distribution's installer might boot properly and manage the steps correctly. All too often though, Linux distributions handle these tasks poorly enough that you'll want to prepare your computer first by

partitioning manually and determining how to boot your computer in UEFI mode for installation—or in BIOS mode, in some cases.

In this article, I look at the nuts and bolts of installing Linux on an EFI-based computer. The topics I cover are booting in EFI mode; quirks of Debian, Ubuntu, Fedora and OpenSUSE in EFI mode; and converting a BIOS-based installation to boot in EFI mode. My next article will conclude this series with a look at the tools and techniques you can use to troubleshoot and keep an EFI-based Linux installation working correctly.

I've heard of some implementations that provide such poor control that the only way to switch modes is to employ tricks like rebuilding the installation medium to omit the files for the mode you don't want to use!

Booting in EFI Mode

In most cases, launching a Linux installer on an EFI-based computer is just like launching the same installer on a BIOS-based computer. You insert the disc, boot the computer and perhaps select a boot-time option to boot from the optical disc rather than from a hard disk. On most UEFI-based PCs, you must be sure to use a 64-bit version of your distribution; the 32-bit versions usually lack EFI boot files, and such files generally are useful only on early Intel-based Macs in any event.

Unfortunately, the fact that most EFI implementations provide a BIOS compatibility mode (also known as a Compatibility Support Module, or CSM) can make it hard to control whether your computer boots using EFI or BIOS mode. I've heard of some implementations that provide such poor control that the only way to switch modes is to employ tricks like rebuilding the installation medium to omit the files for the mode you don't want to use! In most cases though, firmware boot

options provide some way to launch the disc in one mode or another (as described in Part II of this series).

Some distributions, such as Debian, don't yet ship with any EFI boot support, at least not for the x86-64 platform. With such distributions, you have three choices:

- You can install Linux in BIOS mode and continue booting in BIOS mode, ignoring the computer's EFI features.
- You can install Linux in BIOS mode and then convert the working installation to boot in EFI mode, if this is necessary or desirable.
- You can install an EFI bootloader on your hard disk or installation medium and use that to boot the installer in EFI mode. (Part II of this series covered EFI bootloader installation.)

Which method works best depends on your computer and your needs. For instance, BIOS-mode booting is

usually fine for a Linux-only installation, although it's a bit slower than a native EFI boot. You even can use the GUID Partition Table (GPT) on your disks, if necessary. If you have a working EFI-mode Windows installation though, switching between EFI- and BIOS-mode boots for the two OSes may be awkward.

If you're not sure whether your distribution supports EFI-mode boot, you can check the files on the installation disc. Discs that support EFI-mode boots typically include files with names that end in "efi", such as `boot/x86_64/efi` (OpenSUSE) or `EFI/BOOT/BOOTX64.efi` (Fedora or, in all lowercase, Ubuntu). In actuality, EFI-bootable discs use a secondary El Torito entry, so these files might not be present in the main ISO-9660 filesystem, but they usually are. If you don't see such files, you can try booting the disc, but if it boots, you might be left with the question of whether it booted in BIOS mode or in EFI mode. To make this determination, try this:

1. Boot the installation disc.
2. Switch to a text-mode console by pressing `Ctrl-Alt-F2` (or some other function key, if `F2` doesn't work).
3. Type `dmesg | grep EFI`.

If the `dmesg` command produces no output, or perhaps just one or two

lines, you've booted in BIOS mode. If the `dmesg` output includes numerous lines, most of which relate to memory mappings, you've booted in EFI mode. An example EFI boot might include lines like these (the full output is much longer than this):

```
[ 0.000000] Command Line: BOOT_IMAGE=
                atapi2:\EFI\ELILO\bzImage-3.0.4
                root=/dev/seeker/u1104 ro
[ 0.000000] EFI v2.00 by American Megatrends
[ 0.000000] Kernel-defined memdesc doesn't match
                the one from EFI!
[ 0.000000] EFI: mem00: type=3, attr=0xf, range=
                [0x0000000000000000-0x0000000000048000)
                (0MB)
```

Once you've verified that you've booted in EFI mode, you can press `Alt-F7` (or `Alt-F1` in Fedora) to return to the installer and proceed with installation. Alternatively, on some systems, it may be easier to install Linux in BIOS mode and then install an EFI-capable bootloader to enable the computer to boot in that way. Such a switch is especially likely to be useful with Debian and Ubuntu installs, as I describe shortly.

Debian's EFI Support

As of version 6.0.3, you can install Debian on an EFI system in EFI mode, but only by adding your own EFI bootloader; the standard install DVD image lacks this critical feature. To do a native EFI installation, you can install ELILO (or

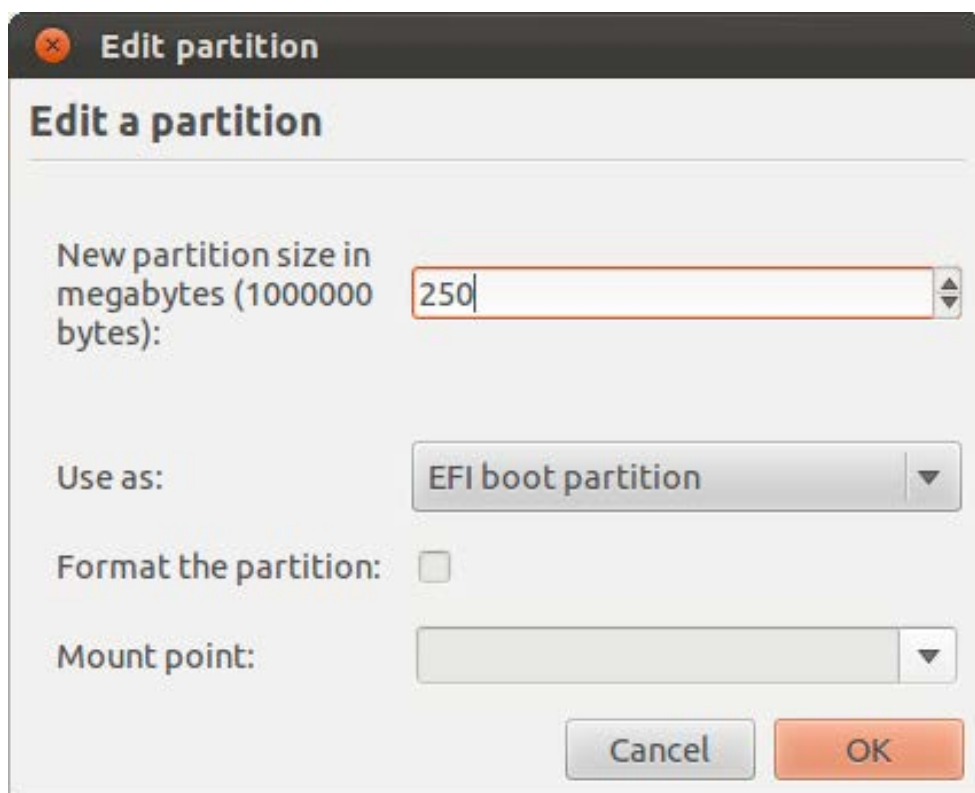


Figure 1. Ubuntu identifies the ESP as an “EFI boot partition”.

another bootloader, if you prefer) to the computer’s EFI System Partition (ESP), as described in Part II of this series. You must copy the `vmlinuz` and `initrd.gz` files from the disc’s `install.amd` subdirectory (assuming an x86-64 installation) as the kernel and initial RAM disk files. You can omit the `root=` line in the `elilo.conf` file.

Unfortunately, the reward for making this extra effort is an installer that runs but has some serious bugs, which are identical to those for Ubuntu. I describe them shortly in the Ubuntu section, because Ubuntu officially supports booting its standard installation discs in EFI mode. In addition, the Debian installer installs a BIOS version of GRUB,

so you’ll need to install an EFI-capable bootloader manually when you’re done.

Instead of installing Debian in EFI mode, it’s usually easier and safer, particularly for multiboot systems, to install the OS in BIOS mode, install an EFI-enabled bootloader, and then switch the computer into EFI boot mode to use the computer in this way. I cover this procedure in more detail later.

Debian’s repositories include ELILO (elilo), the EFI-enabled version of GRUB 2 (grub-efi) and rEFIt (refit), so you can install any of these via packages. However, you will need to copy the relevant files to the ESP, as described in Part II of this series.

Ubuntu’s EFI Support

As of version 11.10, Ubuntu can install directly in EFI mode, and on a single-boot computer, such an installation can work fairly well. One important caveat exists: if you use the advanced partitioning option (called “Something Else” on the “Installation Type” screen), you must identify the ESP as such by selecting the ESP, clicking Change and

selecting “EFI Boot Partition” in the “Use As” field, as shown in Figure 1. If you don’t identify the ESP in this way, the installer won’t install the bootloader to the ESP, rendering the system unbootable. If this happens, you can install a bootloader using an emergency boot system.

Another problem is that if you don’t create an ESP before you begin the installation, Ubuntu creates one that’s on the small side—often well under 100 MiB, depending on your hard disk’s size. Such a small ESP is fine if you use Ubuntu’s GRUB 2 bootloader, but not if you use ELILO. Thus, I recommend creating an ESP before you begin the Ubuntu installation process.

More serious complications arise in a multiboot environment. Specifically, when you properly identify the ESP, Ubuntu creates a fresh FAT-16 filesystem on the ESP. This has two undesirable consequences:

- If you install Ubuntu after another OS, the Ubuntu installation process destroys your existing OS’s bootloader.
- If you install Ubuntu before Windows 7, the Windows installer may not like the fact that the ESP uses FAT-16—Windows 7 is a stickler for the ESP being FAT-32. The result is that Windows either will complain that there’s no ESP or it will create a new ESP, proceed most of the way through

the installation and become confused late in the process. Either way, the result is a failed Windows installation.

Several workarounds to this problem exist, such as:

- You can install Windows (or any other OS) first, back up the ESP, install Ubuntu, and then restore the backed-up ESP files atop the Ubuntu ESP files. This procedure works well, but it leaves the ESP as a FAT-16 partition, which could cause problems if you need to re-install Windows in the future.
- You can install Ubuntu first, back up its ESP, create a fresh FAT-32 filesystem on the ESP, restore the ESP files, edit `/etc/fstab` to mount the ESP using its new UUID value, and then install your other OSes.
- You can install Ubuntu and deliberately *not* identify the ESP, or you can install it in BIOS mode rather than in EFI mode. Either way, this will leave the ESP untouched. You then will have to install a bootloader manually.
- You can prepare a FAT-32 ESP, and when you install Ubuntu, deliberately *not* identify the ESP. Once you’ve passed the partitioning screen, press `Ctrl-Alt-F1` to open a

text-mode terminal and manually mount the ESP at `/target/boot/efi`. The installer then will add Ubuntu's GRUB 2 to the ESP but leave its existing contents untouched. In many ways, this is the best solution, but it requires that you be prompt in manually mounting the ESP.

By default, Ubuntu uses GRUB 2 as its EFI bootloader, installed from a package called `grub-efi`. Ubuntu also provides `elilo` and `refit` packages, should you prefer one of these bootloaders; however, Ubuntu's package management system includes scripts to update the GRUB 2 configuration when you upgrade a kernel, and these scripts won't update an ELILO configuration. Thus, if you want the easiest configuration, GRUB 2 has an edge. On the other hand, GRUB 2 is less reliable than ELILO, in my experience. Overall, I recommend you try GRUB 2, and if it gives you problems, switch to ELILO (with `rEFIt`, if necessary, to select a non-Linux OS).

Fedora's EFI Support

Fedora 16 provides the most complete and reliable EFI support during installation of any of the distributions described here. Unlike Ubuntu, Fedora doesn't replace a working ESP with a new one; it adds its own files to the working ESP, as a well-behaved EFI OS installer should. Fedora does,

however, have a tendency to create ESPs that are on the large side—often well over a gigabyte. Also, if you install in BIOS mode to a GPT disk, Fedora mis-labels its `/boot` partition (or the Fedora root, `/`, partition if you don't create a separate `/boot` partition) as an ESP. This last bug doesn't affect native EFI installations.

Fedora uses a modified version of GRUB Legacy on its EFI installations. (Fedora 16 has changed to GRUB 2 for BIOS installations, but has stuck with its modified GRUB Legacy for EFI.) This configuration works well on most systems in my experience.

OpenSUSE's EFI Support

OpenSUSE 12.1 supports EFI installation, although the installer creates a new ESP even if a valid one already exists. What's more, the ESP that OpenSUSE creates is a bit small (156 MiB) and is FAT-16. As noted with respect to Ubuntu, a FAT-16 ESP can cause problems if you subsequently attempt to install Windows on the computer. You can work around some of these problems by using the manual partitioning option, but that creates other problems if you're starting from scratch. For instance, the manual partitioner sets the wrong type codes on most of the partitions it creates, and you can only force it to create a FAT-32 filesystem by making the partition bigger than about 520 MiB.

GPT on BIOS

Some EFI-capable firmwares have problems booting from GPT disks in BIOS mode because of BIOS bugs. You often can overcome this problem by setting the boot flag (aka the active flag) on the 0xEE partition in the disk's *protective MBR*, which is a part of the disk that's intended to keep older GPT-unaware utilities from messing with a GPT disk. To do this, use Linux's `fdisk` utility:

1. Type `fdisk /dev/sda` to launch `fdisk`.
2. Type `a` followed by `1` for the partition number. (The 0xEE partition is almost always partition #1.)
3. Type `w` to save your changes.

Thereafter, the computer should boot fine in BIOS mode from the disk. Be aware that some GPT partitioning tools will undo this change, so you may need to re-enable it. Note that you should *not* attempt to use GNU Parted, GParted or any other libparted-based tool to do this; they won't work.

Therefore, if you need to dual-boot with Windows, it's probably best to prepare your partitions before you run the OpenSUSE installer and then use the manual partitioning option to identify each partition to the installer. (You can identify the ESP by telling OpenSUSE to mount it at `/boot/efi`.)

OpenSUSE uses ELILO as its default EFI bootloader. ELILO generally works well, in my experience. If you dual-boot with another OS, though, you'll need to replace or supplement ELILO, since it doesn't support chainloading to another OS's bootloader.

Converting an Existing System to Boot with EFI

Because of the bugs in so many distributions' installers' EFI support, it's often better to install the OS in BIOS mode and then convert it to use an EFI bootloader than it is to install directly in EFI mode. You may need to perform such a conversion if you have a working system that you want to begin booting in EFI mode. Fortunately, Linux can switch between BIOS and EFI booting with no problems, provided you have both BIOS and EFI bootloaders installed. (If

you're converting permanently, you can uninstall the BIOS bootloader when you're done with the conversion.)

If you're starting from scratch, the easiest way to proceed is as follows:

1. Partition your disk as described in Part II, with a 200–300 MiB ESP and whatever partitions you need for Linux. If your distribution uses GRUB 2, you also should create a 1 MiB BIOS Boot Partition, which is identified by a `bios_grub` flag in parted or GParted or by a type code of EF02 in GPT fdisk (`gdisk`, `cgdisk` or `sgdisk`).
2. Boot the Linux installer and verify that it's running in BIOS mode.
3. Install Linux normally. The result should boot in BIOS mode, but not in EFI mode.
4. Mount your ESP at `/boot/efi`, creating that mountpoint if necessary. You also should create an `/etc/fstab` entry to make this configuration permanent.
5. Install an EFI bootloader. Part II of this series described how to install ELILO, but you can install another bootloader if you prefer. If you install GRUB 2 using your distribution's packaging system, the package name probably will be `grub-efi`, and it may replace the

BIOS version of GRUB 2, `grub-pc`. This can render the computer unbootable in BIOS mode, so you should be careful if you use this option. Note that because `efibootmgr` won't work in BIOS mode, you'll probably have to install the bootloader as `/boot/efi/EFI/BOOT/bootx64.efi`.

6. Reboot and activate EFI boot mode in your firmware. With any luck, you'll see your Linux bootloader and be able to boot into Linux in EFI mode. If not, Part IV of this series will provide some troubleshooting tips.

At this point, you can verify that you booted in EFI mode by examining your `dmesg` output, as described earlier. You then can add or change bootloaders—say, adding `rEFIt` if you want to dual-boot with Windows.

Next Time

Part IV of this series will cover tools and techniques to help keep your EFI Linux installation working correctly. Specific issues include kernel updates, dual-boot configurations and troubleshooting procedures. ■

Roderick W. Smith is a Linux consultant, writer and open-source programmer living in Woonsocket, Rhode Island. He is the author of more than 20 books on Linux and other open-source technologies, as well as of the GPT fdisk (`gdisk`, `cgdisk` and `sgdisk`) family of partitioning software.

Rock Out with Your Console Out

Playing and managing your music in text mode.

REBECCA “RUJI” CHAPNIK

Some of you probably have played audio files from the terminal with one-line commands, such as `play`, or even used the command line to open a playlist in a graphical music player. Command-line integration is one of the many advantages of using Linux software. This is an introduction for those who want the complete listening experience—browsing, managing and playing music—without leaving the text console.

Thanks to the Ncurses (New Curses) widget library, developers can design text user interfaces (TUIs) to run in any terminal emulator. An Ncurses application interface is interactive and, depending on the application, can capture events from keystrokes as well as mouse movements and clicks. It looks and works much like a graphical user interface, except it's all ASCII—or perhaps ANSI, depending on your terminal. If you've used GNU Midnight Commander, Lynx or Mutt, you're already familiar with the splendors of Ncurses.

An intuitive interface, whether textual or graphical, is especially

important in a media player. No one wants to sift through a long man page or resort to `Ctrl-c` just to stop an annoying song from playing on repeat, and most users (I'm sure some exceptions exist among *Linux Journal* readers) don't want to type out a series of commands just to `ls` the songs in an album's directory, decide which one you want to hear and `play` it, and then `play` a song in a different directory. If you've ever played music with a purely command-line application, such as SoX, you know what I'm talking about. Sure, a single command that plays a file is quite handy; this article, however, focuses on TUI rather than CLI applications. For many text-mode programs, Ncurses is the window (no pun intended) to usability.

Note to developers: if you want to write a console music player, take advantage of the Curses Development Kit (CDK), which includes several ready-made widgets, such as scrolling marquees and built-in file browsing.

Now, on to the music players!

```

ruji@sabayonengine:~
[ 1 ] Select Files      [ 2 ] Add Group      [ 5 ] Set Group Title  ( - )
[ l ] Load/Add Playlist [ w ] Write Playlist  [ C ] Clear Playlist
[ m ] Move Files After  [ M ] Move Files Before [ / ] Start Search
[ f ] Toggle File Display [ 6 ] Toggle Repeat  [ 7 ] Toggle GroupShuffle
                                                                    ( + )
Global Playback: Play current group, including subgroups
Next Song      : ??? (can't know yet)
                                                                    Mpg1 Layer3
                                                                    44Khz 160kb
                                                                    JointStereo
                                                                    Threads:100
                                                                   
                                                                    [Default]
[Russian Paul songs]
Predator Drone.mp3
Shalom Chaverim.mp3
Sea of Putridity.mp3
                                                                    [ ] shuffle
                                                                    [ ] repeat
                                                                   
                                                                    8:38
                                                                    8:51
                                                                   
                                                                    |< | | >|
                                                                    b p n
                                                                   
                                                                    << [ ] >>
                                                                    B s N
                                                                   
                                                                    [ t ] Mixer
                                                                    null
                                                                    [<]+888% [>]

```

Figure 1. A Playlist in Mp3blaster

Mp3blaster

Mp3blaster was the first console music player I ever used. That was in 2007, by which time it already was a mature and full-featured application. Its history actually dates back to 1997, before the mainstream really had embraced the MP3 format, let alone the idea of an attractive interface for controlling command-line music playback. Back then, it was humbly known as “Mp3player”.

Despite the name, Mp3blaster supports several formats besides MP3s. Currently, these include OGG, WAV and SID. Keep an eye out for FLAC support in the future, as it is on the to-do list in the latest source tarball.

One nice feature of Mp3blaster is the top panel showing important keyboard shortcuts for playlist management. You can scroll through this list using + and -. There is also a useful chart on the right side that shows ASCII art playback symbols (such as |> for play) above their respective shortcut keys. Press ? for detailed help.

You can customize any of the keybindings in your configuration file, which is usually located at ~/.mp3blasterrc. I had to change several of these in order to use Mp3blaster in GNOME due to conflicts with my global hot keys. Mp3blaster’s default keybindings are better suited for use without X.

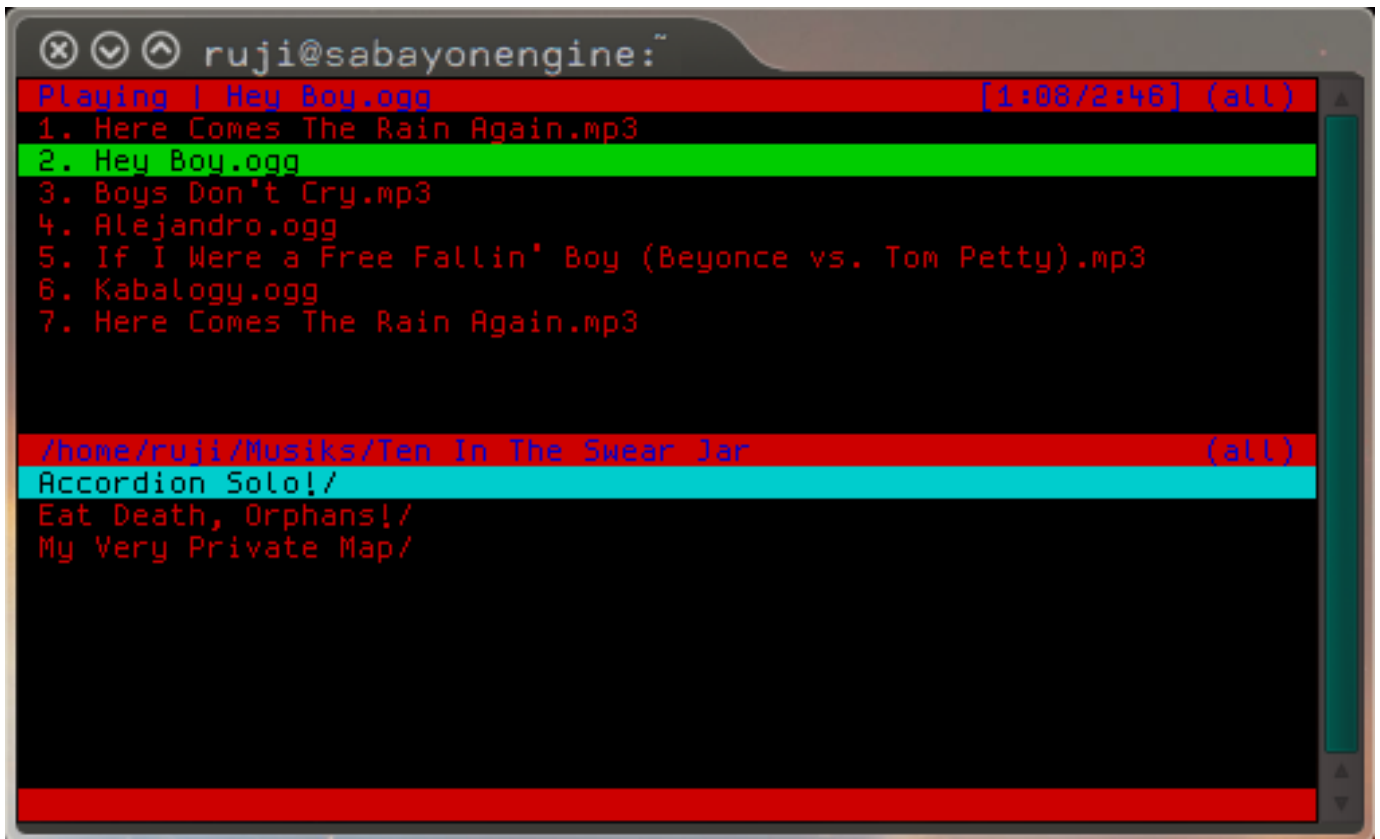


Figure 2. Herrie shows the current playlist on top and a file browser on the bottom.

Herrie

Herrie, meaning “clamour” in Dutch, was first released in 2006. Somehow it has escaped mention in many articles on console music players, but the Herrie community group on the music Web site Last.fm shows true fan dedication.

Herrie is great for Last.fm users because it’s so easy to set up track scrobbling. Most of the music players in this article support scrobbling in some capacity—it’s all open-source software, after all, and in theory, you can write a script to make anything do anything—but configuration is exceptionally simple with Herrie. All you have to do is put your user name and password in your

~/.herrie/config/herrie.conf file. Note that the password should not be in plain text; rather, you should type in the output of `printf %s p4ssw0rd | md5` as stated in the configuration file itself.

MOC

Music on Console (MOC) is a good choice for music libraries that consist of OGG, WAV and MP3 files. It’s easy to use out of the box, boasting a two-paned interface similar to that of Midnight Commander, with a file browser on the left and your playlist on the right. The default keybindings are intuitive—mostly single letters that stand for what they do, such as n for

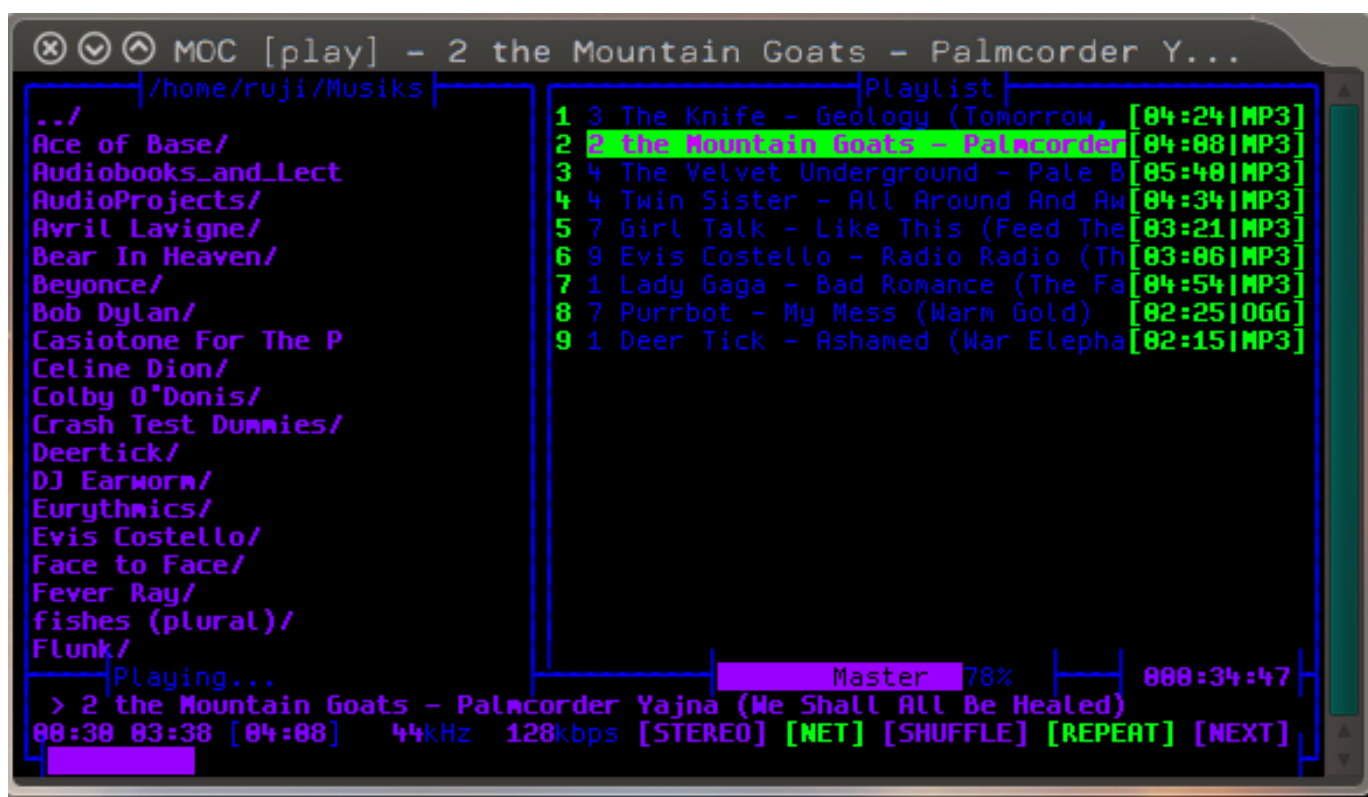


Figure 3. My Customized MOC Layout

“next track” and R to toggle random play, so command-line newbies need not fear any Emacs-style digital acrobatics.

MOC is my go-to Linux music player these days. It’s fast and slick, and it looks just how I want now that I’ve edited my `~/moc/config` file to adjust the colors and the widths of each window pane. Another plus is its support for the JACK Audio Connection Kit (JACK).

The command to start MOC is `mocp`.

Emacs + Bongo/EMMS

If Emacs-style digital acrobatics are your modus operandi, check out Bongo and the Emacs Multi-Media System (EMMS). Both media players run inside

Emacs and provide similar functionality. The main difference is that EMMS is designed to run unobtrusively in the background, while Bongo emphasizes the user interface.

Bongo and EMMS are written in Emacs Lisp. You can install them the same way you’d install any other Emacs package; this may vary from distro to distro, but no matter what operating system you’re using, you’ll probably end up editing some Lisp configuration files. One of the first things to configure is your list of back ends. These programs don’t actually do the dirty work of playing your music files; rather, they are front ends for other programs.

You can link any back end of your choice to a file type as well as pass custom command-line arguments. For example, one of the back ends Bongo recognizes by default is mpg123. If you want it to use, say, mpg321 instead, it's just a matter of editing that line in your configuration file or using Emacs to access Bongo's built-in customization dialog with `M-x customize-group RET bongo RET`. You can add a custom

back end with a few lines such as these:

```
(define-bongo-backend mpg321
  :pretty-name "MPG-Thr33-Tw0-0ne"
  :extra-program-arguments '("--loop 0")
  :matcher '(local-file "mp3" "wav"))
```

Although I use Emacs from time to time, I'm no guru; I admit that the time I spent with Bongo was flustering. For instance, I pressed Return to start playing

```
File Edit Options Buffers Tools Bongo nXhtml ECB Help
File Edit Options Buffers Tools Bongo nXhtml ECB Help
[-] /
  [x] bin
  [+] boot
  [+] dev
  [+] etc
  [-] home
  [x] ftp
M-6 ...i/.emacs.d/bongo
(u) bongo.texinfo
(u) COPYING
(u) HISTORY
(u) lastfm-submit.el
(u) NEWS
(u) README.rdoc
M-1 ...i/.emacs.d/bongo
Enabled backends: mpg123, VLC, ogg123, speexdec, T
iMidity
To modify this list, customize `bongo-enabled-back
ends'.
Bongo is free software licensed under the GNU GPL.
Report bugs to <bongo-devel@nongnu.org>.
81 The Other Side Of Your Face
82 Lady Daydream
83 Milk And Honey
84 All Around And Away We Go
85 Galaxy Plateau
86 Phenomenons
M-2 README.rdoc
[-] ...i/.emacs.d/bongo
(u) README.rdoc
M-3 History
-UUU:%*--F1 *Bongo Library* Bot L25 (Library)--
85 Galaxy Plateau
```

Figure 4. An Emacs session with the Bongo player in the bottom window, Bongo's README in the top window and Emacs Code Browser (ECB) on the left side.

MPD is technically a server-side application; it's great for setting up networked audio in a home media center.

a track—easy enough—but then realized I didn't know how to turn it off. I entered `M-x apropos RETURN bongo` and read through the list of Bongo commands until I found the one I needed: `M-x bongo-stop`. The GitHub home page reveals that you also can stop playback immediately with `C-c C-s`, and there are other key combinations for fancier tricks, such as `3 C-c C-s` to stop playback after the next three tracks finish playing.

That example is a fair representation of my whole experience with Bongo so far. It can be scary if you don't know your way around Emacs very well, but it's extremely powerful and full of options that you'd probably never thought of before.

If you're a Vi/Vim fanatic, consider `Vimmpc` and `Vimp3`.

MPD + Ncmpcpp

In the vast universe of Linux audio, the Music Player Daemon (MPD) could be considered a red giant. Chances are you've at least heard of it if you've done any research on playing music in Linux. It comes preinstalled in many distributions, and I have yet to find a major repository that doesn't include it.

MPD is technically a server-side

application; it's great for setting up networked audio in a home media center. You also can use it simply for local playback. The advantage here is that you can use any client you want to control MPD, and there are many from which to choose. I easily could devote pages to discussing MPD, but that's beyond the scope of this article. Documentation is easy to find on-line.

Now, let's move on to `Ncmpcpp`. This is an Ncurses MPD client, based on `Ncmpc` but more advanced. It includes support for Last.fm scrobbling and music visualization via external libraries. Lyrics fetching and display are built in and can be activated for a selected track by pressing `l`. The lyrics feature is, in fact, what attracted me to `Ncmpcpp` in the first place. I'd tried various scripts to fetch lyrics in other console music players, particularly MOC, but nothing worked for me until `Ncmpcpp`. `Ncmpcpp` can fetch artist information as well.

Although `Ncmpcpp` is terrific once you get it set up, using an MPD client to listen to music isn't always a pragmatic choice. You'll most likely be up and running much faster with a player like `Mp3blaster`, `MOC` or `Herrie`.

I'm someone who likes to experiment with various Linux distributions by

```

0:42/3:35                               The Mess Inside                               Vol: 100%
[playing]                                the Mountain Goats - All Hail West Texas (2002)  [-----]

      Lyrics: the Mountain Goats - The Mess Inside

We took a weekend, drove to Provo
The snow was white and fluffy
A weekend in Utah won't fix what's wrong with us
The gray sky was vast and real cryptic above me

I wanted you
To love me like you used to do

We took two weeks in the Bahamas
Went out dancing every night
Tried to find the creeping sense of dread with temporal things
Most of the time I guess I felt alright

But I wanted you
To love me like you used to do

But you cannot run
=====➔

```

Figure 5. Ncmpcpp Showing Lyrics

installing them on old computers and in virtual machines, and I often test out software in these environments. The truth about MPD is that a lot can go wrong. I let out a (silent) cheer every time I manage to install and use it successfully. Grappling with dependencies is half the battle, and configuring your system, especially your `~/.mpdconf` file, is the other half. I've gotten it to work on some systems without a hitch, but more often than not, I've encountered problems and solved them through trial and error.

Don't let this discourage you; MPD and its wide selection of clients are worth the

effort to set up if you take advantage of their features, and there are plenty of places to get help if you need it. The MPD man page is essential reading; beyond that, read through the official wiki and forums. Your distribution may provide documentation as well. Gentoo's on-line wiki, for instance, has a lengthy section on MPD.

XMMS2 + Kuechenstation/CCX2

Like MPD, XMMS2 is a daemon you can control over a network, and there are various clients for it. The XMMS2 Wiki acknowledges

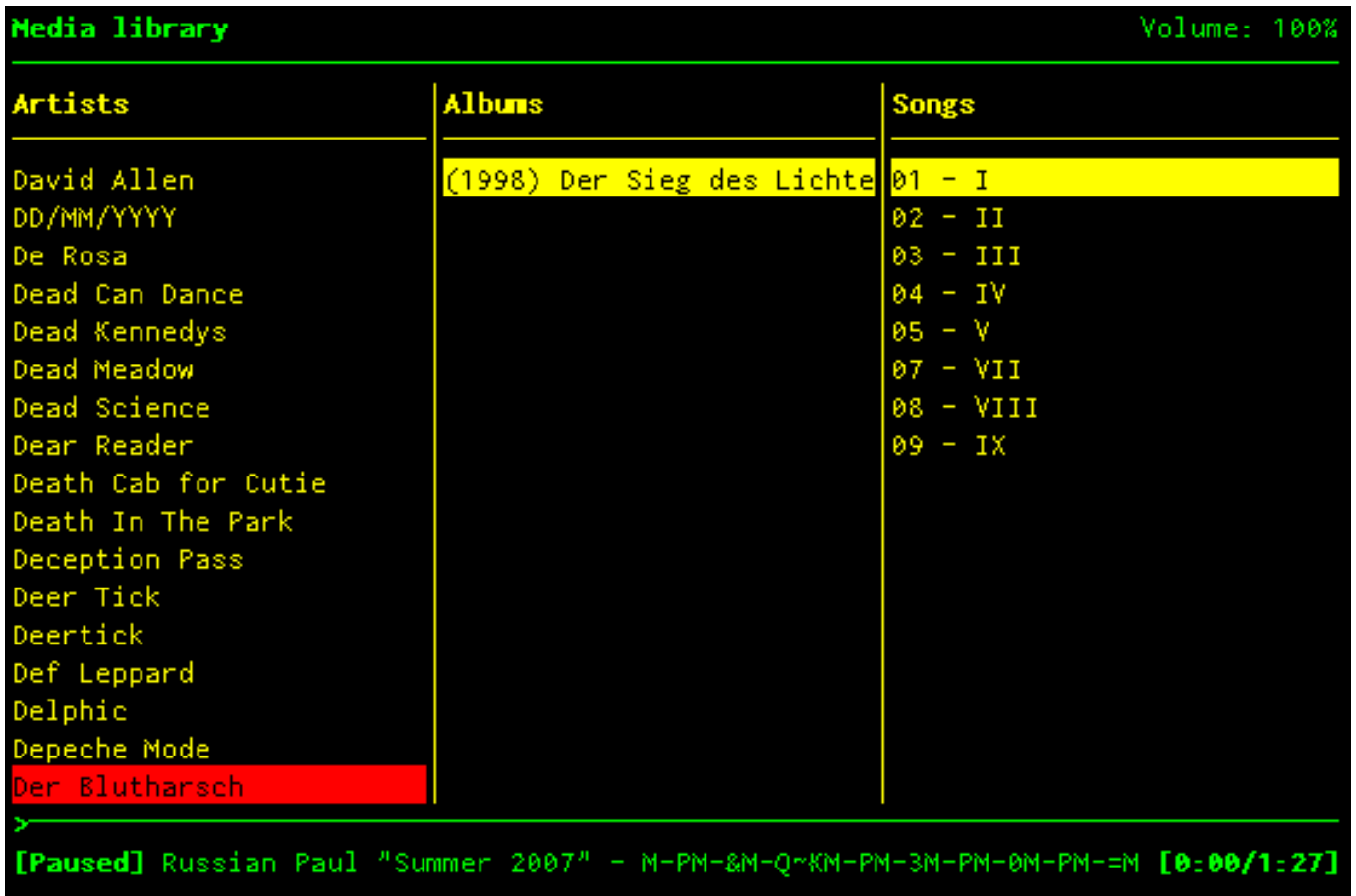


Figure 6. Browsing Files in Ncmpcpp

that the developers of these two applications have similar goals and that a collaboration could eventually be possible. For now though, they are separate packages with separate clients. Two text-mode XMMS2 clients that caught my attention were Kuechenstation and CCX2.

Kuechenstation is one 1337 music player. Okay, I was being mostly facetious there, but take one look at it, and you'll think "eighties demoscene". (Kuechenstation actually has been around only since 2008.) It uses the FIGlet library to display

the current song title in a scrolling marquee of oversized letters made from ASCII characters.

The whole interface is attractive and friendly. You can navigate through several full-screen modes using keybindings that are helpfully listed at the bottom of the screen. These modes include playlist mode, artist information mode and podcast mode, to name a few. The podcast feature is especially notable; I haven't seen podcast support in any of the other music players discussed in this article. Kuechenstation helps you get started

```

-----
[+] [X] [O] [P] [E] [R] [I] [O] [N] [G] [I] [N] [G] [S]
-----
Last track:
Deer Tick - Diamond Rings 2887                               4:41

now playing:
artist: Casiotone For The Painfully Alone
album: Twinkle Echo
title: Jeane, If You're Ever In Portland
path: ...Painfully Alone/Twinkle Echo/82 Jeane, If You're Ever In Portland.mp3
duration: 2:29  id: 138  rating: 0

next tracks:
Twin Sister - Milk And Honey                                4:54
Twin Sister - The Other Side Of Your Face                    7:06

F1: statusinfo F2: playlist F3: medialib F4: streams F10: log
playing (1:57)                                           activated StatusInfoWindow

```

Figure 7. Kuechenstation with My Customized Color Theme

with a few pre-subscribed feeds, which are all in German.

The Kuechenstation configuration file is located at `~/.config/xmms2/clients/kuechenstation.conf`. There you can choose your podcast subscriptions, interface colors and even the scrolling FIGlet font.

CCX2, written in Python, is another solid XMMS2 client. Its command mode will come naturally to Vi/Vim users. All the standard playback and playlist management features are there: search, rename, browse, metadata display and so forth.

So why did I decide to write about two TUI XMMS2 clients instead of just choosing the one with more features? My reasoning is, first of all, that the interfaces of Kuechenstation and

CCX2 are quite different, and each will appeal to different users solely on the basis of personal taste. Second, each has a major feature that the other lacks. CCX2 doesn't come with podcast support as Kuechenstation does, but it does support lyrics fetching out of the box, which Kuechenstation does not.

I suggest trying them both. They are young and in active development, so there's a reasonable chance that a feature you're missing could be added in the future. And, of course, if you're a developer, you can try to add it yourself.

nvlc

The famous VLC media player, known for its ability to play almost any media file you throw at it, comes with a lesser-known Ncurses

```

PLAYING # Deer Tick > Diamond Rings 2007 -- War Elephant [02:09/04:41]
1:he mediainfo
search And I turn from sad to blue
      Yeah I'm choking after you
      Miss, I'm choking now, do you?

      (Go ahead now, tell them all about it)

Wis The So what good would fingers be
Wel If it weren't for diamond rings?
Cau Now I know you're leaving me
      And I know that I'm no king

So plugin/file
If lmod 1285639696
Now size 8144403
And plugin/id3v2
      album War Elephant
      artist Deer Tick

And
And
And then I stay the whole night through

```

Figure 8. One of CCX2's Lyrics Display Layouts

control interface. To start it up, type `nvlc`. The interactive features are noticeably limited in comparison with the vast array of options you may be used to seeing in the GUI version. Press `B` to browse your files and `Return` to add a file to the playlist. Toggle help view with `h` for a complete list of hot keys.

At first glance, `nvlc` doesn't seem all that special. It might not be for you if you want a player that's preconfigured with a hefty arsenal of hot keys, but you can do a lot with it—including adding custom hot keys—if you're willing to experiment.

The path to `nvlc`'s power is through command-line arguments. You can pass arguments ranging from a directory or

LINUX JOURNAL ARCHIVE DVD



NOW AVAILABLE

www.linuxjournal.com/dvd

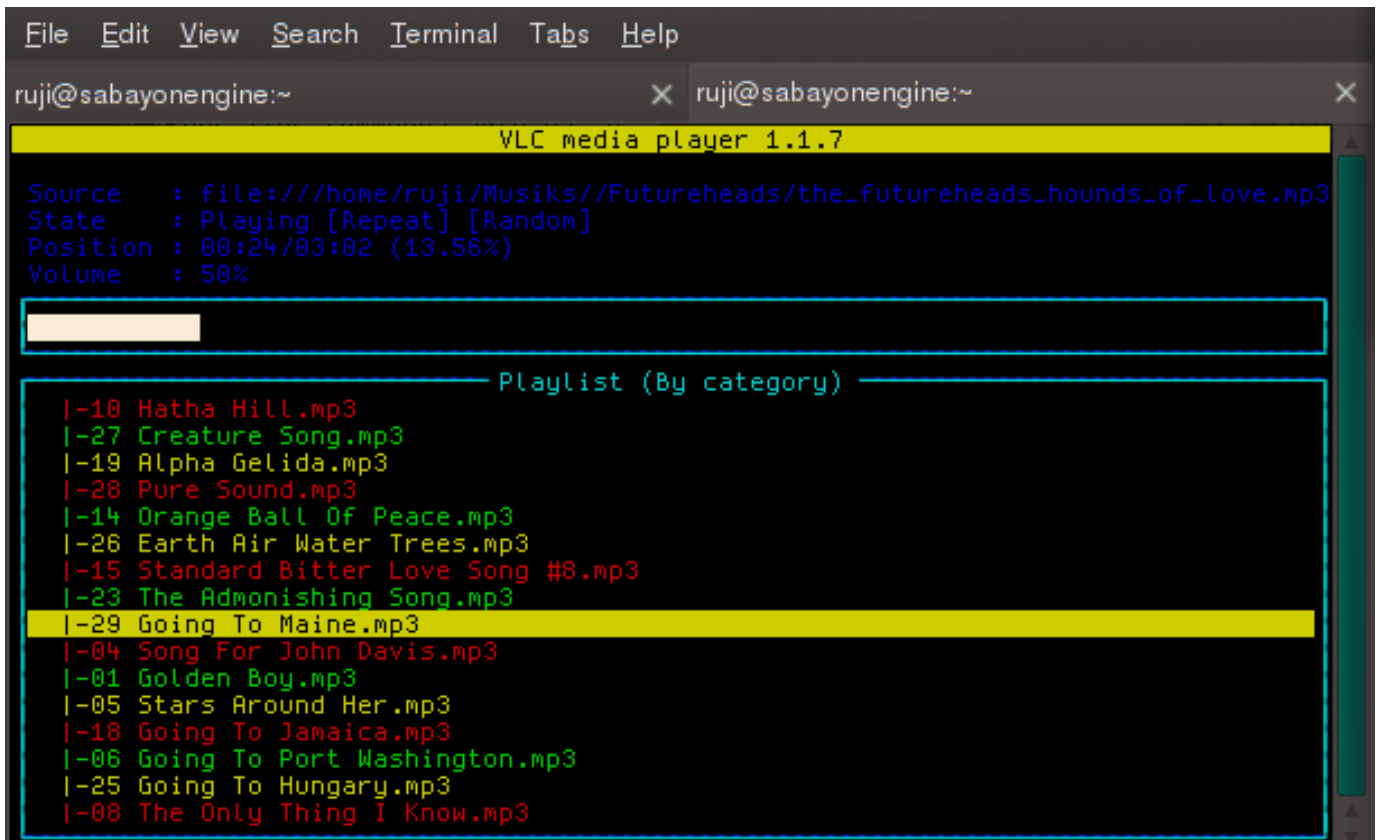


Figure 9. A Playlist in nvlc

playlist (a la, `nvlc /path/to/my/music`) to complex chains of filters. Anything you can do in the GUI version of VLC is possible with `nvlc` if you know which arguments to pass.

Hint: enter `nvlc -h` for basic help, which is actually quite lengthy, or `nvlc -H` for even lengthier help. Enter `nvlc --list` to see what modules are available in your installation or `nvlc --list-verbose` for more-detailed output.

For starters, try:

```

nvlc --audio-filter chorus_flanger --delay-time 150
  --dry-mix 0.8 --wet-mix 0.6 --feedback-gain -0.3
  /path/to/my/music.fileextension

```

MikMod

For the old-schoolers among you who collect modules—and perhaps scoff when you hear the phrases “MP3 player” and “music player” used interchangeably—there is MikMod. MikMod is an old standby from pre-Windows Microsoft DOS. You can use it as a back end for other applications, such as Bongo or EMMS in Emacs, or as a standalone module player.

MikMod will play many module formats. If your file extensions include MOD, XM, IT or S3M, you’re in luck. Sorry, MP3s—no MikMod for you, or for all you WAVs and OGGs and AIFFs. In a way, this is a bit sad, because I’d

love to play my standard music files in a player as awesome as MikMod. I have to keep in mind that many of MikMod's features, such as on-the-fly tempo change and instrument-specific volume bars, are built specifically for module file formats. Perhaps this will be an incentive for me to make some sounds in MilkyTracker.

Conclusion

Many console music players are available for Linux. I chose the few I covered in this article based on my

level of experience with them and on what I considered to be unique and notable features. If the topic intrigues you, go out (or Google) and explore. ■

Rebecca "Ruji" Chapnik is a freelance creator of miscellanea, including but not limited to text and images. She studied art at the University of California, Santa Cruz, and book publishing at Portland State University. She went on to study Linux in her bedroom and also in various other people's bedrooms, crouched anti-ergonomically before abandoned Windows computers. Ruji currently lives in Portland, Oregon. You can find her experiments at <http://rujic.net>.

LINUX JOURNAL on your Android device

Download
app now in
the **Android
Marketplace**



www.linuxjournal.com/android

Resources

Ncurses: www.gnu.org/software/ncurses

Mp3blaster: mp3blaster.sourceforge.net

Herrie: herrie.info

MOC: moc.daper.net

Bongo: <https://github.com/dbrock/bongo>

EMMS: www.gnu.org/software/emms

MPD: www.musicpd.org

Ncmpcpp: unkart.ovh.org/ncmpcpp

Kuechenstation: kuechenstation.sourceforge.net

CCX2: palbo.github.com/ccx2

VLC: www.videolan.org/vlc

MikMod: mikmod.raphnet.net

Please visit us at <http://northeastlinuxfest.org> for more information

Join us for the 2nd annual Northeast GNU-Linuxfest!

MARCH 17, 2012

WORCESTER STATE UNIVERSITY

STUDENT CENTER

486 CHANDLER STREET

WORCESTER MA

<http://northeastlinuxfest.org>



<http://northeastlinuxfest.org>

Among the highlights -

Speakers:

Dru Lavigne

John Sullivan

James Turk

Events

Hackerspace competition

Open Street Map workshop

LPI & BSD exams

Please visit us at <http://northeastlinuxfest.org> for more information



DOC SEARLS

Does Linux Have an Economy?

How can you value something that isn't scarce and costs nothing?

In the midst of a long thread on the ProjectVRM list, Joe Andrieu wrote this summary statement about economics and markets: "Economics is about allocating limited resources. The market is a mechanism for discovering prices." He then added, "You can allocate other ways, of course."

Although one might argue those definitions, they do explain why Linux, along with the rest of free software and open source, gets little attention in economics discussions and little credit for influence on markets—mostly because it allocates in other ways.

Chief among those is use. Linux strives to be useful. But kernel development doesn't happen in userspace, which is where economics and markets live. It happens in kernel space, which is boring, unless you're a contributor to it or you depend on keeping up with what's happening there.

Even though the only code I know is

Morse, I often find reassuring the constant hum of progress coming off the Linux Kernel Mailing List (<http://lkml.org>). The latest post today (January 4, 2012, as I write this) concerns a patch for better low memory handling from Mike Waychison. The hottest message is one from Greg KH announcing the Linux 3.1.7 release, which has a single bugfix for resume issues, and which he says you won't need if those issues aren't yours.

Maybe it's just that I spent too much of my life in marketing and, therefore, love too much the sound of its absence, but I find in that kind of talk a reassurance that something right is going on in the world.

Or, maybe it's just that the instruction programmed into my head by my old-school parents—"make yourself useful"—remains a Platinum Rule, more valuable even than the Golden one, perhaps because it's something you do *for* others, rather than "unto" them.

It's around "for" that Linux allocation

Advertiser Index

Thank you as always for supporting our advertisers by buying their products!

succeeds. Linux has saturated the networked world (and our lives with it) with almost no help from marketing of the usual sort. Our collective dependence on Linux today verges on the absolute. Take Linux away from the world, and all but the most electronics-free markets would cease to operate.

And still, because Linux is neither scarce nor sold, economics doesn't know how to value it. Not exactly, anyway. That is, not in terms of costs or prices.

Economics does have terms that cover the kind of stuff that Linux is, however. Two are *non-rivalrous* and *non-excludable*. Non-rivalrous means use of it by one party does not prevent another party from using it; non-excludable means it can't be made scarce. Linux is both, making it what economists call a *public good*. Wikipedia sorts out *rivalrous*, *excludable* and their *non-* counterparts as shown in Table 1.

This differs little from what it said when I posted the same table for "Greater Goods", an essay for the December 2006 issue of *Linux Journal* (<http://www.linuxjournal.com/article/9344>). Back then I noted: "Yet it seems that the purpose of free and open-source development is to produce a common pool resource. As Craig Burton has often observed, the idea is to create

ADVERTISER	URL	PAGE #
EMAC, INC.	http://www.emacinc.com	55
EMPEROR LINUX	http://www.emperorlinux.com	29
FLOURISH	http://flourishconf.com	119
HPC ON WALL STREET	http://www.flagmgmt.com/linux	49
IXSYSTEMS	http://www.ixsystems.com	7
LINUXFEST NORTHWEST	http://linuxfestnorthwest.org	93
LULLABOT	http://store.lullabot.com	2
MICROWAY	http://www.microway.com	46, 47
NORTHEAST LINUXFEST	http://northeastlinuxfest.org	115
O'REILLY WHERE	http://whereconf.com/where2012	85
THE PERCONA LIVE MySQL CONFERENCE AND EXPO	http://form.percona.com/PLMCELJ.html	57
POSSCON	http://posscon.org	17
RACKMOUNTPRO	http://www.rackmountpro.com	15
SILICON MECHANICS	http://www.siliconmechanics.com	3

ATTENTION ADVERTISERS

The *Linux Journal* brand's following has grown to a monthly readership nearly one million strong. Encompassing the magazine, Web site, newsletters and much more, *Linux Journal* offers the ideal content environment to help you reach your marketing objectives. For more information, please visit <http://www.linuxjournal.com/advertising>.

Table 1. Rivalrous, Excludable and Their Non- Counterparts (from Wikipedia)

	EXCLUDABLE	NON-EXCLUDABLE
RIVALROUS	Private goods: food, clothing, cars, personal electronics	Common goods (common-pool resources): fish stocks, timber, coal
NON-RIVALROUS	Club goods: cinemas, private parks, satellite television	Public goods: free-to-air television, air, national defense

common infrastructural building material that supports whole industries, rather than just one player in that industry. We do this by making goods that become abundant by being both open and in the public domain.”

And we still do, as proven by the continuing progress of useful free and open code-making. But, as the T-shirt says, “Yes, it works in practice, but will it work in theory?”

The best theories I’ve found so far are those of Yochai Benkler, Elinor Ostrom and Lewis Hyde. (Yochai and Lewis I know well. Elinor I’ve never met but would like to.) Yochai addresses the issue directly in *Coase’s Penguin, or Linux and the Nature of the Firm*. Elinor does so less directly in *Governing the Commons: The Evolution of Institutions for Collective Action*. Lewis’ latest and most direct approach is in *Common as Air: Revolution, Art, and Ownership*. My own fave, however, is Lewis’ first book, *The Gift: Creativity and the Artist in the Modern World*, because it goes more deeply into the matter of motivation.

In *The Gift*, Lewis makes a distinction between *logos* and *eros*. The first, he says, is “reason and logic in general”,

and adds that “a market economy is an emanation of *logos*”. The second is giving without expectation of exchange, or even of calculation. For example, the love we give to spouses and children, and the respect we give to friends, are gifts. To put a price on *eros* would violate its nature, and make it something else.

“People live differently who treat a portion of their wealth as a gift”, Lewis writes. “To begin with, unlike the sale of a commodity, the giving of a gift tends to establish a relationship between the parties involved. Furthermore, when gifts circulate within a group, their commerce leaves a series of interconnected relationships in its wake, and a kind of decentralized cohesiveness emerges.”

Clearly we have that, at least between those who create Linux and its conscious beneficiaries. Right now, that’s a superset of *Linux Journal* readers. Perhaps in time it will include economists and policy-makers as well. ■

Doc Searls is Senior Editor of *Linux Journal*. He is also a fellow with the Berkman Center for Internet and Society at Harvard University and the Center for Information Technology and Society at UC Santa Barbara.

ATTEND FLOURISH
2012

INSPIRING TALKS,
ILLUMINATING WORKSHOPS

20
12

● OPEN SOURCE, ● OPEN FUTURE

MARCH
30-31

UIC STUDENT CENTER EAST
750 S. HALSTED STREET
CHICAGO, IL



FLOURISH!

SIX YEARS ● OF SUPPORTING
● OPEN-SOURCE IN THE MIDWEST