AN INDEPTH LOOK AT REMOTE BACKUPS

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

WATCH: ISSUE OVERVIEW

MARCH 2016 | ISSUE 263
LinuxJournal.com

NEW: Single-Column Format!

tmux Guide
Get Up to Speed Quickly

Use an initrd
as a Real Root Filesystem for Your Servers

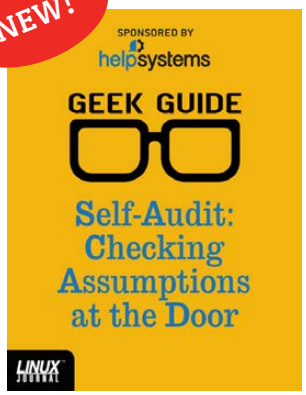
PLUS:

Data Science Methods for Analyzing Logfiles

Convert Numeric Bases from the Command Line

Add an OEM-Style Factory Installer to Your GRUB Menus

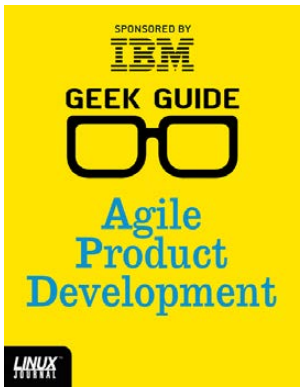
NEW!



Self-Audit: Checking Assumptions at the Door

Author: Greg Bledsoe

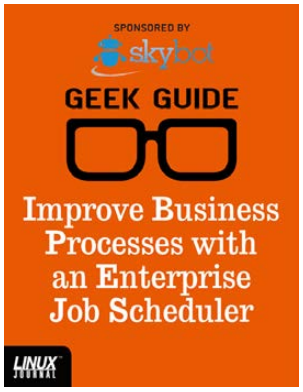
Sponsor: HelpSystems



Agile Product Development

Author: Ted Schmidt

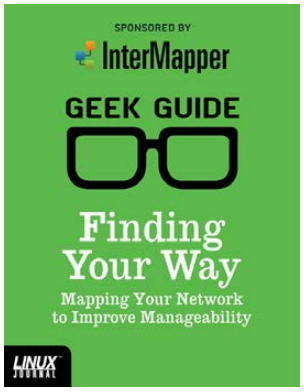
Sponsor: IBM



Improve Business Processes with an Enterprise Job Scheduler

Author: Mike Diehl

Sponsor: Skybot



Finding Your Way: Mapping Your Network to Improve Manageability

Author: Bill Childers

Sponsor: InterMapper

DIY Commerce Site

Author: Reuven M. Lerner

Sponsor: GeoTrust

Combating Infrastructure Sprawl

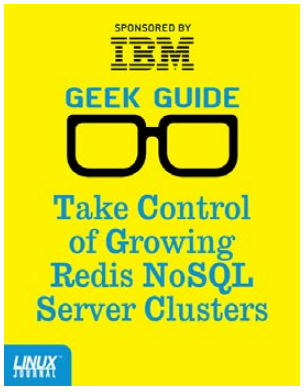
Author: Bill Childers

Sponsor: Puppet Labs

Get in the Fast Lane with NVMe

Author: Mike Diehl

Sponsor: Silicon Mechanics & Intel



Take Control of Growing Redis NoSQL Server Clusters

Author: Reuven M. Lerner

Sponsor: IBM

CONTENTS

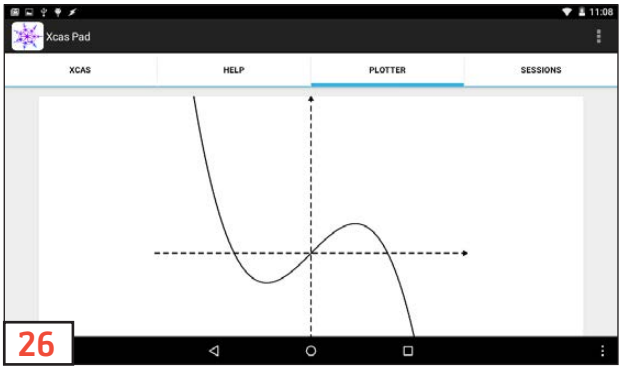
MARCH 2016
ISSUE 263

FEATURES

ON THE COVER

COLUMNS

IN EVERY ISSUE



LINUX JOURNAL (ISSN 1075-3583) is published monthly by Belltown Media, Inc., PO Box 980985, Houston, TX 77098 USA.
Subscription rate is $29.50/year. Subscriptions start with the next issue.

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 • Adam Monsen

President	Carlie Fairchild publisher@linuxjournal.com
Publisher	Mark Irgang mark@linuxjournal.com
Associate Publisher	John Grogan john@linuxjournal.com
Director of Digital Experience	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

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 980985, Houston, TX 77098 USA

Now We're the Cool Kids!

SHAWN POWERS

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.

I wish I could go back and tell eight-year-old me that someday it would be a point of pride that I wrote BASIC programs on a TI-99/4A connected to a black-and-white TV. Back then, even playing video games was looked down on, and if you actually typed out a 100-line program to make a worm wiggle across the screen, you were a straight-up nerd. Oh, how the times have changed. Now all those years of nerdery have paid off, and I'm the guy everyone asks tech questions. I've turned that nerd badge of shame into one of honor. Dear eight-year-old me: you'll turn out fine. PS: Learn to code while you're young; you'll regret it later if you don't!

This month, we have columns and articles written by a bunch of nerds. Awesome, right? We start out with Reuven M. Lerner, who talks about analyzing data—specifically, analyzing Apache Web server logfiles. Python turns out to be a great tool for analyzing this sort of data, and Reuven shows how to use some awesome Python tools to do so. Dave Taylor follows up with bconvert, which is a great program for converting base numbers in a script. If you are a fan of that "There are only 10 types of people in the world" shirt, you'll want to read Dave's column.

Kyle Rankin describes how to tweak GRUB2 so that you easily can boot from an ISO file. Although it's still possible to extract an ISO and create a bootable

VIDEO: Shawn Powers runs through the latest issue.

filesystem to boot from a USB drive, GRUB2 allows booting from an ISO file itself. I add my nerdiness to the mix this month by showing multiple ways to back up your files—specifically, backing them up off-site. As someone who has had my house burn down, I can tell you firsthand how important it is to back up your data somewhere geographically different from where you create it. Thankfully with the cloud, there are lots of options, and many of them are free or cheap.

If you've booted Linux in the past decade, you've used an initrd in order to load the system. Using an init ramdisk (or initram filesystem) is a great way to load a temporary, stripped-down root filesystem during bootup. Eduardo Arcusa Les explains the nuances of initrd and shows how to take advantage of the brilliant concept that makes booting Linux so easy. Using a real-world example, he describes how initrd has been incredibly useful for him and can be for others as well.

Finally, my friend Charles Thomas covers how to use tmux to split your terminal window into multiple window panes and rearrange them on the fly. I have to admit, I'm so used to using screen that I haven't really tried tmux. Part of my hesitation is that screen works, and tmux seems a bit confusing. Charles takes away one of my complaints this month by explaining how powerful (and usable) tmux can be. You might even find tmux to be better than screen depending on your situation.

We have all the same product announcements, tech tips and cool upfront pieces this month as well. Plus, you might notice we've switched to a single-column format, which should make reading on digital devices much easier—especially reading code snippets! We enjoy being nerds, and we enjoyed putting this nerdy issue together for you. We hope you enjoy it as well!■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

PREVIOUS
Current_Issue.tar.gz

NEXT
UpFront

Upload.cgi Code

Regarding Charles Fisher's "Secure File Transfer" article in the January 2016 issue: I was surprised to see in his article on securing FTP the use of an ancient and insecure piece of software, upload.c, which is more than 15 years old, and has no protection against trivial stack corruption. strcat, strcpy, sprintf and the like are dangerous and should be avoided unless in completely straightforward cases. The standard replacements for these are strncat, strncpy and snprintf, which are all three POSIX. upload.c is simply no longer from this time and should in my opinion never be used without an almost complete rewrite.

—Mischa Salle

Charles Fisher replies: *I agree that Kessels' upload.c is of questionable quality due to its age and (lack of) security. It was, however, the only publicly available C code for complete handling of RFC-1867 as of 2015. I have since written the following prototype to replace it, coded for compactness over clarity, intended as a starting point for discussion—not as a fully featured server. It uses modern libraries (including BSD strlcat/cpy, which are safer for C string handling). Note that Conte's sha256 functions do not appear to work properly on big-endian architectures. This code is (hopefully) some improvement over the predecessor:*

```
#include <stdio.h>    /*xmit.c *prototype* RFC-1867 file transfer with:*/
#include <string.h>   /* http://libccgi.sourceforge.net - cgi by Losen */
#include <stdlib.h>   /* http://bradconte.com/sha256_c  - sha by Conte */
#include <unistd.h>   /* Copyright 2015 Charles Fisher. Distributed    */
#include <sys/types.h>/* under the terms of the GNU Lesser General     */
```

```
#include <sys/stat.h> /* Public License (LGPL 2.1)                    */
#include "ccgi.h"

/* Compile with: gcc -static -Wall -I. -O2 -o xmit.cgi xmit.c ccgi.c \
                  strlcpy.c strlcat.c sha256.c
##BEWARE:
http://www.slideshare.net/phdays/chw00t-breaking-unices-chroot-solutions
*/

#define UPL_PATH "/upload/" /* Trailing slash or filename prefix.      */
#define TMP_PATH "/upload/cgi-upload" /* Must point to same filesystem.*/
#define uchar unsigned char /* 8-bit byte                             */
#define uint unsigned int /* 32-bit word                              */

typedef struct { uchar data[64];  uint datalen;  uint bitlen[2];
 uint state[8]; } SHA256_CTX;

void sha256_init(SHA256_CTX *);
void sha256_update(SHA256_CTX *, uchar *, uint len);
void sha256_final(SHA256_CTX *, uchar *hash);
size_t strlcat(char *, const char *, size_t);
size_t strlcpy(char *, const char *, size_t);

int main(int argc, char **argv)
{CGI_varlist *vl;  const char *name;  int mask_len = strlen(TMP_PATH);
 char prefix[BUFSIZ] = UPL_PATH, dst[BUFSIZ], *p = getenv("SCRIPT_NAME");

 /* Removing write and execute should constrain uploads to 400.       */
 umask(umask((mode_t)0)|S_IWUSR|S_IWGRP|S_IWOTH|S_IXUSR|S_IXGRP|S_IXOTH);

 printf("Content-type: text/plain\r\n\r\n");

 if(p != NULL) /* Use the SCRIPT_NAME as a filename local prefix.     */
 {char genbuf[BUFSIZ];

  if(strlcpy(dst, p, BUFSIZ) >= BUFSIZ) return 1;
```

```
 if((p = strrchr(dst, '/')) != NULL) p++; else p = dst;
 if(strlcpy(genbuf, p, BUFSIZ) >= BUFSIZ) return 1;
 if((p = strchr(genbuf, '.')) != NULL) *p = '\0';
 if(strlcat(prefix, genbuf, BUFSIZ) >= BUFSIZ ||
  strlcat(prefix, "-", BUFSIZ) >= BUFSIZ) return 1;
} else if(strlcat(prefix, "IN-", BUFSIZ) >= BUFSIZ) return 1;

if((vl = CGI_get_all(TMP_PATH "-XXXXXX")) == 0)
{ printf("CGI_get_all() failed\r\n"); return 1; }

sync(); /* Rather: sync && echo 3 > /proc/sys/vm/drop_caches          */

for(name = CGI_first_name(vl); name != 0; name = CGI_next_name(vl))
{FILE *fp;  CGI_value *val;  struct stat junk_buf;  int i, j;

 if(!(val = CGI_lookup_all(vl, 0))) continue;
 for(i = 0; val[i]; i++)
 { /* Does filename match TMP_PATH, and does it exist?               */
  if(!strncmp(val[i], TMP_PATH, mask_len) && !stat(val[i], &junk_buf))
  { /* Abort if sent an empty|malicious|oversized filename.          */
   j = i++;
   if(!strlen(val[i]) || strchr(val[i], '/') || strchr(val[i], '\\') ||
    strlcpy(dst, prefix, BUFSIZ) >= BUFSIZ ||
    strlcat(dst, val[i], BUFSIZ) >= BUFSIZ) {printf("error");return 1;}

   if(link(val[j], dst))
   { /* On link failure, try our best to keep this data.             */
     if(strlcat(dst, val[j] + mask_len, BUFSIZ) >= BUFSIZ ||
      link(val[j], dst)) /* mkstemp suffix appended to filename.     */
     { printf("name_error\t%s\r\n", val[i]); continue; }
   }

   if(unlink(val[j])) { printf("tmp_error\t%s\r\n", val[i]); }

   if((fp = fopen(dst, "r")))
   {SHA256_CTX ctx;  uchar buf[BUFSIZ];
```

```
    sha256_init(&ctx);
    while((j = fread(buf, 1, BUFSIZ, fp))) sha256_update(&ctx, buf, j);
    sha256_final(&ctx, buf); fclose(fp);
    for(j = 0; j < 32; j++) printf("%02x", buf[j]);
    printf("\t%s\r\n", val[i]);
 }}}}
 CGI_free_varlist(vl); return 0;
}
```

Dave Taylor and Scripts

I've followed Dave's column for many years, and I must say his scripting skills are without equal. I thought I would pass along an idea for a future column. I've been in IT for 47 years, and what amazes me is parallel tasking. Start a script that starts many programs and monitors the PID, plus use interrupt to put you back into a menu to control those tasks. To me, that is bang-for-the-buck in scripting. I'd be interested in a generic version where you could just drop those tasks in a list or keep track of one starting a series of scripts—just an idea.

—Larry Dalton

Dave Taylor replies: *That's a cool idea, Larry, and thanks so much for writing in! I don't have a column due to the boss for a few weeks yet, so let me keep this on the proverbial drawing board and see if I can work it into my next column.*

And, you've been in IT for 47 years. Did you start with punch cards or paper tape? I remember visiting my Dad's workplace back in the early '70s and they had paper tape as a storage medium. I was quite impressed. Now I have more storage on my watch.

Security?

In Shawn Powers' Editors' Choice article recommending Team Viewer (see "Help Me, Uncle Shawn" in the January 2016 Upfront section), he fails to mention the major problem with this type of solution in that the data path goes through a third-party server and, thus, poses an unknown security risk.

It is much better, in my opinion, to use something like RealVNC that provides a direct point-to-point encrypted link between the two computers. It is a simple, one-off job to put port 5900 through the router.

—Ian

Shawn Powers replies: *Your security and privacy concerns are valid, and perhaps I should have pointed them out. Unfortunately, most folks needing the sort of help I can offer over Team Viewer don't know what a port is, much less how to forward TCP port 5900 through their router to whatever private IP their computer might have. In the case of my daughter at college, she doesn't even have that option. (She uses the university's Wi-Fi.) It would be possible to set up SSH keys and an automatic outgoing SSH tunnel to a server I own with a public IP address, and then reverse-tunnel port 5900 through that. But, for simple help with formatting a college paper or installing a printer, it's not worth the effort to guarantee privacy—at least not for me. You are correct, however, that it would have been good to include that information in my article. Thanks for pointing out the potential privacy issue.*

PHOTO OF THE MONTH

Remember, send your Linux-related photos to ljeditor@linuxjournal.com!

WRITE LJ A LETTER

We love hearing from our readers. Please send us your comments and feedback via http://www.linuxjournal.com/contact.

RETURN TO CONTENTS

A Technical Conference Exploring Open Tech and the Open Web

GREAT WIDE OPEN

March 16 & 17

Downtown Atlanta

In the heart of Technology Square
and Georgia Tech University

www.greatwideopen.org

SOME OF THE TOP TECHNOLOGISTS IN THE WORLD WILL BE FEATURED:

Kelsey Hightower
Developer Advocate
Google Cloud Platform

Danese Cooper
Distinguished Member/CTO
PayPal/Wikimedia Foundation

Chris Van Tuin
Chief Technologist/
Western
Red Hat

Steve Klabnik
Developer/Author
Mozilla

Erica Stanley
Founder
Acire Studios, Women Who Code - Atlanta

Trek Glowacki
Core Team Member
Ember.js

PREVIOUS
Letters

NEXT
Editors' Choice

diff -u

What's New in Kernel Development

Sometimes it's necessary to change function semantics inside the kernel, and then find and update all users of that function to match the new semantics. Such changes can result in huge patches going into the source tree, affecting hundreds of files.

Al Viro wanted to do a change like that to a bunch of memory handling routines. He'd noticed that the existing memory allocation tools all returned plain numbers that users then would have to convert to pointers in the vast majority of cases. Al posted a mega-whopper patch, making those functions all return pointers instead of plain numbers.

Linus Torvalds didn't like it though. One of the problems with those immense semantic-changing patches, he said, was that back-porting other unrelated patches became more difficult. For each patch that needed to be backported—security fixes, new drivers and so on—Linus said the port would need to be reworked significantly just in order to get across the barrier of Al's changes. That would be time-consuming for the developer, would increase the likelihood of new bugs, and it didn't seem to carry enough value to justify it.

The way to go about it, Linus said, was to create all new functions, with new names, with the new semantics, and let the various parts of the kernel switch over to the new calls as they pleased. But, even that seemed

hard to justify to him.

Ultimately, Al dropped his big patch and posted a new set of guidelines for memory allocation that would help users resolve questions of which functions to use in which circumstance:

> 1) Most of the time kmalloc() is the right thing to use. Limitations: alignment is no better than word, not available very early in bootstrap, allocated memory is physically contiguous, so large allocations are best avoided.
>
> 2) kmem_cache_alloc() allows to specify the alignment at cache creation time. Otherwise it's similar to kmalloc(). Normally it's used for situations where we have a lot of instances of some type and want dynamic allocation of those.
>
> 3) vmalloc() is for large allocations. They will be page-aligned, but *not* physically contiguous. OTOH, large physically contiguous allocations are generally a bad idea. Unlike other allocators, there's no variant that could be used in interrupt; freeing is possible there, but allocation is not. Note that non-blocking variant *does* exist - __vmalloc(size, GFP_ATOMIC, PAGE_KERNEL) can be used in atomic contexts; it's the interrupt ones that are no-go.
>
> 4) If it's very early in bootstrap, alloc_bootmem() and friends may be the only option. Rule of the thumb: if it's already printed 'Memory: ...../..... available.....' you shouldn't be using that one. Allocations are physically contiguous and at that point large physically contiguous allocations are still OK.
>
> 5) if you need to allocate memory for DMA, use dma_alloc_coherent() and friends. They'll give you both the virtual address for your use and DMA address referring to the same memory for use by device; do *NOT* try to derive the latter from the former; use of virt_to_bus() et.al. is a Bloody Bad Idea(tm).
>
> 6) If you need a reference to struct page, use alloc_page/alloc_pages.
>
> 7) In some cases (page tables, for the most obvious example), __get_free_page() and friends might be the right answer. In principle,

it's case (6), but it returns page_address(page) instead of the page itself. Historically that was the first API introduced, so a _lot_ of places that should've been using something else ended up using that. Do not assume that being lower level makes it faster than e.g. kmalloc() - this is simply not true.

System calls notoriously have insufficient error reporting. Some take lots of inputs, and if any of them are wrong in any way, or fail some obscure bounds check, the call returns "EINVAL" for invalid data, but doesn't give any other clue about which piece of data had the problem, or what the value was, or where in the code the problem occurred.

Alexander Shishkin recently tried to implement a solution to this. The real issue though is that the kernel can't simply change the way system calls handle return values. There's code all through the kernel and in userland that depends upon the current behavior. Any solution, therefore, would somehow have to provide additional reporting information, without changing the way existing calling routines received syscall return values.

Alexander's technique took advantage of the fact that system calls generally were processed through a set of macros before sending their return values back to the calling routines. By designing an entirely new set of return values for the actual system calls, Alexander's code could reference an error message holding tank that the macros would be able to process while still returning the originally intended error code to the calling routine.

The macros would place a pointer to the detailed error reports, in JSON format, into the task_struct data structure, where it could be retrieved by the calling routines, using a prctl() call.

Jonathan Corbet, however, had strong doubts about this approach. For one thing, if the calling routine didn't actively query and reset the new debugging data, that data would just sit in the task_struct, getting stale. Although clearing out the debugging data automatically would defeat the purpose of placing it there originally.

And, **Johannes Berg** also pointed out that with Alexander's changes in effect, applications could break if they had to run on older kernels and expected the new debugging data to be available.

Ultimately, Alexander's approach was not adopted, although no better idea emerged. It's a thorny and persistent problem. It's not clear that any solution will be able to answer all objections, but maybe something will be able to answer more objections than the status quo.

There's a **Y2038** bug in Linux. It's the day when the 32-bit UNIX timestamp rolls back to zero. Since Linux basically runs the known universe these days, the bug has to be dealt with, probably by updating the timestamp to hold a 64-bit value. **Deepa Dinamani** posted some patches to do that, but the problem didn't end there.

The solution had to account for a wide range of possibilities. For example, each different filesystem (NFS, ext4, FUSE and so on) needed its own hand-crafted Y2038 bugfix. It wasn't the kernel alone that needed the fix. Also, after the year 2038, even if all the filesystems had their own fixes, how would a user be able to mount an older filesystem instance that did not have the fix in place? That needed to be solved as well. Additionally, there were corporate interests to consider. Certain service contracts would require a Y2038 fix to be in place, perhaps decades before the bug actually would hit.

Overall, it's going to be a lot of work. **Arnd Bergmann**, **Dave Chinner** and Deepa had a long technical conversation about the ins and outs, but the clearest sense of direction to emerge from the discussion was that they should ignore everything that wasn't directly relevant, and they should hew off as many smaller chunks to solve as they possibly could in the hopes that the main chunk might get easier and more manageable.—Zack Brown

THEY SAID IT

Sometimes we do a thing in order to find out the reason for it. Sometimes our actions are questions not answers.
—John Le Carré

The greatest justice in life is that your vision and looks tend to go simultaneously.
—Kevin Bacon

There are some things you learn best in calm, and some in storm.
—Willa Cather

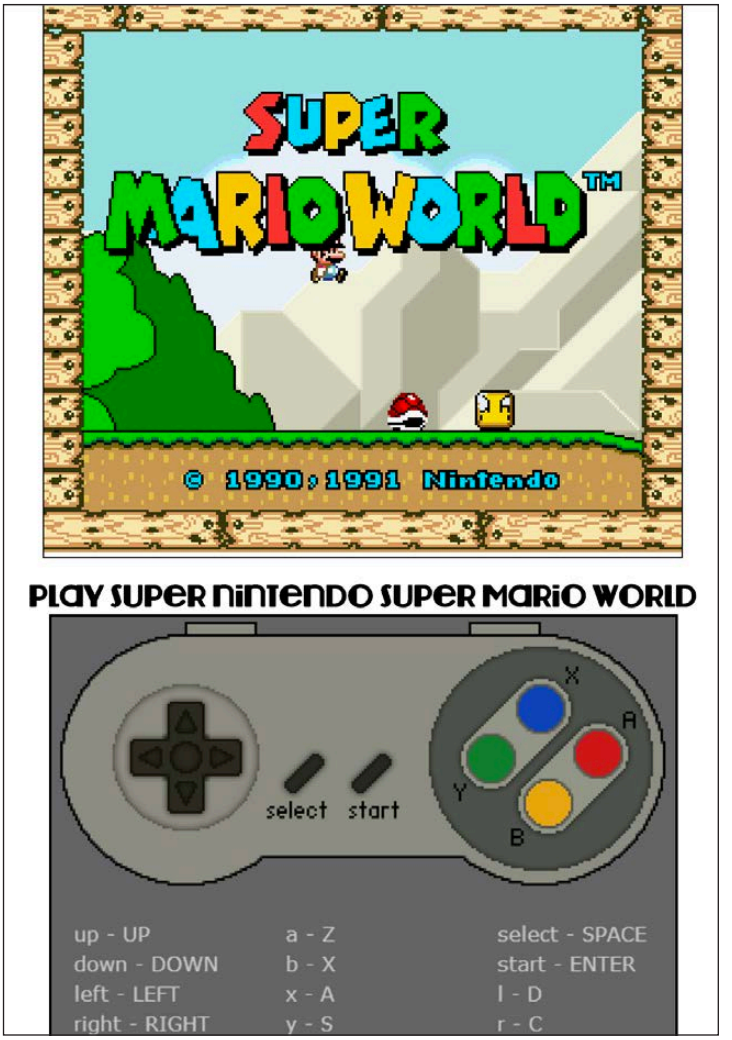
The only true happiness comes from squandering ourselves for a purpose.
—William Cowper

If your ship doesn't come in, swim out to it!
—Jonathan Winters

Your Youth, in a Browser!

I've mentioned many times before the questionable nature of downloading video game ROMs in order to emulate them on your computer. (Heck, my very first article in *Linux Journal* was a description on how I built a MAME cabinet!) Although I have some emulation tricks up my sleeve for future issues, here I thought I'd mention the vast number of Flash-based emulators available on the Internet.

Some sites are more ad-laden than others, but many of them work quite well. For example, for old-school Nintendo games, check out http://www.nintendoemulator.com for a huge list of fully functional games. I played *Super Mario Brothers* on my laptop using Chrome's built-in Flash and wasted a great deal of time doing so!



As with downloading and emulating ROMs locally, you should be aware of potential legal and moral issues. I actually own so many old games that I don't have any moral problem playing with emulators, but I'm not the ultimate authority. If you want to try out some online emulators, either search Google for a few options, or visit the site mentioned above. It's a great way to spend a boring afternoon!

—Shawn Powers

LinuxFest Northwest

April 23rd & 24th

Bellingham, WA

- All things Open Source
- 40+ Exhibitors
- 80+ Presentations
- 1500+ Attendees
- Prizes and after party
- FREE admission & parking
- Bring the whole family!

Hosted By

Bellingham TECHNICAL COLLEGE

linuxfestnorthwest.org

Android Candy: Digital Funnies

One thing I truly miss about the "old-school" way of reading the newspaper is that I don't get to read the funny pages. No, that's not all I would read (although admittedly it may have been the first page I turned to), but a little levity always makes the day better. I'm not a big fan of graphic novels or even comic books, but the daily funny pages are just my speed.

Whether you love them or hate them, the GoComics Web site is full of new and old comics alike (http://www.gocomics.com). The GoComics app for Android is a free way to consume those daily comic strips whether they're new or decades old. My personal favorite comic of all times is Garfield (shown in the screenshot). I collect Garfield memorabilia, Garfield books, and as a kid, I even had Garfield sheets and curtains in my room. Anyway, thanks to the GoComics app, I'm able to read decades of Garfield comics at my leisure, and see the latest strips as well.

If it weren't for GoComics, I'd have to wait to buy the books every year for my daily dose of fuzzy fun. Get the free version in the Google Play Store, or pay an annual fee of about a buck a month to get the ad-free version. Either way, it's a wonderful way to spend a few minutes every day (or an entire afternoon binge session).—Shawn Powers

The **Free Software Foundation**, GNU Project and MIT's SIPB invite you to

LIBRE PLANET 2016

"Fork the System"

A conference for everyone who loves free software

Snowden

Sandler

Stallman

Sessions by

+ Hackers
+ Activists
+ Beginners
+ Users
+ Writers
+ Artists

March 19 & 20 at MIT. Students and FSF members attend gratis.

libreplanet.org/conference

Non-Linux FOSS: CreateUserPkg

For Linux users, scripting user installation is fairly simple. It's possible, but not quite as simple with OS X. Thanks to Per Olofsson, it's possible to distribute user accounts as installable packages that are as simple as a double-click to install.

Managing user accounts is something all sysadmins have to do, and with CreateUserPkg, it's simple to install and distribute. It also includes the ability to create auto-login users, which is convenient, but it does add security concerns since the password can't be encrypted in the PKG file if it's going to be activated as an automatically logged in account. Like all powerful tools, this one can be dangerous, so use only as appropriate!

Check out the program or its source code at http://magervalp.github.io/CreateUserPkg.

—Shawn Powers

13th Annual

2016 HPC for Wall Street – Cloud & Data Centers Show & Conference

April 4, 2016 (Monday)

Roosevelt Hotel, NYC

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

Plan to attend. Low-Cost Conference Program and Free Show.

2016 Capital Markets are coming to the 2016 HPC for Wall Street.

All-Star Conference program for 2016.

Plan to attend the largest meeting of HPC, Cloud, Big Data, Data Centers, Virtualization, Low Latency for the Capital Markets.

See the program from 2015.

The 2016 program will have the same all-star lineup of speakers.

Location. Location. Location. The Roosevelt is next to Grand Central Station and within walking distance of JPMorgan Chase, Deutsche Bank, Morgan Stanley, NASDAQ – all in midtown.

Register online today: www.flaggmgmt.com/linux

2015 Sponsors

IBM, Cisco, redhat, hp, lenovo, ORACLE, Juniper Networks, ARISTA, NYC NoSQL, Linux Journal, HPCwire, TRADERS, datanami, Linux New Media – The Pulse of Open Source, Money management executiv, Enterprise Tech, Integration developer news, WSTA Wall Street Technology Association

www.flaggmgmt.com/linux

Show Hours: Mon, April 4	8:00 - 4:00
Conference Hours: Mon, April 4	8:30 - 4:50

Show & Conference:
Flagg Management Inc
353 Lexington Avenue, New York 10016
(212) 286 0333 fax: (212) 286 0086
flaggmgmt@msn.com

The all-star lineup of speakers from HPC 2015

Dave Weber
Global Financial Services Director, Lenovo

Ken Barnes
SVP Corp Dev, Options Information Technology

Bernard S Donefer
Associate Director, Baruch College

Mike Blalock
Global Sales Director, Intel

Andy Bach
Chief Architect, Financial Service, Juniper Networks

Jeffrey M. Birnbaum
Founder and CEO, 60East Technologies

Dino Vitale
TD Securities

Harvey Stein
Head of Credit Risk Modeling, Bloomberg

Fadi Gebara
Sr Manager, IBM Research

Terry Keene
CEO, iSys

Rob Krugman
VP Digital Strategy, Broadridge Fin Sols

Lee Fisher
VP Marketing, Redline Trading Solutions

Jeremy Eder
Perf Engineering, Red Hat

Matt Smith
Sol Architect, Red Hat

David B. Weiss
Sr Analyst, Aite

Rick Aiere
Architect Specialty, AIG

Shagun Bali
Analyst, TABB Group

Jeffrey Scheel
Senior Technical Staff, IBM Linux Tech Center

Ed Turkel
Mgr WW HPC Mkting, Hewlett-Packard

Charles Milo
Enterprise Technical Specialist, Intel

Alex Tsariounov
Principal Architect - Adv. Platforms, London Stock Exchange

Ugur Arslan
Quantative Analyst

Davor Frank
Sr Solutions Architect, Solarflare

Phil Albinus
Editor, Traders Magazine, SourceMedia

David Malik
Sr Director, Advanced Services, Cisco Systems

Russ Kennedy
SVP of Product Strategy, Cleversafe

Ryan Eavy
Exec Dir, Architecture, CME Group

Markus Flierl
VP Software Dev, Oracle

Nick Ciarleglio
Distinguished Syst. Engineer, FSI Product Mgr Arista Networks

Symbolic Math on Android

For this article, I'm returning to portable science software on Android. In a previous article, I looked at a program called xcas/giac. This program is an open-source engine that is used to handle symbolic manipulation of mathematical equations. Because it is open source, it has been ported to several different platforms. Because Android's core is really Linux, a port to the Android platform has been made, and it's available on the Google Play store. Installation is as easy as a quick search on the store and clicking install.

When you first start Xcas Pad, it asks you to enable the keyboard included with the application. It takes you to the Language and Input section of the settings so you can activate the keyboard. When you finish, click the back button and go back to Xcas Pad's main screen. The main screen has four tabs along the top where you can access the

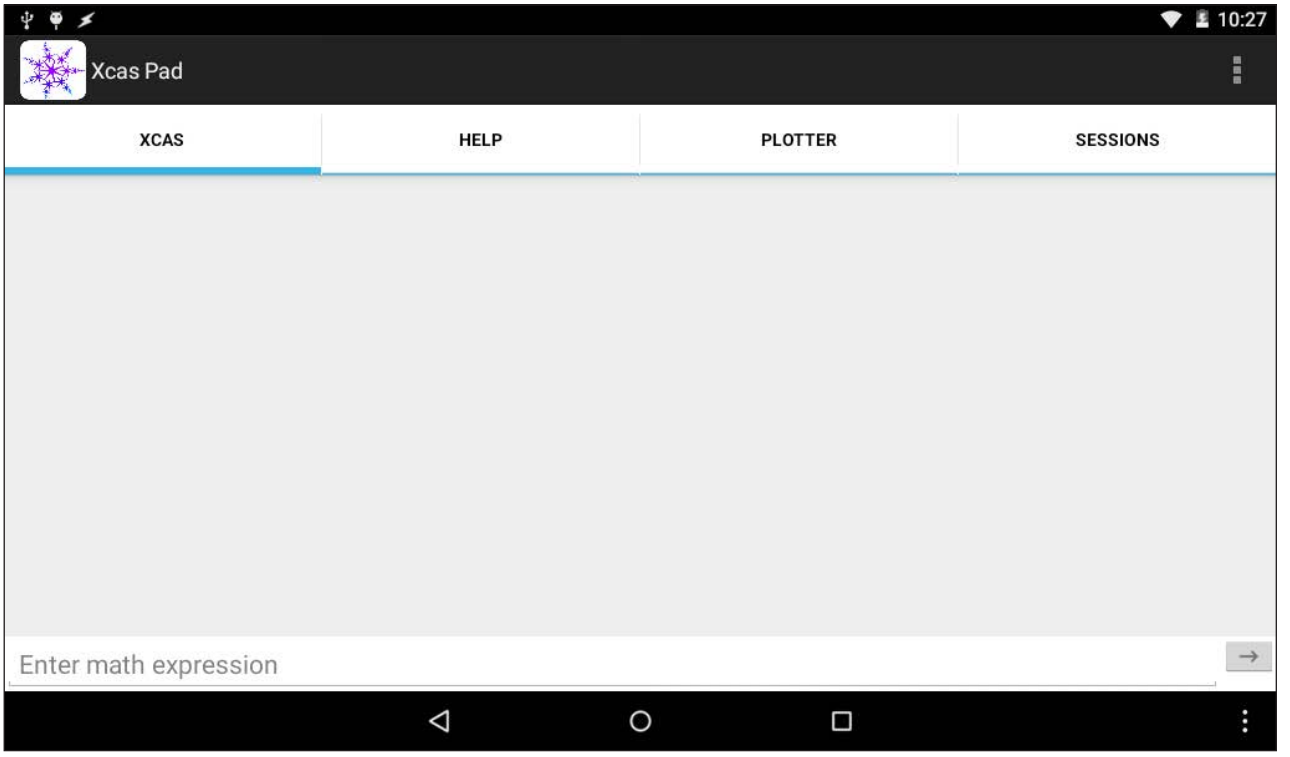


Figure 1. The main screen is a worksheet where you can start working right away.

main worksheet, a help screen, a plotter screen and a session pane. At the bottom of the xcas pane is an entry line where you can enter the individual xcas commands. The output from each command is displayed in the main portion of the xcas pane. As an example, you can find the derivative of the equation x^3–x with the command:

```
diff(x^3-x)
```

Once you enter the equation, you either can tap the done button on the keyboard, or if you tap the back button, you also can tap the enter button at the far right side of the entry line. You then will see a pretty-printed version of your command entry and the results line immediately below it (Figure 2).

If you tap on the entered command in the display pane, it will be copied and pasted into the entry line, ready for you to edit. You also can tap on the result line to get it copied and pasted into the entry line so you can use it in the next step of your calculation. This is very useful, especially when you are doing discovery-level work.

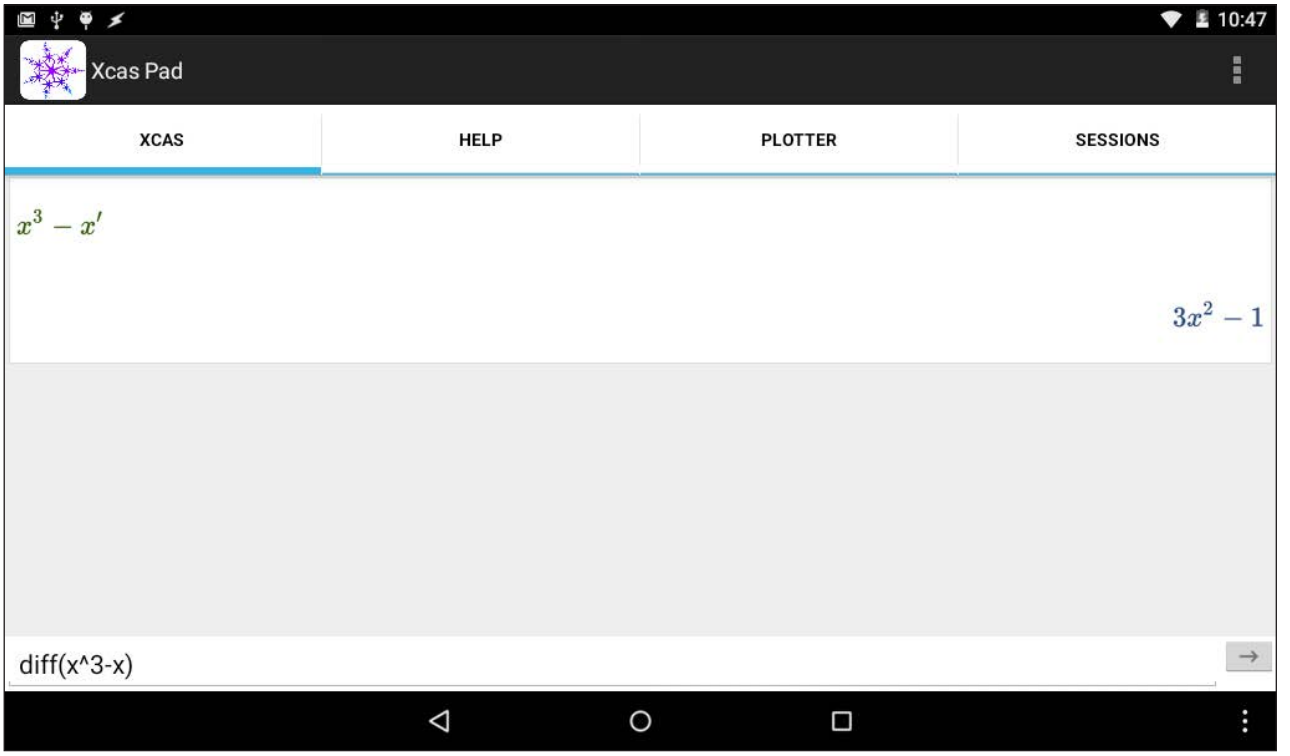


Figure 2. The commands and results are displayed in pretty print on the main panel.

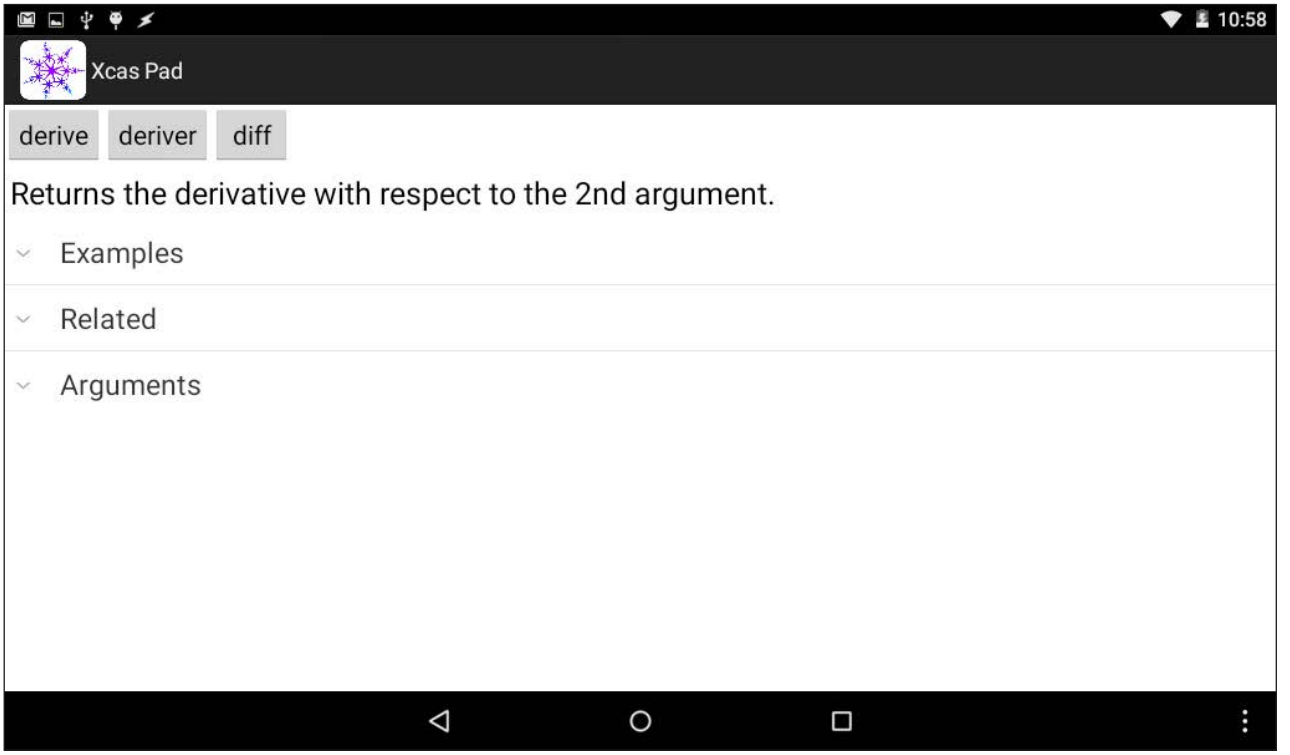


Figure 3. Help information is available for the commands within Xcas Pad.

Xcas is a very large system, however, with many different commands. Part of the problem is trying to find exactly the command you need to use. Tapping on the Help tab at the top of the screen brings up the help pages available within Xcas Pad. At the top, you can enter a search string to narrow the list a bit. You can find a description of the above example command by searching for the string "diff". Tapping the entry for "diff" pulls up a help page with a short description, a list of examples, related commands and the arguments for the command.

At the very top of the help page is a list of command names that are aliases and equivalent to each other. Tapping on one of them enters that command name into the entry line of the main worksheet tab.

The list of examples is especially useful. You can tap on one of the examples available, and it will be copied into the entry line of the main worksheet tab. This is a good way to get a starting point for some calculation that you need to do, leaving you with just having to do some edits before you are doing useful work.

One thing to remember is that Xcas Pad is like most other symbolic

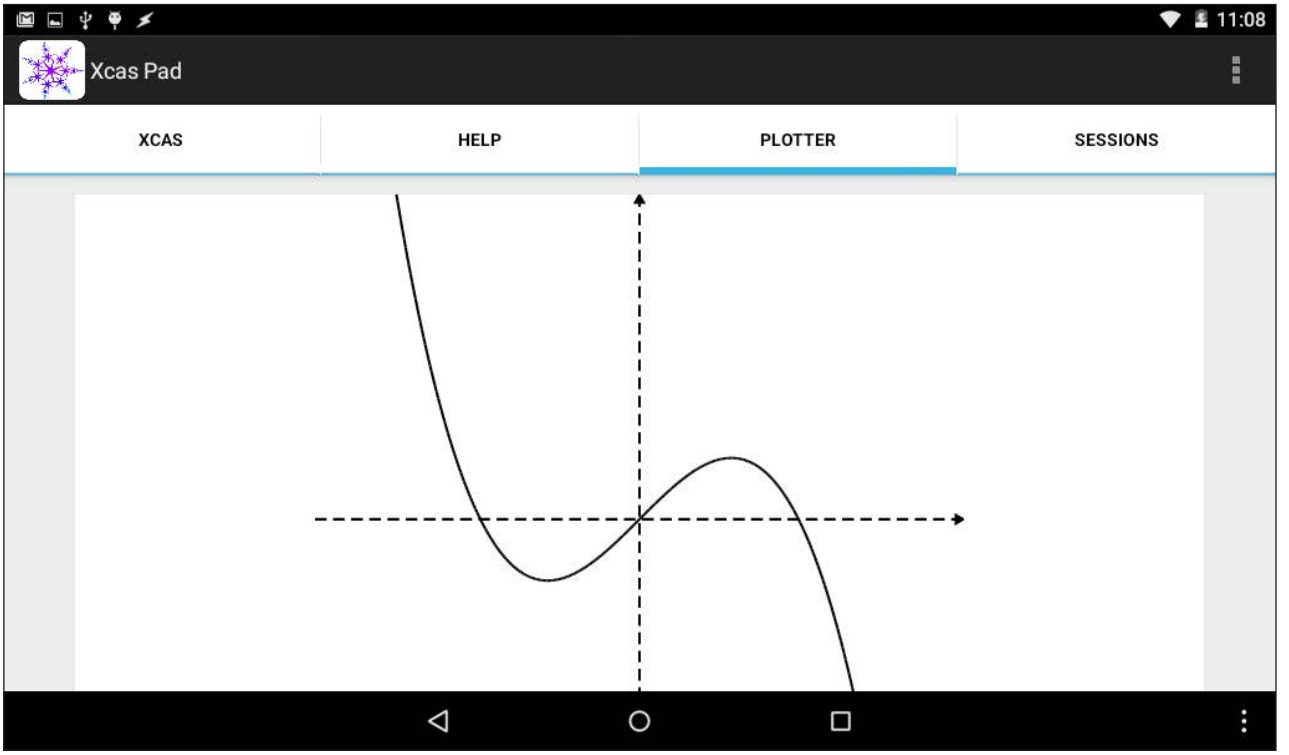


Figure 4. Executing a plot command pops open the plotter tab.

mathematical programs that are used in that the commands are run sequentially. This means if you want to rerun an earlier command, it will be rerun again based on the current state of the engine. This might be different from one run to another based on what you have been doing between the two runs. For example, you may have rewritten a function that is used within the command in question.

You also can do plotting within Xcas Pad. Doing a search for "plot" in the help page will bring up a rather large number of available commands. Scrolling down to the command plot and tapping it will give you a list of simple examples for basic plots to see what the plots can look like. Entering the following example gives the plot shown in Figure 4.

```
plot(x-1/6*x^3,x)
```

Plots also are pretty-printed and look fairly nice. If you just accept the defaults, you will get bare axes with no labels. Unfortunately, there is not as much customizability available as there is with the

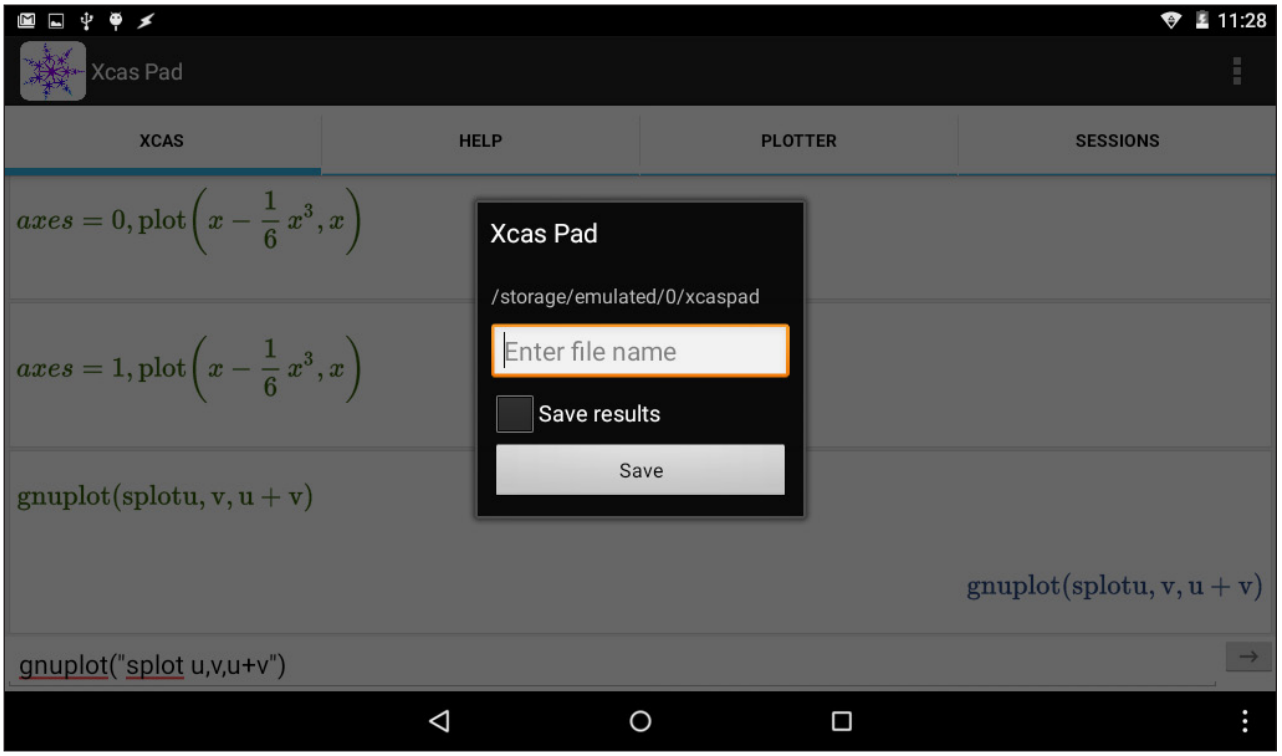


Figure 5. The save session window allows you to enter a filename on your Android device.

desktop versions, so you are kind of stuck with just creating and looking at basic plots. This still can be extremely useful when you are trying to figure out what a particular equation is doing.

Let's say you've been working on some problem for the last hour and want to save your work. These worksheets are called sessions within Xcas Pad. In order to save your current worksheet, you need to tap the option button in the top right corner to get a pull-down menu. From this menu, tapping the entry Save Session brings up a new window where you can enter a filename to use.

There is a check box where you can select whether you also want to save off the results along with the commands from your worksheet. You can reload these saved sessions from the session tab. When you click on it, it gives you a file and directory listing for the default "home" directory on your Android. Any sessions you saved from Xcas Pad will be in the xcaspad sub-directory. You also can copy over other xcas-saved files from work you may have done on your desktop.

When you select one of these session-saved files, you only have

the option of running it as a script. This is one major deficiency right now. If you want to make changes to a file before running it, you need to open the session file in a text editor first and make your edits there. Then you can open and run it within Xcas Pad. There are many very good text editors available on Android, so that shouldn't be a blocking problem.

Now you have another tool to help you get some heavy-duty science done on the go. With just your phone or tablet, you can work on your next big idea wherever you like. And if you use a file sharing service, such as Dropbox or Google Drive, you simply can pick up your work from the office wherever you have a few minutes to spare.

—Joey Bernard

RETURN TO CONTENTS

PREVIOUS
UpFront

NEXT
Reuven M. Lerner's
At the Forge

Tune Up Your Databases!

My last full-time job was manager of a university's database department. Ironically, I know very, very little about databases themselves. I'm no longer in charge of college databases, but I still do have a handful of MySQL servers that run my various Web applications. Apart from `apt-get install`, I have no idea how to make databases work. Thankfully, help is available.

```
[spowers@docboy:~$ perl mysqltuner.pl
[Please enter your MySQL administrative login: root
[Please enter your MySQL administrative password:  >>  MySQLTuner 1.6.2 - Major Hayden
 <major@mhtx.net>
 >>  Bug reports, feature requests, and downloads at http://mysqltuner.com/
 >>  Run with '--help' for additional options and output filtering
[--] Skipped version check for MySQLTuner script
[OK] Currently running supported MySQL version 5.5.46-0ubuntu0.14.04.2
[OK] Operating on 64-bit architecture

-------- Storage Engine Statistics -------------------------------------------
[--] Status: +ARCHIVE +BLACKHOLE +CSV -FEDERATED +InnoDB +MRG_MYISAM
[--] Data in MyISAM tables: 300K (Tables: 52)
[--] Data in InnoDB tables: 31M (Tables: 28)
[!!] Total fragmented tables: 35

-------- Security Recommendations  -------------------------------------------
[OK] There are no anonymous accounts for any database users
[OK] All database users have passwords assigned
[!!] User 'xbmc@%' has user name as password.
[!!] User 'xbmc@%' hasn't specific host restriction.
[--] There are 605 basic passwords in the list.
```

MySQLTuner is a Perl script that checks your local (or remote) MySQL server and gives recommendations for improving security and performance. It does not edit files or actually make changes to the server, but it does give a very lengthy list of recommendations. If you (like me) are the sort of person who just tends to copy/paste database setup instructions, running MySQLTuner is a really good idea.

You can download your copy at http://mysqltuner.com. Be sure to read the documentation to get the most use out of the program. And, if you discover security problems like the ones shown in my screenshot? Fix them!

Thanks to its ability to help improve and secure MySQL servers that otherwise might be vulnerable, MySQLTuner gets this month's Editors' Choice award. If you're imperfect like me, download a copy today and fine-tune your databases!—Shawn Powers

RETURN TO CONTENTS

Analyzing Data

Want to analyze Apache logfiles using the latest data science techniques? Start off by importing and cleaning them.

REUVEN M. LERNER

Reuven M. Lerner trains companies around the world in Python, PostgreSQL, Git and Ruby. His ebook, "Practice Makes Python", contains 50 of his favorite exercises to sharpen your Python skills. Reuven blogs regularly at http://blog.lerner.co.il and tweets as @reuvenmlerner. Reuven has a PhD in Learning Sciences from Northwestern University, and he lives in Modi'in, Israel, with his wife and three children.

PREVIOUS
Editors' Choice

NEXT
Dave Taylor's
Work the Shell

MY FIRST WEB-RELATED JOB WAS IN 1995, developing Web applications for a number of properties at Time Warner. When I first started there, we had a handful of programmers and managers handling all of the tasks. But over time, as happens in all growing companies and organizations, we started to specialize. One of the developers took charge of the logfiles—storing them and then performing some basic analysis on them.

Although I recognized that this work was important, it took years for me to realize that in some ways, his job was more important to the business than the applications that I was writing. The developer who worked on these logfiles, and who analyzed them for our bosses, made it possible to know who was

using our system, what they were viewing and using, where they came from, and what correlations we could find among the different data points provided by the logs. Sure, we were providing the content and the applications that brought people to the site, but it was the person analyzing the logfiles who was ensuring that our work was paying for itself and meeting our business goals.

During the past decade, I've come to appreciate the need for such analysis even more, as the Web has exploded in popularity, as businesses have learned to use such data to increase profitability and as data science has become a growing field. We're now drowning in data, and being able to make sense of it using analytical tools and libraries is more important than ever.

In this article, I start an exploration of data science using Python, and how you can take something as ordinary as an Apache logfile and extract information from it to understand your visitors better and what they do. In upcoming articles, I plan to cover how you can use data science methods to analyze this logfile in a number of different ways, gaining insights into the raw data it provides and answering questions about your Web application. I'll describe how this analysis also can be presented to your managers and clients, providing powerful visualizations of the analysis you've performed.

Data Science and Python

I studied something called "learning sciences" in graduate school. While I was there, we often would joke that any discipline that includes the word "science" in its name is probably not a real science. Regardless of whether data science is a "real" science, it is a large, important and growing field—one that allows businesses to make decisions based on the data they have gathered. The more data, and the more intelligently you use that data, the better you'll be able to predict your users' and customers' wants and needs.

Data science has been defined loosely as the intersection of programming and statistics, applied to a particular domain. You gather some data and then use statistical methods to answer questions the data might be able to answer. A background in statistics can be helpful, not only because it'll show you the types of analysis you might want to apply,

but also because it gives you a healthy sense of skepticism regarding the correlations you find. Did you really discover that your Web site is popular only with people in a particular area of the world? Or, did you just advertise it heavily in one part of the world, influencing who is more likely to visit?

You can start a data science project by asking a question, or you can start to explore the data in a variety of ways, hoping you will find an interesting correlation. Regardless, data science expects you to know a variety of methods from which you can choose one or more that are appropriate for answering your questions. You then apply the methods, using statistical tests to determine whether your answers are significant—that is, whether they merely could have been random.

Python, long used by system administrators, Web developers and researchers, is an increasingly popular choice among people working in data science. This is the result of several factors coming together. First, Python has a famously shallow learning curve, allowing non-programmers to get started and do things in a short amount of time.

Second, Python works easily and cleanly with a variety of data formats and databases. Thus, whether your raw data is in a text file, relational database, NoSQL database, CSV file, Excel file or something more unusual, the odds are very good that Python will be able to read from it easily and quickly.

Third, a number of libraries for analyzing data in Python, such as NumPy, SciPy and Matplotlib, have been under development for many years, providing a terrific balance of usability, expressive power and high-efficiency execution. In recent years, the Pandas library has added an even more useful layer on top of this.

Finally, the development of IPython, now known as Jupyter, has been nothing short of revolutionary, providing developers and data scientists with the ability to interact with their programs and data (as with a traditional REPL), but to do so on a Web page that easily can be shared among collaborators or sent via e-mail for off-line usage and analysis. Indeed, I now use the IPython Notebook in all of my Python courses. Not only does it provide me with a high-quality way to display the live coding demos I do during my classes, but I then can send the document to my students, who can replay, modify and better understand what I discussed in class.

Importing Data

The first step of any data science project is to get the data ready. In the case of wanting to analyze Apache logfiles, you might think it's enough just to get the file. However, Pandas—the Python library that I'll be using to analyze the data for this example—is like many other data science systems (for example, the R language) that expects the data to be in CSV (comma-separated values) format. This means you'll need to convert the logfile into a CSV file, in which the fields from the Apache log are converted into fields in CSV.

Performing such a transformation is actually quite straightforward in Python. Here is a sample from the Apache logfile from my blog:

```
122.179.187.119 - - [22/Jan/2016:11:57:26 +0200] "GET
 ➥/wp-content/uploads/2014/10/3D_book.jpg HTTP/1.1" 200 302222
 ➥"http://blog.lerner.co.il/turning-postgresql-rows-arrays-array/"
 ➥"Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like
 ➥Gecko) Chrome/47.0.2493.0 Safari/537.36"
122.179.187.119 - - [22/Jan/2016:11:57:27 +0200] "POST
 ➥/wp-admin/admin-ajax.php HTTP/1.1" 200 571
 ➥"http://blog.lerner.co.il/turning-postgresql-rows-arrays-array/"
 ➥"Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like
 ➥Gecko) Chrome/47.0.2493.0 Safari/537.36"
54.193.228.6 - - [22/Jan/2016:11:57:29 +0200] "GET
 ➥/category/python/feed/ HTTP/1.1" 200 25856 "-" "Digg Feed
 ➥Fetcher 1.0 (Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_1)
 ➥AppleWebKit/534.48.3 (KHTML, like Gecko) Version/5.1
 ➥Safari/534.48.3)"
```

Each line has the following components:

- IP address from which the request was made.
- Two fields (represented with - characters) having to do with authentication.
- The timestamp.

- The HTTP request, starting with the HTTP request method (usually `GET` or `POST`) and a URL.
- The result code, in which 200 represents "OK".
- The number of bytes transferred.
- The referrer, meaning the URL that the user came from.
- The way in which the browser identifies itself.

This information might seem a bit primitive and limited, but you can use it to understand a large number of factors better having to do with visitors to your blog. Note that it doesn't include information that JavaScript-based analytics packages (for example, Google Analytics) can provide, such as session, browser information and cookies. Nevertheless, logfiles can provide you with some good basics.

Two of the first steps of any data science project are 1) importing the data and 2) cleaning the data. That's because any data source will have information that's not really useful or relevant for your purposes, which will throw off the statistics or add useless bloat to the data you're trying to import. Thus, here I'm going to try to read the Apache logfile into Python, removing those lines that are irrelevant. Of course, what is deemed to be "irrelevant" is somewhat subjective; I'll get to that in just a bit.

Let's start with a very simple parsing of the Apache logfile. One of the first things Python programmers learn is how to iterate over the lines of a file:

```
infile = 'short-access-log'
for line in open(infile):
    print(line)
```

The above will print the file, one line at a time. However, for this example, I'm not interested in printing it; rather, I'm interested in turning it into a CSV file. Moreover, I want to remove the lines that are less interesting or that provide spurious (junk) data.

In this way, I have turned each line of my logfile into a Python dictionary, with each key-value pair in the dictionary referencing a different field from my logfile's row.

In order to create a CSV file, I'm going to use the `csv` module that comes with Python. One advantage of this module is that it can take any separator; despite the name, I prefer to use tabs between my columns, because there's no chance of mixing up tabs with the data I'm passing.

But, how do you get the data from the logfile into the CSV module? A simple-minded way to deal with this would be to break the input string using the `str.split` method. The good news is that split will work, at least to some degree, but the bad news is that it'll parse things far less elegantly than you might like. And, you'll end up with all sorts of crazy stuff going on.

The bottom line is that if you want to read from an Apache logfile, you'll need to figure out the logfile format and read it, probably using a regular expression. Or, if you're a bit smarter, you can use an existing library that already has implemented the regexp and logic. I searched on PyPI (the Python Package Index) and found clfparser, a package that knows how to parse Apache logfiles in what's known as the "common logfile format" used by a number of HTTP servers for many years. If the variable `line` contains one line from my Apache logfile, I can do the following:

```
from clfparser import CLFParser
infilename = 'short-access-log'
for line in open(infilename):
    print CLFParser.logDict(line)
```

In this way, I have turned each line of my logfile into a Python dictionary, with each key-value pair in the dictionary referencing a different field from my logfile's row.

Now I can go back to my CSV module and employ the DictWriter class that comes with it. DictWriter, as you probably can guess, allows you to output CSV based on a dictionary. All you need to do is declare the fields you want, allowing you to ignore some or even to set their order in the resulting CSV file. Then you can iterate over your file and create the CSV.

Here's the code I came up with:

```
import csv
from clfparser import CLFParser

infilename = 'short-access-log'
outfilename = 'access.csv'

with open(outfilename, 'w') as outfile, open(infilename) as infile:
    fieldnames = ['Referer', 'Useragent', 'b', 'h', 'l', 'r', 's',
     ➥'t', 'time', 'timezone', 'u']
    writer = csv.DictWriter(outfile, fieldnames=fieldnames,
     ➥delimiter='\t')
    writer.writeheader()

    for line in infile:
        writer.writerow(CLFParser.logDict(line))
```

Let's walk through this code, one piece at a time. It's not very complex, but it does pull together a number of packages and functionality that provide a great deal of power in a small space:

- First, I import both the `csv` module and the CLFParser class from the `clfparser` module. I'm going to be using both of these modules in this program; the first will allow me to output CSV, and the second will let me read from the Apache logs.

- I set the names of the input and output files here, both to clean up the following code a bit and to make it easier to reuse this code later.

- I then use the `with` statement, which invokes what's known as a "context manager" in Python. The basic idea here is that I'm creating two file objects, one for reading (the logfile) and one for writing (the CSV file). When the `with` block ends, both files will be closed, ensuring that no data has been left behind or is still in a buffer.

- Given that I'm going to be using the CSV module's DictWriter, I need to indicate the order in which fields will be output. I do this in a list; this list allows allow me to remove or reorder fields, should I want to do so.

- I then create the csv.DictWriter object, telling it that I want to write data to outfile, using the field names I just defined and using tab as a delimiter between fields.

- I then write a header to the file; although this isn't crucial, I recommend that you do so for easier debugging later. Besides, all CSV parsers that I know of are able to handle such a thing without any issues.

- Finally, I iterate over the rows of the access log, turning each line into a dictionary and then writing that dictionary to the CSV file. Indeed, you could argue that the final line there is the entire point of this program; everything up to that point is just a preface.

Cleaning the Data

You've now seen that you can import the data from another form into a CSV file, which is one of the most common formats used in data science. However, as I mentioned previously, one of the key things that you also need to do is clean the data; analyzing bogus data will give you bogus results.

So, what sort of data here needs to be cleaned?

One obvious candidate is to remove anything that wasn't a real human. Perhaps you're interested in finding out what Web crawlers, such as those from Google and Yahoo, are up to. But it's more likely that you want to know what humans are doing, which means removing all of those robots.

Of course, this raises the question of how you can know whether a

request is coming from a robot. As humans, you can examine the User-agent string and make an educated guess. But given that you're trying to remove all of the robots, and that new ones constantly are being added, something automatic would be better.

There's no perfect answer to this, but for the purposes of this article, I decided to use another Python module from PyPI, albeit one that's a bit out of date—one known as robot-detection. The idea is that you import this module and then use the `is_robot` function on the `Useragent` field. If it's a robot, `is_robot` will return True. Here's my revised code:

```
import csv
from clfparser import CLFParser
from collections import Counter
import robot_detection

infilename = 'medium-access-log.txt'
outfilename = 'access.csv'
robot_count = Counter()

with open(outfilename, 'w') as outfile, open(infilename) as infile:
    fieldnames = ['Referer', 'Useragent', 'b', 'h', 'l', 'r', 's',
     ➥'t', 'time', 'timezone', 'u']
    writer = csv.DictWriter(outfile, fieldnames=fieldnames,
     ➥delimiter='\t')
    writer.writeheader()

    for line in infile:
        d = CLFParser.logDict(line)
        if robot_detection.is_robot(d['Useragent']):
            robot_count[d['Useragent']] += 1
        else:
            writer.writerow(d)
```

The above code is mostly unchanged from the previous version; the two modifications are that I'm now using `robot_detection` to filter out the robots, and I'm using the Python `Counter` class to keep track of how

many times each robot is making a request. This alone might be useful information to have—perhaps not now, but in the future. For example, from examining the most recent 100,000 requests to my blog, I found that there were more than 1,000 requests from the "domain re-animator bot", something I hadn't even heard of before.

Given that I'm currently concentrating on user data, filtering out these bot requests made my data more reliable and also a great deal shorter. Out of 100,000 records, only 27,000 were from actual humans.

Conclusion

The first step of any data-analysis project is to import and clean the data. Here, I have taken the data and put it into CSV format, filtering out some of the lines that are of less interest. But this is just the start of my analysis, not its end. Next month, I'll explain how you can import this data into Python's Pandas package and start to analyze the logfile in a number of different ways.■

RESOURCES

Data science is a hot topic, and many people have been writing good books on the subject. I've most recently been reading and enjoying an early release of the *Python Data Science Handbook* by Jake VanderPlas, which contains great information on data science as well as its use from within Python. Cathy O'Neil and Rachel Schutt's slightly older book *Doing Data Science* is also excellent, approaching problems from a different angle. Both are published by O'Reilly, and both are great reads.

To learn more about the Python tools used in data science, check out the sites for NumPy (http://numpy.org), SciPy (http://SciPy.org), Pandas (http://pandas.pydata.org) and IPython (http://IPython.org). There is a great deal to learn, so be prepared for a deep dive and lots of reading.

Python itself is available from http://python.org, and the PyPI package index, from which you can download all of the packages mentioned here, is at http://PyPI.python.org.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

Fancy Tricks for Changing Numeric Base

bconvert—ways to convert numeric bases from the command line.

DAVE TAYLOR

Dave Taylor has been hacking shell scripts since the dawn of the computer era. Well, not really, but still, 30 years is a long time! He's the author of the popular *Wicked Cool Shell Scripts* and *Teach Yourself Unix in 24 Hours* (new edition just released!). He can be found on Twitter as @DaveTaylor and more generally at his tech site: www.AskDaveTaylor.com.

PREVIOUS
Reuven M. Lerner's
At the Forge

NEXT
Kyle Rankin's
Hack and /

IN MY LAST ARTICLE, I did what business writers would call a "deep dive" into `getopts` and different ways to parse starting flags and arguments in shell scripts. I know you've tested that in your latest programs, which is great, because it's something that you could find yourself using quite a lot.

In this article, I'm covering something that's a bit more abstruse: converting numeric bases within shell scripts. There are really four commonly used numeric bases to consider: binary, octal, decimal and hexadecimal. You're used to working in base-10, so 10 = 1 * 10**1 + 0 and 100 = 1 * 10**2 + 0 * 10**1 + 0.

That maps to other numeric bases, so 1010 base-2 or binary is really 1 * 2**3 + 0 * 2**2 + 1 * 2**1 + 0 or

Hexadecimal presents a different challenge because a base-16 numbering system doesn't fit neatly into our Arabic numerals 0, 1, 2, ... 9.

8 + 0 + 2 + 0 = 10 decimal. Octal is the same thing, so 33 base-8 converts to decimal as 3 * 8**1 + 3 = 27.

Hexadecimal presents a different challenge because a base-16 numbering system doesn't fit neatly into our Arabic numerals 0, 1, 2, ... 9. "Hex", as it's known informally, adds A, B, C, D, E and F, so that the decimal value 10 is represented in Hex as "A". That's where the math gets interesting, so 33 base-16 = 3 * 16**1 + 3 = 48 + 3 = 51.

The long, complicated way to create a base conversion utility is therefore to disassemble every value given and apply the translation shown, then have an internal value that's a common base (probably base-10), then have another routine that converts the common base to the desired output base.

There are smarter ways to do this, as I'll discuss, but for now, let's look at the `bc` command, which supports users specifying both the input and output numeric bases. `bc`, the binary calculator, is a bit tricky to work with as it's an old-school UNIX command. As I discuss at length in my book *Wicked Cool Shell Scripts*, the most common way to work with the crude but interactive `bc` program is to use `echo` to send it the commands needed, as demonstrated here:

```
$ echo '333 * 0.35' | bc
116.55
```

Useful (particularly since `expr` and `$(( ))` can't work with floats and decimal values), but where this gets really interesting is with those input and output numeric bases.

Let's say I want to confirm a conversion I listed earlier, by converting 33 hex into decimal. This is easily done:

```
$ echo 'ibase=16; 33' | bc
51
```

That's simple. Now, let's do something bigger and more complicated:

```
$ echo 'ibase=16; FEF33D9' | bc
267334617
```

`ibase` is the input numeric base. The output base is specified as `obase`. And that's it—easy enough!

So let's take the same hex value as input but force the output to octal instead of the default decimal:

```
$ echo 'ibase=16; obase=8; FEF33D9' | bc
1773631731
```

Would you rather work in binary? You can do that too:

```
$ echo 'ibase=16; obase=2; FEF33D9' | bc
111111101111001100111011001
```

That's a lot of ones and zeroes, for sure. It makes me think of *Interstellar*, but that's another article entirely!

Armed with this knowledge, it's pretty easy to push out a rudimentary shell script that converts between any of binary, octal, decimal and hexadecimal:

```
ibase=10; obase=10      # set up defaults
usage() {
  echo "Usage: $(basename $0) -i base -o base value" 1>&2
  echo "  where base can be 2, 8, 10 or 16." 1>&2
  exit 1
}
while getopts "i:o:" value ; do
  case "$value" in
    i) ibase=$OPTARG
       (( ibase == 2 || ibase == 8 || ibase == 10 ||
          ibase == 16 )) || usage
       ;;
    o) obase=$OPTARG
```

```
        (( obase == 2 || obase == 8 || obase == 10 ||
           obase == 16 )) || usage
        ;;
    *) usage ;;
  esac
done
shift $(( OPTIND - 1 ))

echo Converting $1 from base-$ibase to base-$obase\:
echo "obase=$obase; ibase=$ibase; $1" | bc
exit 0
```

Almost the entire program is involved with parsing and checking input values, which isn't that uncommon with well written shell scripts. Notice some shortcuts I include in the script too, notably the test structure:

```
(( condition || condition )) || usage
```

This is the same as saying "if not condition1 and not condition 2 ; then ; usage", just more succinct. Also, as I discussed in my last article, note the use of `OPTARG` to get the argument value and `OPTIND` with the `shift` command to axe all of the parameters so that `$1` will be the value to convert.

A few quick runs of the program reveal that it's working fine:

```
$ bconvert.sh -i 16 33
Converting 33 from base-16 to base-10:
51
$ bconvert.sh -i 16 -o 2  33
Converting 33 from base-16 to base-2:
110011
$ bconvert.sh -i 2 -o 16 110011
Converting 110011 from base-2 to base-16:
33
```

Notice the last two examples demonstrate the mirror function of

converting between 33 base-16 and 110011 base-2. It works!

A common numeric notation in the Linux world is to recognize that numbers prefaced with a zero are octal, and those prefaced with "0x" are hexadecimal. (Binary isn't particularly useful so it's not included in the common notation.) Here are a few examples: 0700, 0xFFc39. You could modify the script to accept these as inputs and infer the appropriate base, but I'm going to leave that as an exercise for you, dear reader.

There's another way you can convert values without involving `bc`—by utilizing the `printf` command-line program. If you know C programming, you're already familiar with `printf()` and `scanf()`, but unfortunately, only the output function is available at the shell command line. Usage is quite similar, however, as you can see in this quick example:

```
$ printf "> %d <\n" 42
> 42 <
```

In this case, the format string (argument #1) details the desired output, with `%d` indicating that a decimal value will be printed, then argument 2 is that value, 42.

Where this gets interesting is because you actually can use other values in the format string to force octal or hexadecimal:

```
$ printf "octal: %o\nhex: %x\n" 42 42
octal: 52
hex: 2a
```

Because of the notational convention mentioned earlier for non-decimal numbers in the shell, you also can specify an octal or hexadecimal value too:

```
$ printf "%o\n" 0500
500
```

Wait, what happened in that last example? It's simple: I specified that I wanted octal (base-8) output, but by using the leading zero, I also indicated that I was specifying a value in octal too. Ergo, 0500 = 500.

That's nice, but no binary, which is a definite limitation.

But, I'm not done yet. There's one more way you can convert values, and it's actually directly within the shell. It turns out that using the `$(( ))` notation, you actually can specify a numeric base for numbers!

This is something I stumbled across recently, having had no idea that this was even a capability of the shell, but check this out, a quick conversion of 33 base-16 to decimal:

```
$ echo $((16#33))
51
```

Not only that, but the leading zero and leading "0x" are both valid too:

```
$ echo $(( 0xFF ))
255
```

If you don't care about binary values, you can see that there are three completely different ways to convert numeric bases from within a shell script. Now take what I've shown here and do something really slick!

In a future article, I'll explore some other shortcuts for conditional statements that let you skip the mundane "if condition ; then XX else XX fi" notational sequence.■

The bconvert.sh script is available for download at http://www.linuxjournal.com/files/linuxjournal.com/code/bconvert.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

GRUB Boot from ISO

A few tweaks to GRUB2 make booting from an ISO relatively easy.

KYLE RANKIN

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.

PREVIOUS
Dave Taylor's Work the Shell

NEXT
Shawn Power's The Open-Source Classroom

I RECENTLY WORKED ON A PROJECT to add an OEM-style rescue partition to a computer. Where most OEM installs have a custom program that just rewrites an install image over the top of the partition, in this case, everything was based on open-source software. This rescue partition would be only a few gigabytes in size—big enough to contain the install DVD ISO image and a few preseed files to help automate the install. If users ever wanted to get back to the factory-installed version of the operating system, they could select a special option from the GRUB menu that would boot off the ISO and start the install as though they had used a USB key or DVD.

If you read some past guides on how to boot an ISO from GRUB, you will find a number of pretty complicated instructions in some cases because they were writing for older versions of GRUB. With the recent versions of

GRUB2, booting from a standard ISO is fortunately not that complicated. I base my example here off a recent Debian Stretch install DVD, but the same steps should work for other distributions and install ISOs with some tweaks.

Create the Partition

The first step is to create the rescue partition. In my case, I automated my install with a preseed script, but basically I just created a partition that was big enough to hold the install DVD. I made sure to give it a label so I could tell it apart from other disks later on and made it the second partition on the disk. Because I was doing all of this via an automated means, I ended up using `dd` to create an image of the install DVD that was currently in use as an end-of-install script and dumped it to the root of that partition in a file called install_dvd1.iso. Of course, you could just copy over your ISO directly to the disk if you do this from a system that already has an OS on it.

Build the GRUB Config

The next step was to build a GRUB config that would mount the ISO loopback and boot off the kernel and initrd file within that ISO. On Debian-based systems, you can add bash scripts that output extra GRUB configuration to /etc/grub.d/ and run `update-grub` to build a new grub.cfg file. But, you also could just edit grub.cfg directly or otherwise use your distribution's GRUB configuration scripts to add the following menu items:

```
set root='hd0,msdos2'
set isofile="/install_dvd1.iso"
menuentry "Install OS" {
 loopback loop (hd0,msdos2)$isofile
 linux (loop)/install.amd/vmlinuz vga=788 auto=true
  ➥file=/media/preseed.cfg -- quiet
 initrd (loop)/install.amd/initrd.gz
}
menuentry "Install OS (Expert)" {
 loopback loop (hd0,msdos2)$isofile
 linux (loop)/install.amd/vmlinuz vga=788 -- quiet
 initrd (loop)/install.amd/initrd.gz
}
```

Let's break this GRUB configuration down a bit. First, you set two variables: the root partition GRUB will use (the second partition on the first disk, which GRUB refers to as hd0,msdos2) and the ISO file:

```
set root='hd0,msdos2'
set isofile="/install_dvd1.iso"
```

Next are two separate GRUB menus (each are within `menuentry {}` stanzas). The first points to a preseed configuration file and starts a partially automated Debian install, while the other just boots the Debian installer with no preseed file and acts like an "Expert" mode so you can choose every install option by hand.

Up to this point, the GRUB options were pretty similar to other GRUB configurations, but here is where that changes. Next in each menu, you define a loopback device GRUB will use:

```
loopback loop (hd0,msdos2)$isofile
```

Now, whenever you refer to `(loop)` further in the config, GRUB knows to access the loopback filesystem labeled loop, which points to (hd0,msdos2)/install_dvd1.iso. The next two lines should look pretty familiar to someone who has worked with Linux GRUB configuration before, but with a twist:

```
linux (loop)/install.amd/vmlinuz vga=788 auto=true
 ➥file=/media/preseed.cfg -- quiet
initrd (loop)/install.amd/initrd.gz
```

In the first line, you define what kernel to boot and what options to pass it, and in the following line, you point GRUB to the initrd file you want it to use. The main difference here though is that you precede each file path with `(loop)` to instruct GRUB to look in that loopback filesystem for the file.

Once this configuration finds its way into grub.cfg, you should see the two new menu options in place the next time you boot. Now the first time I tried to boot the most recent Debian installer this way, I ran into a

bit of a problem. It turns out that the initrd that comes on the ISO itself does not contain the installer scripts you need to install from an ISO on a hard drive. It assumes you will boot only off a DVD or USB disk. Because of that, I discovered I had to download a different Debian installer initrd and put it on the rescue disk for things to work. I was able to find an initrd that worked at http://mirrors.kernel.org/debian/dists/stretch/main/installer-amd64/current/images/hd-media/initrd.gz.

Of course, with that new initrd.gz file at the root of my rescue partition, I had to change my GRUB config a little bit to point to it:

```
set root='hd0,msdos2'
set isofile="/install_dvd1.iso"
menuentry "Install OS" {
 loopback loop (hd0,msdos2)$isofile
 linux (loop)/install.amd/vmlinuz vga=788 auto=true
  ➥file=/media/preseed.cfg -- quiet
 initrd /initrd.gz
}
menuentry "Install OS (Expert)" {
 loopback loop (hd0,msdos2)$isofile
 linux (loop)/install.amd/vmlinuz vga=788 -- quiet
 initrd /initrd.gz
}
```

Note that the initrd lines now point to /initrd.gz. Since I already defined the root variable earlier in the config, it knows to look on hd0,msdos2 for the file. Once this new initrd was in place, I was in business, and the installer worked as expected. If you want to use this with a different installer, just make sure that once the kernel boots up, it has the ability to scan disks for an installer ISO to use.■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

Back It Up, Buster!

Remote backups aren't even remotely optional. Do it!

PREVIOUS
Kyle Rankin's Hack and /

NEXT
New Products

SHAWN POWERS

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.

IT STILL SHOCKS ME how many people don't keep backups of their files. Every time people ask me to look at their computers (almost always a malware-infected Windows machine), I ask them if they have backups before I start poking around. Invariably, the answer is no. Sometimes the computers even contain vital business data, and the closest thing to a backup is a local copy of the database in the same folder as the active copy. As someone who had a house burn down, I can tell you, off-site backups aren't just for fancy government organizations!

Things That Aren't Backups

RAID (Redundant Array of Independent Disks), as wonderful as it is, must not be mistaken as a backup. Simply put, RAID is a method for ensuring your single, local copy of data is less prone to failure. And

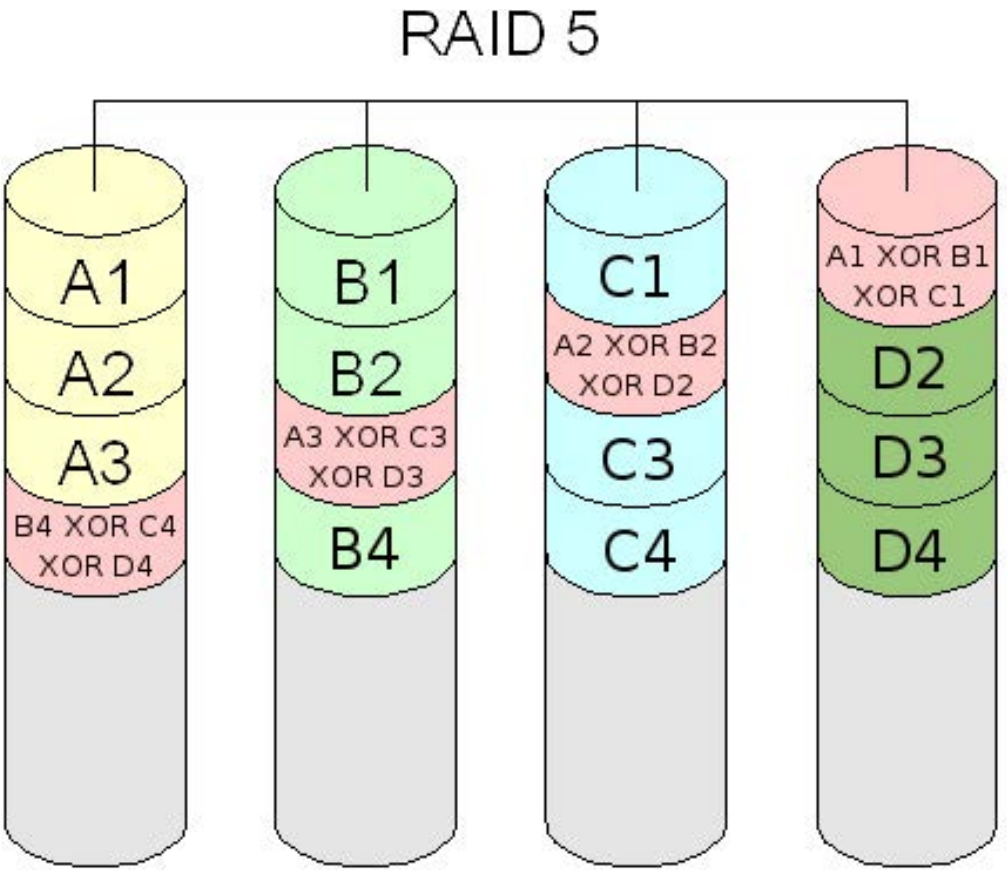


Figure 1. RAID 5 is awesome, but trust me, it's not foolproof.

that's really only for RAID levels that create mirrors or parity. Using RAID level 0, you're actually *more* likely to lose data than by storing it on a single hard drive!

I personally learned the hard way that RAID is not the same as a backup. I've been preaching it for years, but on my local NAS (Network Attached Storage) device, I implemented RAID 6. Since that meant two drives could fail and I still wouldn't lose data, I didn't bother backing up my collection of videos, including irreplaceable home videos. As luck(?) would have it, a power fluctuation during a storm caused three hard drives in my array to fail at the same time, and I lost all 24TB of storage. Most of the data I lost was just digital backups of our DVDs and Blu-rays, but about 10GB of that data was home videos. And since I didn't back it up, it's lost forever.

The moral of the story is this: RAID is awesome. You should use RAID if possible to help prevent data loss resulting from failed hardware. But, don't stop with RAID and assume your data is safe. It's not. Your data is

safe only if there is more than one copy of if. And, that's where actual backup comes into play.

Live Mirrors

Live copies of your data is a backup of sorts. The great part about live backups is in the name; they're live. The minute you create or change a file, those changes are synced to your mirror. The problem with live mirrors is that if you accidentally save over a file, or inadvertently remove a file, those changes are synced and you lose all the copies of the data in real time. Some options do automatic versioning of files, which makes your data a little more secure, but for the most part, live mirrors protect you from hardware failure, but not human error.

My favorite commercial software for live mirroring is Dropbox. It's free for a small amount of storage (probably enough for a folder full of office documents), and it does file revisioning, which helps for accidental overwrites. Unfortunately, in order to back up a significant amount of data, you have to pay annually for Dropbox storage. There are other Dropbox-like solutions, such as SugarSync, Box, Google Drive and so on. Most have a small amount of storage for free and allow extra storage for a subscription fee.

If you have multiple computers, or a computer an a server, another live mirror option is to use BitTorrent Sync. You can get the free client at http://getsync.com, and install the software on most platforms, including headless on a server. I've written about BitTorrent Sync before. The beauty is that all the data is stored on your own machines, so the only limit on space is how much storage you have on your devices.

I personally use BitTorrent Sync to keep a full, live mirror of my wife's home directory on our home server. That way, if she forgets her laptop at work, I can access any file for her, even if her laptop is shut off. She can edit the files at home, and as soon as she turns her laptop back on, BitTorrent Sync updates the files at work.

You can have as many mirrors as you want with Bittorrent Sync, so we also have a "Family" folder that we sync between all our computers. If we need to pass a file to a family member, we just drop it in the Family folder and everyone instantly has access to it. The free

Figure 2. BitTorrent Sync's free version offers unlimited data syncing.

version of BitTorrent Sync does not do file versioning, so if you make a change, the old version of your file is lost forever. That can be very convenient, but it also can be very frustrating. That's why I consider BitTorrent Sync vital, but I don't consider it a backup.

Backup Programs

There are two backup programs of which I'm particularly fond. I've written about them both in the past, but I want to highlight them again here. The first is BackupPC. It is a server-based program that creates true backups of data and keeps multiple copies of files on a rotation. For example, BackupPC will store a complete snapshot backup of every day of the week and then delete the oldest backup daily when the full week is stored. It also can be configured to keep a monthly or even annual backup snapshot as well. The best part about BackupPC is that it uses hard linking to store multiple copies of the same file without using any extra storage. The only storage increase comes with new or changed files, meaning a server can store vast numbers of "full" backups without wasting disk space.

Another great feature of BackupPC is that all backups are pulled from the server rather than being pushed from the client. This can be inconvenient for laptops that are shut off, but it means the server knows (and notifies) if it can't connect to machines in its backup list. Plus, it has an incredible Web interface that allows simple file restoration directly to the original location or downloadable via ZIP file. I love BackupPC, but it works well only over a LAN. If your house burns down, that doesn't help much.

Figure 3. CrashPlan is a commercial product, but it offers incredible free functionality.

Another favorite of mine is CrashPlan. It's a commercial, Java-based application that is so awesome, it's worth installing Java! CrashPlan does have a commercial cloud-based service that will keep your data safe on its servers. It's not free, but it is reasonable. The coolest part of CrashPlan, however, is that it allows backing up from one client to another—even from friend to friend. If you have friends who are willing to store your backups (offering to do the same for them often is a good motivator), you can have automatic, off-site backups for no cost whatsoever. Files can be restored fairly easily using the client software, and since backups are completely encrypted, your friends don't have access to the files they're storing for you.

If you don't have a friend willing to store your data, it might be worth it to ask someone to colocate a small server for you. For the cost of a cheap Raspberry Pi and an external USB drive, you can have a remote CrashPlan server logged in to your own account. It's even possible to limit the bandwidth and time of day you use, so your friend's Internet connection isn't adversely affected. Heck, if you can't find a friend, you might be able to put a small server in an outdoor shed, assuming you have Wi-Fi access and power. (It might be fun to create a solar-powered backup box for the backyard—perhaps that will be a project in the next couple months!)

Data Junkies, There's Hope!

I have a 48TB storage array in my basement. It's not full (yet), but it has many terabytes of data just sitting there waiting for something horrible to happen. And since it has happened to me before, I'm not blindly

Figure 4. My Synology 1815+ has eight drive bays populated with 6TB drives. The software is amazing, and it includes cloud syncing to Amazon Cloud Drive.

trusting my RAID 6 array for protection. The problem is, 48TB of storage is expensive, and that much cloud storage is even more expensive. I have a decent upload speed on my Internet connection, but paying for a place to store it is costly. I think the best solution would be to colocate a NAS device somewhere with fast bandwidth and a static IP. Unfortunately, I can't afford anything like that.

The cheapest storage solution I can find is Amazon Cloud Drive (http://amazon.com/clouddrive/unlimited). It surprised me, honestly, because although S3 storage is affordable, for large buckets, it's not exactly cheap. At the time of this writing, however, it's possible to buy Amazon Cloud Drive Unlimited for $60 per year. That might seem like a lot, but $5 a month is pretty cheap to store multiple terabytes of storage.

Amazon Cloud Drive isn't exactly Linux-friendly, but there are command-line tools for uploading files, and many sync programs support it. In my case, the Synology NAS I use (Figure 4) has full support for Amazon Cloud Drive, and it automatically mirrors my entire NAS. The unlimited storage appears to be the real deal, and I haven't had any issues so far with reaching limits. That said, I don't have 48TB stored yet, so it's possible I'll be contacted as my storage allocation increases.

The Important Part

If you get nothing else from this article, please, make a backup of your important data. Even if it's just a live mirror copy, having your data in two places is so important in the increasingly digital world we live in, you just can't ignore the need.

I'm also very interested in hearing other backup ideas. If you have an affordable, free, innovative or unique way to back up files, please send me an e-mail at shawn@linuxjournal.com with "[BACKUP IDEA]" in the subject line.

In the meantime, I'll work on that solar-powered backup box for the backyard and be sure to write up a how-to so we can all sleep a little better at night!■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

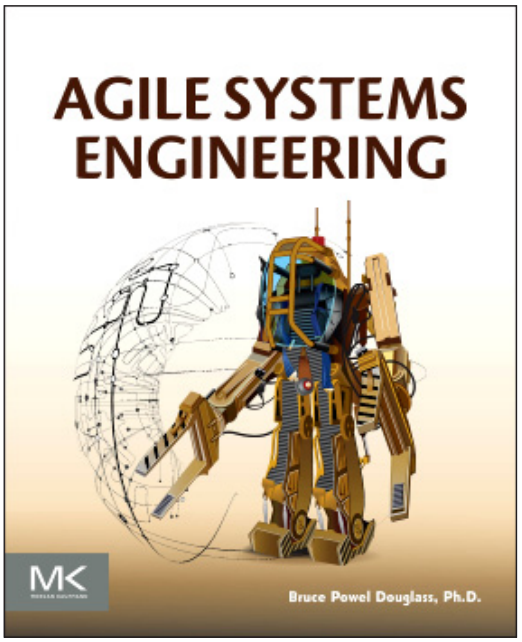
RETURN TO CONTENTS

PREVIOUS
Shawn Power's
The Open-Source
Classroom

NEXT
Feature: The Power
of Tiny initrd

The PostgreSQL Global Development Group's PostgreSQL

With Gartner predicting that 70% of new applications soon will be deployed on an open-source relational database, purveyors of proprietary DBMSes surely long for the golden lock-in age of yore. Said trend is in no small part due to the surging PostgreSQL, which recently stepped up to a new version 9.5. Highlights of this release, reported the PostgreSQL Global Development Group, include the addition of UPSERT capability, Row Level Security and multiple Big Data features, all of which the Group hopes will broaden the user base for the database. The Group added that version 9.5 marks a turning point for use of Postgres with data-driven applications of engagement and high-speed, high-volume mobile, Web and digital apps. Other specific features are performance boosters for today's more powerful "big iron" servers, analytics and productivity enhancements to speed complex query capabilities on extreme data volumes, and a foundation for horizontal scalability across multiple servers for importing entire tables from external databases.
http://postgresql.org



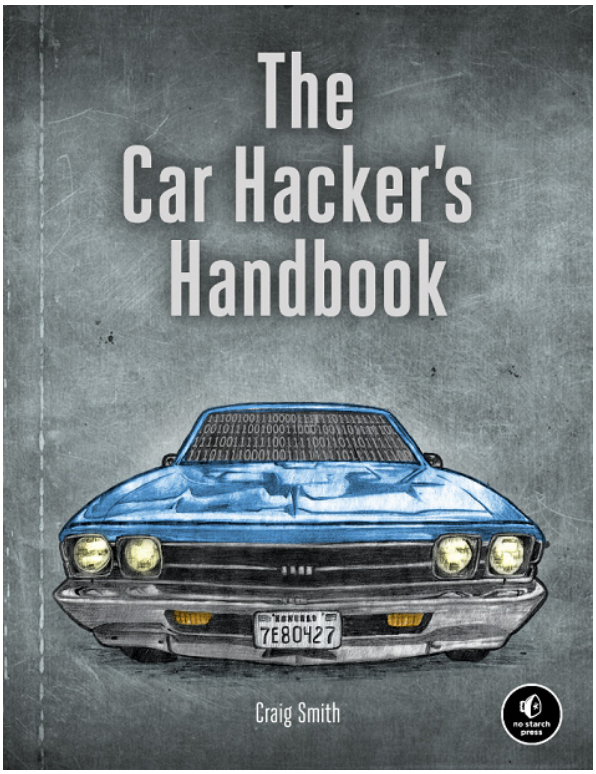
Bruce Douglass' *Agile Systems Engineering* (Morgan Kaufmann)

System engineers in aerospace, defense, automotive, transportation and rail, not to mention embedded software developers across disciplines, will be drawn to the ideas in Dr Bruce Douglass' new book *Agile Systems Engineering*. In the book, world-renowned author and speaker Douglass presents a vision of systems engineering in which precise specification of requirements, structure and behavior fuse with larger concerns, such as safety, security, reliability and performance in an agile engineering context. Douglass incorporates agile methods and model-based systems engineering (MBSE) to define the properties of entire systems while avoiding errors that can occur when using traditional textual specifications. The lifecycle of systems development is covered, including requirements, analysis, design and the handoff to specific engineering disciplines. The goal is to arm systems engineers with the conceptual and methodological tools they need to avoid specification defects and improve system quality while simultaneously reducing the effort and cost of systems engineering.

http://store.elsevier.com

SUSE Linux Enterprise

In order to bridge the traditional enterprise Linux business with the emerging workloads of the cloud era, the new SUSE Linux Enterprise Service Pack 1 (SP1) adopts a modular approach. This means that parts of the OS run a faster lifecycle than the base OS, yet receive full support. It also means accelerated innovation through support for all recent hardware from IBM and Intel, including Intel Xeon and IBM z13, with SUSE claiming exclusive Linux-distribution support for the latter. In addition, SP1 provides new modules to apply updates for Linux container features or security requirements without having to update and recertify the base operating system. Meanwhile, Package Hub enables the latest technologies developed by the Open Source community to be compiled by SUSE for use with SUSE Linux Enterprise Server. Rounding out the SP1 innovations are new capabilities for maintaining application uptime, an improvement in the efficiency of data-center development and operations and Docker support, among others.
http://suse.com



Craig Smith's *The Car Hacker's Handbook* (No Starch Press)

Not long ago, a group of benevolent hackers, from a couch ten miles away, no less, took full control of a friend's Jeep and ran it into a ditch. We should all take pause as automakers rush to transform cars into smartphones on wheels and a Wild West frontier emerges in automotive security. Security-minded people seeking to explore this frontier now have a cowboy-in-chief, Craig Smith, a security expert who has advised automakers and written a new book called *The Car Hacker's Handbook*. The book is a guide that shows how to identify network security risks, exploit software vulnerabilities and gain a deeper understanding of the software running in our vehicles. Along the way, Smith teachers readers how navigation systems can be hacked to take control of vehicles, how systems are interconnected, and even how to bypass dealership restrictions to diagnose and troubleshoot problems. Not only is this book a technical manual for hackers, but it's also a wake-up call for legislators and car manufacturers.
http://nostarch.com

The Linux Foundation's Automotive Grade Linux

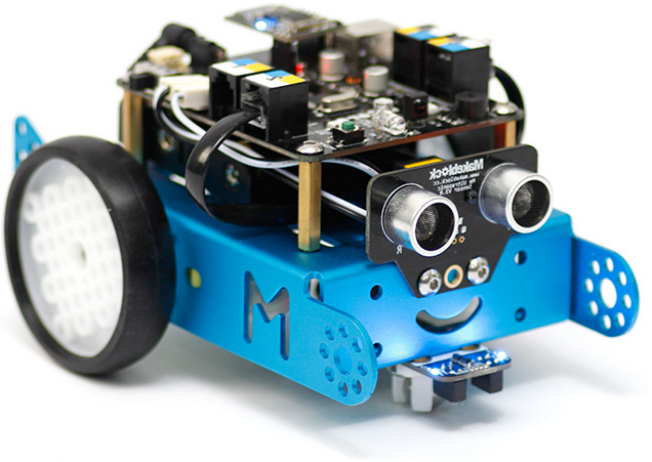
You couldn't ask for a better segue than this, from Smith's book about the pitfalls of automotive security to our community's solution to them—that is, The Linux Foundation's Automotive Grade Linux (AGL) new Unified Code Base (UCB) distribution. AGL is a Linux Foundation Workgroup dedicated to creating open-source software solutions for automotive applications. AGL's UCB distribution is a collaborative open-source project developing a common, Linux-based software stack for the connected car. Leveraging the best software components from AGL and other existing open-source projects, such as Tizen and GENIVI Alliance, UCB enables development of in-vehicle-infotainment systems while allowing different profiles to be created from the same code base to address all applications in the car, such as instrument cluster, heads-up display, telematics and connected car. UCB is based on the Yocto Project and offers a complete embedded-Linux development environment with tools, metadata and documentation. AGL's members make up a "who's who" of the automotive, IT and electronics industries, including Toyota, Ford, Intel, Sony, Linaro, Wind River and scores of others.
http://automotivelinux.org

Samsung's SmartThings

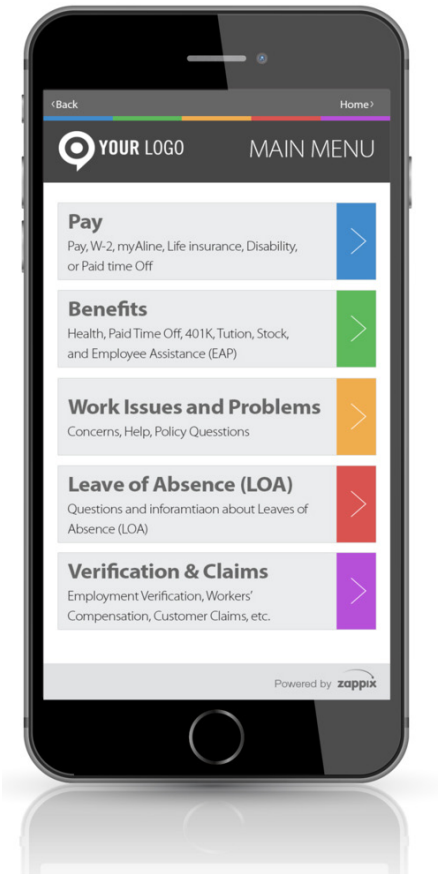
If you pick up a Samsung Smart TV this year, you'll be certain to find "Linux Inside" in many ways. Samsung continues to build on its Tizen-powered Smart TV UI, which this year it will enhance with integrated SmartThings IoT hub technology, enabling the TV as the control center for a smart home. Samsung's SUHD TVs for 2016 will enable users to connect with, control and monitor hundreds of other compatible devices including lights, locks, thermostats, cameras, speakers, appliances, sensors and the like. This world of devices is enabled via the SmartThings Extend USB adapter, which plugs directly in to the TV, enabling the monitoring and controlling of ZigBee and Z-Wave devices. The SmartThings SUHD application will allow users to trigger SmartThings Routines, receive notifications and observe what's happening in and around the home with compatible cameras and control connected devices. This integration of the SmartThings functionality eliminates the need to obtain Samsung's existing SmartThings Home Monitoring Kit.

http://smartthings.com

Makeblock's mBot



The new kid-friendly mBot from Makeblock is marketed as a Science, Technology and Mathematics (STEM) educational learning tool. The product is a full, all-in-one solution for kids—of all ages, of course—to enjoy the hands-on experience of programming, electronics and robotics. Working with a graphical programming interface called mBlock, which is inspired by Scratch 2.0, kids can learn programming, control the robot and realize multiple functions. Users can program the robot to light up, send it messages and move it around using open-source ecosystems like Raspberry Pi or Arduino. mBot can connect to a mobile device via Bluetooth so that users can control movements with an app. The 38 parts can be assembled in around ten minutes, and color-labeled RJ25 ports ensure that wiring is convenient and maximum time can be spent on programming and creativity. Extensibility is infinite thanks to compatibility with the broader Makeblock platform as well as most LEGO bricks. Finally, two manuals and courses, both on-line and developed and maintained by teachers, enable maximum utilization of mBot for STEM education.
http://makeblock.cc/mbot



Zappix Visual IVR

Zappix's development of its Visual IVR customer service platform is informed by research showing that 77% of consumers report that valuing their time is the most important element of good service. Zappix is a highly intuitive self-help Visual IVR software suite platform for Android, iPhone, tablets and mobile Web that integrates voice and non-voice visual content with customer service channels—for example, phone/voice, Web, mobile on-line forms and multimedia resources. An updated version of the platform now supports voice-enabled commands for integrating with the visual IVR prompts. The new voice capability, reports Zappix, enables a better and smoother Visual IVR experience. Users now can search for problems, solutions and even their channel of choice using speech and receive visual prompts on their smartphone. The new functionality is intended to empower companies to provide the best of both formats—that is, visual and voice commands—allowing organizations to process high call volumes effectively and reduce the frustration callers experience searching for an item in complex IVR systems.
http://www.zappix.com

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.

RETURN TO CONTENTS

The Power of Tiny initrd

A success story of how a PACS server ends up in a tiny initrd.

EDUARDO ARCUSA LES

PREVIOUS
New Products

NEXT
Feature:
Introduction to tmux

Those of you who have tried it will agree with me on the benefits of using an initrd/initramfs as a real root filesystem and of the possibility of not using a hard disk drive (HDD) in your servers. If you haven't tried it, I invite you to continue reading.

If you don't already know, initrd/initramfs is a scheme to load a temporary root filesystem into RAM. initrd and initramfs refer to two different methods of achieving this. Both are commonly used to make preparations before the real root filesystem can be mounted. Years ago, initramfs didn't exist, and everyone used initrd. Today, people continue using initrd.

But at my workplace, instead of using initrd as a temporary root filesystem, we use initrd as the real root filesystem. That's why in many of our servers we don't need an HDD—everything is in RAM. The only data stored in the HDD is that which may vary from boot to boot and needs to be preserved. That's the case of a database in which we put the PostgreSQL/MySQL binaries and libraries into the initrd and only the data files into the HDD.

Having a basic tiny initrd/initramfs, to which we can add anything needed for each system, with a tiny custom kernel, and booting them with PXE has saved us more than one catastrophe and makes our work easier.

The advantages of using initrd/initramfs are considerable:

- **RAM:** memory is cheaper, faster, less energy-consuming and less susceptible to failures than HDDs or even SSDs.

- **Modern hardware, more RAM:** new hardware supports a great amount of RAM memory installed.

- **Easier crash recovery:** if a server crashes or someone accidentally erases a file, you just need to reboot it, and it will wake up again without any problems. Also, if a server crashes because of a hardware problem, you just need to replace the broken hardware, add your new server in a DHCP config file and create a new MAC entry in the PXE server.

- **Easier backups:** the size of the initrd is minuscule, and it's easier to have a copy compared with a whole system in HDD.

- **64-bit:** 64-bit systems can handle a lot of RAM.

- **Easier scaling:** adding a server is easy, just connect it, enable PXE boot in the BIOS, add an entry in the DHCP server and create a MAC entry in the PXE server.

- **Security:** if an intruder modifies any system file on your server, it will be restored when it restarts. If the server is infected with a worm, virus or rootkit, just restart it and move on.

initrd as a Real Root Filesystem

One of the reasons administrators don't use initrd as a real root filesystem is because they know that changes will disappear when restarting the server if they are not incorporated into the initrd. The steps for doing this are annoying: uncompressing the initrd file, mounting the filesystem, making changes, unmounting, compressing and restarting the server to take the new changes.

To work around this issue, we consider an initrd as a good enough filesystem when there's almost no need for modifying it.

In order to achieve this, we put files known to be susceptible to changes over time outside the initrd. These files normally are configuration files of services, scripts and files inside the /etc directory, such as ethers, resolv.conf or hosts.allow. These files are stored in our TFTP/SSH server. When the server boots, it gets its kernel and initrd, and then copies all these files via SSH.

For example, think of a caching proxy server. If you have a minimal basic initrd, you need to put only the libraries and binaries into it. But, the configuration file should be copied remotely from a different server where it can be modified in an easier way than having to modify the whole initrd filesystem.

All of these initrds, kernels and configuration files for each server are stored in our TFTP server. This server has DHCP, SSH and PXE services installed. When a server boots, it gets its kernel and initrd files from the

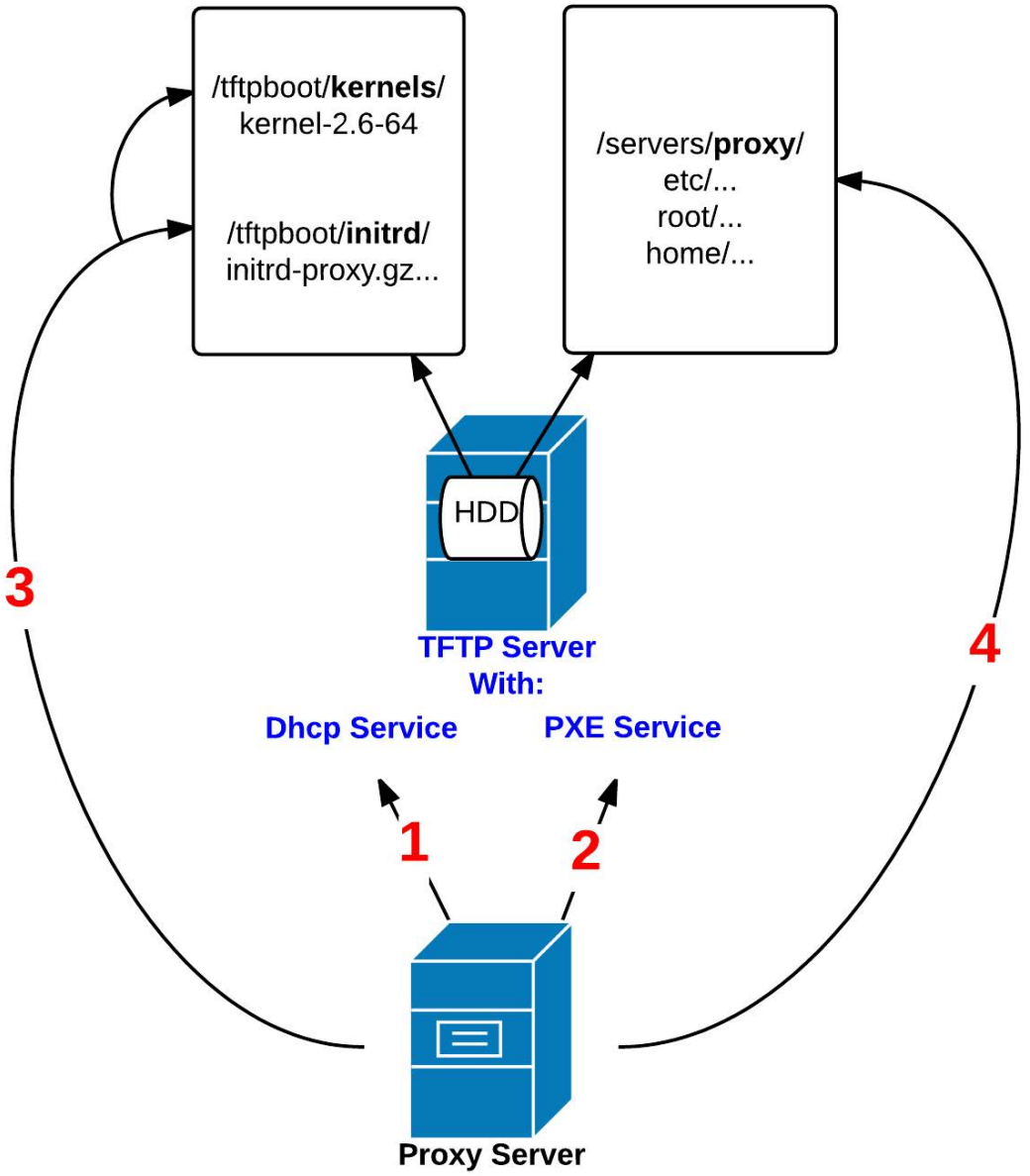


Figure 1. Here's an example of a proxy server with initrd as a real root filesystem. 1) Configure the proxy server BIOS to boot with PXE and configure the DHCP service to tell this server the IP of the TFTP server. 2) Configure the PXE service in the TFTP server to serve the kernel and initrd files to the proxy server. 3) The proxy server knows the IP of the TFTP server and asks it for the kernel and initrd files. 4) The proxy server boots and executes the /etc/rc script. This script mounts a directory of the TFTP server via SSHFS and copies the configuration files.

TFTP server and the configuration files via SSH.

There's no need to restart the server with every change we make; we just edit the file on the TFTP server, get it with sshfs or rsync into the proxy server and restart the proxy service.

Perhaps a graphic will make it easier to understand—see Figure 1.

Introduction to the PACS Server

At my workplace, we have many servers without HDDs that get their kernels and initrds from a TFTP server and then copy their configuration files. We also have some servers with HDDs to store critical data, such as database files, but they also boot with PXE. And when you see that it works magnificently, you will want to implement it on as many servers as possible.

And, this is the reason why this story began—the story of an uncommon server that finally ends in a tiny initrd.

A PACS (Picture Archiving and Communication System) is a system that stores all kind of digital medical images, so other computers can consult them later. The universal format for PACS image storage and transfer is DICOM (Digital Imaging and Communications in Medicine). DICOM enables the integration of scanners, servers, workstations, printers and network hardware from multiple manufacturers into a picture archiving and communication system. At my workplace, we use dcm4chee for such purposes because it's very powerful.

We administer a university dental clinic with 56 medical boxes, each one equipped with a workstation that allows users to view and upload radiological images. These sorts of images also are generated and uploaded from different kinds of scanners: intraoral, 2D/3D and CAT. For medical and academic purposes, teachers and students can work and do research from a lab classroom equipped with 15 workstations. Even though the images come from different manufacturers' equipment, all of them are stored in the same format (the DICOM standard) and in a unique repository, a PACS server.

The images are stored in a big disk array with 10TB connected with fiber to the PACS server, with all the image data (date, modality, description) and patient information (unique ID, name, sex, birth date) in the database. We have more than 50,000 patients stored right now.

We also have a DICOM Worklist service in the PACS server. This reduces operator search time, and it allows for a list of cited patients on a certain day to be consulted from any computer and uploads images only for those patients.

With the DICOM Worklist, it's easier to search and upload images for the correct patient, because if you search for a patient ID that

is not in the DICOM Worklist, it will not appear in the list, and you won't be able to upload a patient image that is not in the DICOM Worklist. The possibilities of uploading an image for the wrong patient is significantly reduced.

The clinical management software runs on a cluster of Web servers and allows us to list all the patients, their personal data and medical exams, and to view and create their diagnoses or treatments.

This is interesting: this cluster has eight servers. Seven of them run all in RAM without HDDs, except one, which has an HDD only for the database. Only the TFTP server boots from HDD, because it is the TFTP server of the cluster and has all the kernels, initrds and configuration files of all the other servers in the cluster.

Five years ago, we mounted two PACS servers in Linux. Because of the complexity and the little knowledge that we had, we didn't create initrds for them.

These two PACS servers are PACS-Master and PACS-HA (high availability). Both have a big disk array attached with 10TB to store the images. When someone uploads an image, PACS-Master saves it and sends all the records and images to the PACS-HA. If the PACS-Master has a problem and doesn't respond, the keepalived service gives the control to PACS-HA.

PACS Server to RAM

Through the years, we have had to change the HDD of PACS-Master twice because of failures. We have backups of the database and an image of the whole HDD, but the size of the image is very large, and with the new changes, we needed to create it again. That is not KISS.

We decided to upgrade the server to 64-bit, creating a minimal initrd, upgrading the dcm4chee version and the old database, importing the images and studying how dcm4chee and the DICOM Worklist work to know which files would be included in the initrd and which ones into the HDD.

Table 1 shows the specifications of both the old 32-bit server and the new 64-bit one of PACS-Initrd.

I'm not going to go into how to create an initrd or how to upgrade dcm4chee here; those topics are more technical and beyond the scope of this article.

Table 1. The Specifications of Both an Old 32-Bit Server and the New 64-Bit PACS-Initrd Server

PACS-Master/PACS-HA	PACS-Initrd
CentOS 5.2, 32 bits	CentOS 5.2, *64 bits*
4GB RAM (3.3 available)	*16GB RAM*
dcm4chee 2.14.2	dcm4chee *2.17.3*
DICOM Worklist	DICOM Worklist
Java 5 32 bits	Java *6 64 bits*
PCI Fiber	PCI Fiber
Big disk array 10TB	Big disk array 10TB

The real difficulty was learning how the dcm4chee and DICOM Worklist services operate and which files had to be located in the initrd and which ones in the HDD. Because initrd is a block device with a limited size, if you have files that grow over time and exceed that size, you'll get a kernel panic.

For example, think of a directory that you know a service uses and that contains files that increase in size over time. You only need to create in the initrd a symbolic link with `ln -s` to the HDD:

```
root@pacs1 /dcm4chee/server/default]$ ls -la | grep tmp
lrwxrwxrwx  1 root root   38 Mar 13 17:27 tmp -> /HD/dcm4chee/tmp/
```

It's important to remember that if you change the HDD, you also need to create the directories again, or the symbolic link will be broken.

When we got the job done after months of work and tests, we got a kernel of 2.2MB and an initrd of 204MB compressed, 420MB uncompressed. Normally, our initrds occupy less than 100MB uncompressed. You might think that this initrd is not very tiny, but keep in mind that only the size of the dcm4chee and the DICOM Worklist occupies 80% of the total size.

Into the initrd and HDD

We have incorporated all these applications and services into our tiny initrd filesystem: PostgreSQL, Postfix mail server, NTP, SNMP, syslog, OpenSSH, Keepalived, cron, dcm4chee, DICOM Worklist and Java 6. Consider that all of those services don't start on the HDD; they do it in RAM!

We have incorporated these data files into our HDDs: the database, the dcm4chee and the DICOM Worklist files that increase in size over time.

PACS-Initrd at Startup

The PACS-Initrd boots first from Ethernet, so we enable PXE boot in the BIOS. We add an entry for this server in DHCP and the IP of its TFTP server:

```
[root@tftp-server /]# cat /etc/dhcpd.conf | grep -i pacs1 -A 5
host pacs1 {
            hardware ethernet xx:xx:xx:xx:xx:xx;
            fixed-address 10.0.0.12;
            filename "pxelinux.0";
            next-server 10.0.0.1;
           }
```

We configure the PXE service in the TFTP server to serve the kernel and initrd to PACS-Initrd:

```
[root@tftp-server /]# cat /tftpboot/pxelinux.cfg/pacs
prompt 1
timeout 5
default L
label L
kernel kernels/kernel-2.6-64-pacs
append rw root=/dev/ram0 load_ramdisk=1 ramdisk_size=524288
 ➥initrd=initrd/fs-pacs.gz
 ➥ip=10.0.0.12:10.0.0.1:10.0.0.7:255.255.0.0:pacs1
ipappend 1
```

PACS-Initrd starts with that kernel and initrd files and executes the /etc/rc script. This script mounts a directory of the TFTP server via SSHFS and copies all the configuration files that are often modified into the PACS-Initrd, such as Keepalived, PostgreSQL, the /etc directory or cron, and unmounts the directory:

```
echo "Getting config files via SSHFS ..."
sshfs user@10.0.0.1:/servers/ /mnt

/bin/cp -aL /mnt/pacs/db /
/bin/cp -aL /mnt/pacs/etc/* /etc/
/bin/cp -aL /mnt/pacs/home/sysman/* /home/sysman/
/bin/cp -aL /mnt/pacs/root /
/bin/cp -aL /mnt/pacs/var/* /var/
/bin/umount /mnt

/etc/rc.local
```

PACS-Initrd executes /etc/rc.local and launches all the services.

Creating a High-Availability PACS Server

We waited a few months before changing the PACS-HA to an initrd filesystem. During that time, PACS-Initrd worked perfectly without problems. We had no complaints from our users—and as you know, no news is good news.

Once we had an initial initrd, creating PACS-Initrd-HA was done in 15 minutes (not taking into account the time needed to import and upgrade the old database). We added the new server in DHCP for it to attack the TFTP server and download the same kernel and the same initrd from PACS-Initrd.

Some services needed different configuration files for each PACS server, so the solution was to create new configuration files in the /servers/pacs/ directory of the TFTP server. For example:

```
[root@tftp-server /servers/pacs/etc]# ls -la | grep keepalived
-rw-rw-r--  1 root root  552 Mar  7 11:24 keepalived-pacs1.conf
-rw-rw-r--  1 root root  551 Mar  7 11:24 keepalived-pacs2.conf
```

In order to avoid having two different directories for both PACS servers, we configured /etc/rc.local to choose the appropriate configuration files for each PACS server:

```
[root@pacs1 /]$ cat /etc/rc.local
#!/bin/bash

echo -e "\n--- rc.local ------"

echo "Some extra configuration ..."
HOSTNAME=`/bin/hostname`

echo "Configurations for $HOSTNAME ..."
case $HOSTNAME in
    "pacs1" | "pacs2")
        mv /home/sysman/host-$HOSTNAME.conf /home/sysman/host.conf;
        rm /home/sysman/host-*.conf;
        mv /etc/postfix/main-$HOSTNAME.cf /etc/postfix/main.cf;
        rm /etc/postfix/main-*.cf;
        mv /etc/keepalived-$HOSTNAME.conf /etc/keepalived.conf;
        rm /etc/keepalived-*.conf;
        ;;
    *)
        echo "ERROR in HOSTNAME ($HOSTNAME) ...";
        exit -1;
        ;;
esac
```

Then we formatted an HDD partition in ext4, initialized the database, imported and upgraded the old database of PACS-HA and created the directories that come with a symbolic link from initrd.

Other Configurations

Security: We secured the common binaries with `chattr` with the -i (immutable) option. An intruder gaining access won't be allowed to change a common binary for a rootkit because it has the immutable

attribute. And, of course, our initrd doesn't own the chattr binary.

We configured TCP wrappers to deny all SSH connections except the ones needed by the servers to communicate with the PACS server.

We configured the ethers file with a static MAC to prevent a MITM (Man-In-the-Middle) attack.

We configured snmpd.conf to deny all queries except ones coming from our monitor server.

Centralized Logs: We configured syslog to send all logs to another server that has the syslog-ng service running.

Backup: At night, our backup server backs up the database and the configuration of the dcm4chee service.

Internal Monitoring: We use SMART (Smart Monitoring and Rebooting Tool) for such purposes, because it's not only a passive monitor, but it's also an active monitor that can auto-recover a service without the system administrator's intervention if it detects that it's down.

We added dcm4chee and the DICOM Worklist services to SMART.

```
[root@pacs1 ~]$ /home/sysman/smart

SERVICE         PID     PROCS   STATUS
-------         -----   -----   ------
CRON            3034    1       [OK]
DISK            ?       0       [OK]
KALIVED         6320    2       [OK]
KLOG            175     1       [OK]
NTP             ?       1       [OK]
POSTFIX         163     1       [OK]
POSTGRES        185     9       [OK]
SNMP            5741    1       [OK]
SSH             196     3       [OK]
SYSLOG          173     1       [OK]
DCM4CHEE        ?       1       [OK]
WORKLIST        ?       1       [OK]
```

Figure 2. Viewing All the Services We're Monitoring with SMART

External Monitoring

We monitor the server behavior and status with SNMP and Zabbix. We also monitor the services with Nagios.

Conclusion

The benefits of doing this are considerable. We saw in Zabbix that performance improved substantially. Backups are simple because the size of the initrd is minuscule, and it's easier to have a copy compared with a whole system image of the HDD. And, we need to back up only one initrd, because PACS-Initrd and PACS-Initrd-HA use the same one. If any file is corrupted or accidentally deleted, the solution is just to restart the server, and it'll wake up again in a healthy state without having to re-install the whole OS. If the HDD gets broken, it will be easy to recover, because it stores only the database and the configuration files and not the whole OS. If the hardware breaks down, we just have to replace it and add a new server in DHCP and a MAC entry in the TFTP server. The server's security is improved,

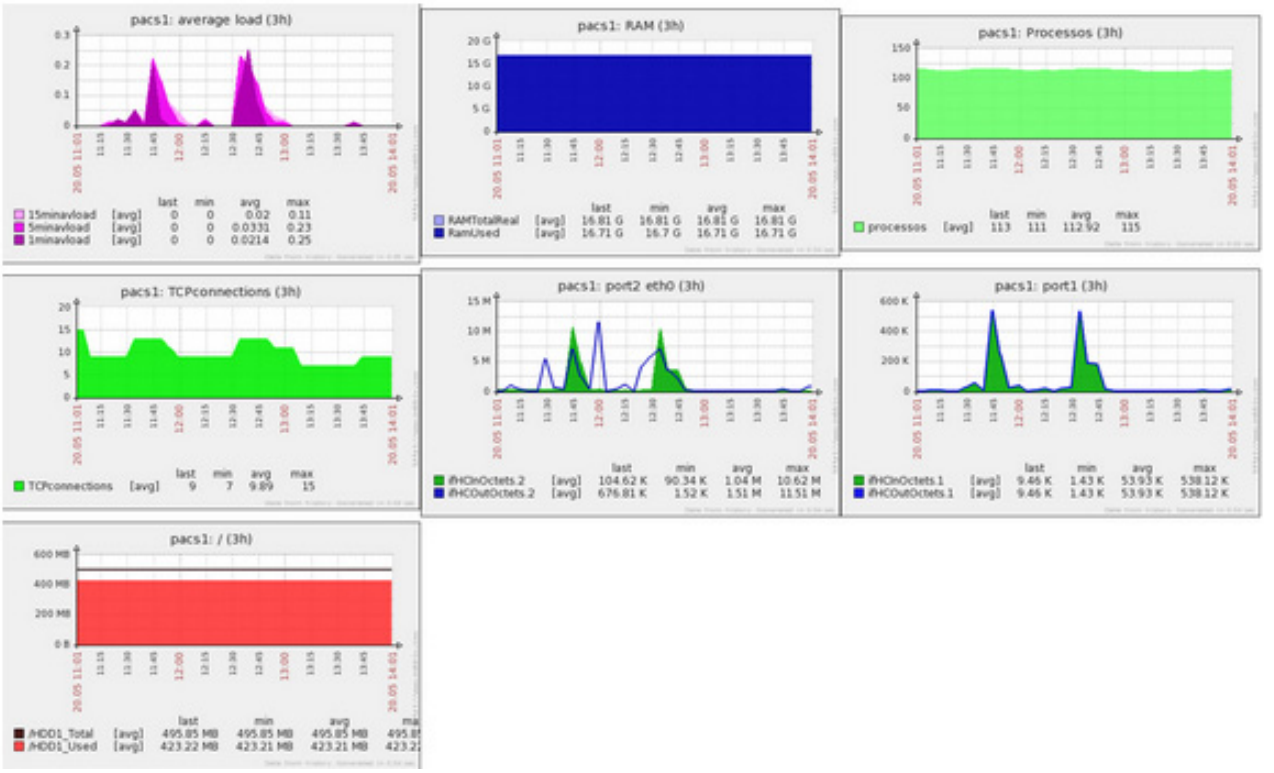


Figure 3. The Zabbix Control Panel Monitoring the PACS Server with SNMP

because if we restart the server, any changes made by viruses, worms or rootkits will disappear. Mounting a second PACS server was easy, because we already had the initial initrd. The most time-consuming part was creating the configuration files, adding a few lines in rc.local and a new entry in the DHCP service.

I hope this article helps you consider the benefits of having a server starting with PXE, and I encourage you to do it on many of your servers.

Acknowledgements

PACS-Initrd was created, developed, tested and enjoyed in the IT Department of the UIC Barcelona. Albert Martorell, Isaac Vázquez, Andreu Garcia, Ildefonso Aranda, Josep Pablo, Vicente Sangrador and Jordi Xavier Prat have collaborated on this project and encouraged me to write this article. Thanks to this awesome team! Without you, this would not have been possible.

Finally, thanks to my girlfriend for always having the patience to listen to me when I talk emotionally to her about the PACS server, initrds and kernels without understanding anything at all.■

Eduardo Arcusa les is 28 years old and has been working as a network administrator in the IT Department of the UIC Barcelona since 2009. He likes doing things with Raspberry Pi and solving *Rubik's Cube*—his office is full of them.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

Introduction to tmux

Learning tmux can be daunting with all of its options and hotkeys. Here's what you need to get up to speed quickly.

CHARLES THOMAS

PREVIOUS
Feature: The Power of Tiny initrd

NEXT
Doc Searls' EOF

tmux is one of those programs that seemed to have so many options and new commands, I thought it wasn't worth learning. Not because it wouldn't be useful, but because I assumed the learning curve would be too steep. Now that I've finally dug in to it and made it a part of my work flow though, I can't imagine going back to life without it, and I regret every time I thought about learning it and didn't. tmux does have a lot of bells and whistles, but knowing them isn't vital to its usefulness. In other words, you can do a lot with just a little tmux knowledge and a good cheatsheet.

tmux (terminal multiplexer) functions like screen: you create a session and do work inside it. At any time, you can detach the session, only to reattach later and find everything set up exactly as you left it when you detached. The feature of tmux that sets it apart from screen is the ability to split your terminal window into multiple window panes and rearrange them on the fly. Screen allows you to create multiple windows per session, but with tmux, you then can split each window vertically or horizontally (or both) and resize the panes on a whim. It's also trivial to swap panes around the window and even rotate through a pre-selected set of pane groupings—like in a tiling window manager.

A Little Background

When I first started working in operations, I found I was in and out of multiple SSH sessions to dozens of machines all day long. Pretty soon, I started looking for a terminal emulator that could handle tabs and the functionality described above, and I found Terminator for Linux.

When I moved to a QA role in a startup, I was handed a MacBook, and the first thing I did was find the OS X equivalent: iTerm2. That served me fairly well until a fateful day working out of a coffee shop while getting my car repaired. My current job involves launching anywhere between 5 and 50 cloud instances at a time to test against. The coffee shop's Wi-Fi cut out so much that I spent all day trying to launch a single set of instances and never got any actual work done.

Once my car was finished and I made it home, I dug an old Raspberry Pi out of a closet and set it up to launch instances from it. I would `ssh` in and open screen, then launch instances to my heart's content. If my laptop's Wi-Fi cut out, it didn't matter. My SSH connection to the Raspberry Pi would die,

but the instances were launching in screen, so the session wasn't lost; I just couldn't see it. When the Wi-Fi came back, I'd `ssh` back in and reattach to my screen session. This gave me the added bonus of being able to issue the tear down command and sleep my laptop right away instead of waiting for the instances to be destroyed. That meant I didn't have to budget an extra 20 minutes into the end of my day for waiting on instance clean up.

This scheme worked pretty well, but eventually I realized I also could move my entire development environment to the Raspberry Pi, and it would mean less context switching and shuffling files and config around. Moving my whole setup to the Raspberry Pi was another big step in the right direction, but I pretty quickly realized I was missing the multi-paned windows of iTerm2, and screen's tabs just weren't enough. So I finally gave in and tried tmux.

Getting Started with tmux

If tmux isn't already installed by default in your distribution, it's almost certainly in your package manager (apt, yum and so on). After installation, tmux will launch your first session. You should see what looks like your normal prompt, but at the bottom of your terminal is the tmux status line. At the far right of the status line is your hostname followed by the time and date. The left side should show something like `[0] 0: bash*`. The 0 in square brackets `[0]` is the session name. Knowing the session name is important if you want to use more than one session and attach to the correct one, but that won't be covered in this introduction. The rest of the text on the left is the window list. Since you've just opened tmux for the first time, you have only the default window (number 0), and it will be displaying the current running program (bash, or your shell of choice, since it's idle). The asterisk denotes the active window. It's worth noting that `"hostname"` in quotation marks on the right side of the status bar is actually whatever the title of your terminal is; hostname is just the default. So if you run something that normally would change the name of your terminal window, that change will now be reflected on the right in the tmux status line.

Pressing Ctrl-b is the magic keystroke to access tmux command mode. This is similar to Ctrl-a in screen, with one big advantage: outside of screen, typing Ctrl-a will move your cursor to the beginning of your

pending command string. Inside screen, you have to press Ctrl-a and then a again. Because tmux uses Ctrl-b for commands, Ctrl-a still works like it does outside of tmux.

To detach from your tmux session, press Ctrl-b, then press d for detach. Attaching is the one area where screen has an edge on tmux. Screen has flags that will attach an existing session if one exists and create one if it doesn't (`screen -dRR`), which means it can be easily aliased. Without extra configuration, `tmux` always will create a new session, and `tmux attach` will attach only to an existing session. There is no "attach or create". One way around this is `alias tm='tmux attach || tmux'` Running `tmux attach` will try to attach to your detached session and return 1 if it failed, because there are no sessions, so `tmux attach || tmux` will try to re-attach and create a new one if that failed.

Moving Around

After you've successfully created a tmux session, you'll want to start opening windows and splitting them into multiple panes—the good stuff. Create a new window with Ctrl-b, c. You'll see in your window list below that you now have something like `[0] 0: bash 1: bash*`. The `1` is now the active window (so the asterisk moved). You can navigate through the list of windows backward with Ctrl-b, p (for previous), forward Ctrl-b, n (for next), and you can jump directly to a numbered window with Ctrl-b followed by the number. So if you're still on window 1, pressing Ctrl-b, 0 will move you back to window 0.

If you have more than ten windows open, that method won't work. Trying to do Ctrl-b, 10 will just put you on window 1 before you can type 0. Once you have a double-digit number of windows open, you'll want to use Ctrl-b, ', which prompts you for an index. Type the number you want to go to, and when you press return, you'll go to that number, so you can type 34 without being sent to window 3.

Moving forward and backward through your window list wraps, so if you have three windows open (0, 1, 2) and you're on two, pressing Ctrl-b, n will take you to window 0. And, then Ctrl-b, p will take you from window 0 back to window 2.

If you find yourself going back and forth between two non-consecutive windows, rather than remembering their numbers or shuffling through

You can keep splitting your screen beyond that too. Each time, the current pane will be split in half leaving the rest of the panes as they were.

with n or p, use Ctrl-b, l to swap back and forth. This will take you to your previous window, kind of like how `cd -` changes back and forth between two directories on the command line.

Splitting Windows into Multiple Panes

Splitting windows into multiple panes is a similar process. To split a window pane in half horizontally, press Ctrl-b, ". To split vertically, press Ctrl-b, %. You can keep splitting your screen beyond that too. Each time, the current pane will be split in half leaving the rest of the panes as they were.

You can navigate between panes with the arrow keys (after pressing Ctrl-b, of course). Or you can cycle through all the panes with Ctrl-b, o.

Knowing which pane is "next in line" can be tricky with lots of panes in lots of different sizes. To find out how the panes are numbered (and which pane will be active next with Ctrl-b, o), use Ctrl-b, q to display an index in each of the active window's panes.

Finally, to swap back and forth between panes, Ctrl-b, ; works with panes like Ctrl-b, l does with windows as described above.

One minor disadvantage of tmux (and screen) is the inability to scroll up through your terminal history with a scroll bar. But, you still can scroll back, just press Ctrl-b, [first. Then you can navigate up the screen with the arrow keys. This is actually copy mode. tmux has its own copy/paste buffers, which are active across the session, so you can copy from one pane into another (or into another window).

Copy Mode

Unfortunately, this is where tmux gets a little funky. tmux has two

different sets of keys to perform the same operations: one that emulates emacs, and the other emulates vi. Describing all of the different operations and their emacs and vi bindings is beyond the scope of this introduction; see the manual page for details. What is important is that the way you highlight text is different depending on the mode. The default mode is emacs. You can change that in a number of ways, but worth mentioning here is by your VISUAL and/or EDITOR environment variables, because it can change the bindings without you expressly making the change yourself. If either of those contain vi (for example, vi or vim), tmux will set your mode to vi.

Once you're in copy mode (by pressing Ctrl-b, [), press the spacebar, and then use the arrow keys to move your cursor around. If the text is highlighted, you are in vi mode. If not, navigate back to what you want to copy and use Ctrl-spacebar to start highlighting instead. After using the arrow keys (or emacs or vi shortcuts) to highlight the text you want to copy, use spacebar or Ctrl-spacebar again to copy the highlighted text into the buffer and exit copy mode. Now use the keys described above to navigate to the window and pane you want to paste into, and use Ctrl-b,] to paste.

Resizing and Rearranging

Once you have your window panes all split up, you may find you want to resize or rearrange them. This process is also slightly more complicated, but it's not as confusing as copy mode's multiple key combination stuff. Resizing can be done by one- or five-line increments. To change a panel's size by one line, use Ctrl-b then Ctrl-arrow key, which will expand or contract the pane (depending on the orientation) in the direction of the arrow key. This also will have the opposite effect on the adjacent pane. tmux has no control over the size of the whole window; that is up to your window manager. If you expand one pane, the adjacent pane must shrink. To resize by five-line increments, the process is identical, except instead of Ctrl-arrow key, press Ctrl-b, Alt-arrow key.

Rearranging panes is also possible. If you have a layout you like but want some panes swapped around, use Ctrl-b, { or Ctrl-b, } to swap the current pane with the previous or next pane. (Press Ctrl-b, q to see the pane indices, so you know which pane will be swapped with the current pane.)

To change your pane layout completely, reshaping and redrawing

If you find that you have one pane you just can't fit into your current window the way you like, there's also an option to break it out.

all of the panes, use Ctrl-b, spacebar to rotate through five different layout presets:

- Even-horizontal: all panes in even columns.
- Even-vertical: all panes in even rows.
- Main-horizontal: one big row on top with the rest of the panes in columns underneath.
- Main-vertical: one big column on the left with the remaining panes in rows on the right.
- Tiled: all panes in even rectangles.

According to the manual page, you also can use Ctrl-b, Alt-[1-5] in order to rearrange the panes directly in the style described above, respectively, without cycling through the others. For whatever reason, I was unable to get that to work, but your mileage may vary.

If you find that you have one pane you just can't fit into your current window the way you like, there's also an option to break it out. Pressing Ctrl-b, ! will remove the current pane from the current window and move it to a new window of its own.

Rearranging whole windows is much simpler. Pressing Ctrl-b, . will prompt for a new index for the current window. You must provide an index that is not already in use. This will rearrange the windows so that the window you re-indexed now will be in the correct spot when cycling

through windows with Ctrl-b, n or Ctrl-b, p.

On the face of it, this would suggest that you can re-index windows only at the end of the current list, but that isn't strictly true. When you close a window, the existing windows are not reordered. So if you have windows 0, 1, 2, 3, and you were to close window 1, you then would have 0, 2, 3. Windows 2 and 3 do not shift down to fill in the gap. So, pressing Ctrl-b, . would re-index any of those windows as window 1.

It's also worth noting that when you create a new window, the first missing index is used rather than adding on to the end. Using the same example, if you have windows 0, 2, 3 and you use Ctrl-b, c to create a new window, you will end up with 0, 1, 2, 3 again, not 0, 2, 3, 4.

Conclusion

This should be more than enough to get you started with tmux. There's plenty more functionality to explore in the man page, such as rebinding keys, customizing your status line, even broadcasting your keystrokes to all the visible window panes instead of just the active one. You also can make tmux config files to automate setting up your tmux environment. You can define a set of windows and panes to be opened and arranged automatically when you start a new session. You even can tell tmux to run a command inside a pane automatically when the session starts!

But, don't be intimidated by all those bells and whistles like I was. You can get a lot of benefit out of tmux, even if all you do is open a couple windows and split them into two panes on occasion. The man page has a reasonably good list of hotkeys, but they aren't in any order that I could make sense of, so I made a chart (see the tmux Cheatsheet sidebar), which should help get you started with the basics described in this article.■

Charles Thomas is a Selenium tester, Python developer and open-source contributor. He can be found all over the Web, but cha.rlesthom.as is a good place to start.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

Advanced Topic—SSH and tmux

There is one advanced topic worth mentioning, since it's beyond the scope of the manual page and took some Internet sleuthing (Duck Duck Go and Stack Overflow) for me to figure out: ssh-ing directly to tmux. My primary use of tmux is on my remote development machine. I am sufficiently lazy that opening a new terminal window, ssh-ing to my dev box, then trying to figure out if I already have a tmux session going and either attaching to it or creating a new one is all just too much work. The alias I described above would take care of the "create new session" versus "attach to existing session" problem. But I can do one better: automatically create or re-attach when SSH connects.

There actually are several ways to approach this, each with pros and cons. The most obvious to me was to set tmux as my shell, using chsh or editing /etc/passwd. Any relatively up-to-date distribution and tmux version should Just Work. tmux does have a flag for running correctly as a shell in older versions (see the man page). The problem with this approach is that without a custom tmux config file, you are connected to a new session. Pressing Ctrl-b, s will show you a menu of sessions and allow you to use arrow keys and enter to select the old session you detached from, but then you have to deal with cleaning up your new session(s), which involves entering tmux commands after using Ctrl-b, : to access the tmux shell (again, see the man page).

You also could invoke tmux inside your .bashrc (or the equivalent in your shell of choice), but you have to add

checks to make sure you aren't already in tmux, or you could end up in a loop. Your shell launches tmux, which launches a shell, which runs your .bashrc again launching another tmux, and so on. This option seemed sufficiently dangerous to me that I didn't even try it, although I was able to find information about other people doing it.

In the end, the option I chose was to edit my SSH authorized_keys file. This is probably considered a bit of a hack. The authorized_keys file has an option for restricting access to certain commands by adding `command=...` before the start of the key. I think this is how GitHub allows SSH access for git but not shell access, for example.

This has a downside as well. Because it's a restriction, if you have more than one SSH key in the authorized_keys, and you're coming from a machine with more than one of those keys, SSH is going to use the one that is the least restricted.

In my case, I generated a new key to connect from my work laptop to my Raspberry Pi dev box. I left the original key in place on the Raspberry Pi and added the new one with `command="tmux attach || tmux"` prepended. When I tried to `ssh` in, even when I specified the identity file that was restricted, SSH would use the original, unrestricted key (without `command=`, etc., in it), and I would get a normal shell, not a tmux session. I resolved this by removing all other authorized keys from the machine. I suppose I also could have added a different user with only that new and restricted authorized key or removed the other private key from my work laptop, but removing the other key seemed like the easiest solution at the time.

tmux Cheatsheet

Starting, Attaching and Detaching:

- `tmux` — start a new tmux session.
- `tmux attach` — attach to an existing session.
- `tmux attach -t mysession` — attach to a session named mysession.
- `tmux attach || tmux` — attach to a session if one exists; if not create a new one.
- Ctrl-b, d — detach from the active session.
- Ctrl-b, : — open the tmux command shell.
- Ctrl-b, ? — display tmux commands and key bindings.
- Ctrl-b, s — prompt to change sessions, selecting from a list.
- Ctrl-b, $ — rename the current session.

Adding and Navigating Windows:

- Ctrl-b, c — create a new window.
- Ctrl-b, n — activate the next window in the window list.
- Ctrl-b, p — activate the previous window in the window list.
- Ctrl-b, [0-9] — activate window number 0–9.
- Ctrl-b, ’ — prompt for a window number (for windows higher than 9).
- Ctrl-b, l — activate previously active window.

Adding and Navigating Panes:

- Ctrl-b, " — split current pane horizontally.
- Ctrl-b, % — split current pane vertically.
- Ctrl-b, arrows — move active pane to the adjacent pane in the direction of the arrow.
- Ctrl-b, o — activate the next pane in order.
- Ctrl-b, q — display each pane's index.
- Ctrl-b, ; — activate previously active pane.
- Ctrl-b, . — re-index the current window.

Resizing and Rearranging Panes:

- Ctrl-b, Ctrl-arrow — resize current pane one line by shrinking/expanding in the direction of the arrow.
- Ctrl-b, Alt-arrow — resize current pane five lines by shrinking/expanding in the direction of the arrow.
- Ctrl-b, { — swap the current pane with the previous pane.
- Ctrl-b, } — swap the current pane with the next pane.
- Ctrl-b, spacebar — rearrange all panes, cycling through even-horizontal, even-vertical, main-horizontal, main-vertical and tiled.
- Ctrl-b, Alt-[1-5] — rearrange all panes, using the numbered layout from the list above.
- Ctrl-b, ! — break the current pane out into a new window.

RETURN TO CONTENTS

The Forrester Wave™: Digital Experience Platforms, Q4 2015

The demand to be at every touchpoint in the customer lifecycle is no longer an option—it's a requirement. To manage and deliver experiences consistently across all touchpoints, organizations are looking to digital experience platforms as the foundation of their digital presence.

Get Forrester's evaluation of the best vendors, including:

- The ten providers that matter most.
- How each vendor stacks up to Forrester's criteria.
- Six needs a digital experience platform architecture must meet.

> **http://geekguide.linuxjournal.com/content/forrester-wave-digital-experience-platforms-q4-2015**

ACQUIA™ The Ultimate Guide to Drupal 8 by Acquia

With 200+ new features and improvements, Drupal 8 is the most advanced version of Drupal yet. Drupal 8 simplifies the development process, enabling you to do more, in less time, with proven technologies that make it easier to be a first time Drupal user. Read this eBook, written by Angie Byron (you may know her as "webchick"), to get up to speed on the new changes in Drupal 8. Drupal 8's improvements include:

- API-driven content approach.
- Rest-first native web services.
- Seamless integration with existing technologies.
- Multilingual features and capabilities.
- Responsive by nature and mobile-first.

> **http://geekguide.linuxjournal.com/content/ultimate-guide-drupal-8**

ACQUIA™ How to Choose a Great CMS by Acquia

Web Content Management Systems serve as the foundation of your digital experience strategy. Yet many organizations struggle with legacy proprietary products that can't keep pace with the new realities of digital marketing. To determine if you are in need of a new CMS, use our guide, which includes:

- An evaluation to see if your current CMS supports your digital business strategy.
- The top considerations when selecting a new CMS.
- A requirements checklist for your next CMS.
- Ten questions to ask CMS vendors.

> **http://geekguide.linuxjournal.com/content/how-choose-great-cms**

XebiaLabs Webinar: Diving into Docker: What it means for your Enterprise DevOps Strategy

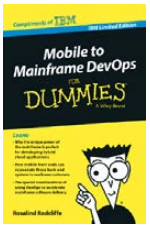
Docker has taken the software world by storm, but what does it actually mean for enterprise IT teams? Containers along with microservices are components definitely worth investigating for any modern software delivery pipeline when considering speed, portability and scalability.

Understanding whether they are right for you, and how you could introduce them into your enterprise tool chain and delivery pipeline can be challenging.

Webinar: Thursday, March 3rd, 8am PT | 11am ET | 5pm CET

Plus, as a special bonus, attendees will receive an advance copy of our newest white paper, Addressing the Enterprise Challenges Presented by Container Technologies!

> **https://xebialabs.com/community/webinars/diving-into-docker-what-it-means-for-your-enterprise-devops-strategy/**



Mobile to Mainframe DevOps for Dummies

In today's era of digital disruption empowered by cloud, mobile, and analytics, it's imperative for enterprise organizations to drive faster innovation while ensuring the stability of core business systems. While innovative systems of engagement demand speed, agility and experimentation, existing systems of record require similar attributes with additional and uncompromising requirements for governance and predictability. In this new book by Rosalind Radcliffe, IBM Distinguished Engineer, you will learn about:

- Responding to the challenges of variable speed IT.
- Why the mainframe is a unique and ideal platform for developing hybrid cloud applications.
- How mobile front ends can rejuvenate back-end systems to reach new customers.
- And, special considerations for using a DevOps approach to accelerate mainframe software delivery.

> **http://devops.linuxjournal.com/devops/mobile-mainframe-devops-dummies**

BRAND-NEW EDITION!

DevOps For Dummies – New Edition with SAFe®

In this NEW 2nd edition, learn why DevOps is essential for any business aspiring to be lean, agile, and capable of responding rapidly to changing customers and marketplace.

Download the E-book to learn about:

- The business need and value of DevOps.
- DevOps capabilities and adoption paths.
- How cloud accelerates DevOps.
- The Ten DevOps myths.
- And more.

> **http://devops.linuxjournal.com/devops/devops-dummies-new-edition-safe**

To Appreciate a Life

We might be forgotten, but some of our works will live on for decades, centuries or more—Ian Murdock's for example.

DOC SEARLS

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.

PREVIOUS
Feature: Introduction to tmux

Over dinner a few years ago, Kevin Kelly told me neither of us would be remembered a thousand years from now—nor would our work, even though we both (especially he) enjoyed a measure of celebrity, our bylines on books and magazine mastheads. Death, rot and other forms of change would erase nearly everybody while altering nearly everything. Absent of persistent interest, memories of memories of memories would include nothing of today's living present. Even the fully famous, celebrities and heads of state, would be forgotten.

Since recall is rewrite, remembering replaces the past with a replica. History drifts from facts. Same with the stories we tell, even about ourselves. "The older I get, the better I was", says Corey Booker, the US Senator from New Jersey, reflecting on his varsity football days

A decade into that century, with Linux more meaningful than ever to our whole networked civilization, I find myself wondering how long the world it maintains will last, and if the world would be lucky to still have Linux, doing what it has always done, and much more—or if the world has come to depend on other ways of computing.

at the University of Michigan.

Short-term memory lasts only a few seconds, so we tend not to know how we'll end the sentences we start, or to remember how we started the ones we finish. Details, including all words said and heard, fall away. All we retain are the generalities we call meaning, which may not even be there. Garrison Keillor says "English is the preacher's language, because it allows us to talk until we think of what to say." But that's true for every language. All of us make stuff up as we go along, climbing to the future while never leaving the present.

The most useful meanings outlive the words, people and media that carry them. On a *Linux Journal* Geek Cruise in 2005, I asked Andrew Morton, the Linux kernel maintainer, if he thought Linux would be around a hundred years in the future. He said yes, and that most work on the kernel in 2105 would still be "stamping out bugs".

A decade into that century, with Linux more meaningful than ever to our whole networked civilization, I find myself wondering how long the world it maintains will last, and if the world would be lucky to still have Linux, doing what it has always done, and much more—or if the world has come to depend on other ways of computing. No way to know, and few if any of us reading this will be around to find out.

But I'm guessing that one of us already gone might, by some small chance, still be remembered. That person is Ian Murdock, who left us

in January 2016. Ian was the father of Debian and the grandfather of Ubuntu and other Debian derivatives. He framed up package management as we've known it ever since. He wrote the Debian Manifesto, which predated the Open Source Definition by half a decade and helped model it. He designed Debian to attract a community of developers who invited and listened to users, and subordinated commercial ambition to the need for good code that served everybody. On March 1, 1994, he wrote:

> The time has come to concentrate on the future of Linux rather than on the destructive goal of enriching oneself at the expense of the entire Linux community and its future. The development and distribution of Debian may not be the answer to the problems that I have outlined in the Manifesto, but I hope that it will at least attract enough attention to these problems to allow them to be solved.

That was in issue #1 of *Linux Journal*. Ian also would contribute to issues #3 and #6 and many others after that. He was one of us. Yet his influence was not limited to Linux. Looking back through my own correspondence with Ian, a big subject was something else that's still with us: "open-source politics".

Back in 2001, when Ian was at Progeny, one of the company's advisors was Joe Trippi, who went on to head the Howard Dean campaign for President. The candidate imploded in Iowa, but the team and its methods have modeled and led network-based grassroots campaigning through the years since—the most notable successes being the election and re-election of Barack Obama. Back when Dean was riding high in 2003, Ian blogged this excerpt from a Larry Lessig interview (now off the Web):

> Trippi: I used to work for a little while for Progeny Linux Systems. I always wondered how could you take that same collaboration that occurs in Linux and open source and apply it here. What would happen if there were a way to do that and engage everybody in a presidential campaign?
>
> Lessig: So is this an open-source presidential campaign?
>
> Trippi: Yes. That moment when that was all going on made me think,

"That's sort of what we're building here." I guess it's about as open as you can do it in modern-day politics.

And then this from Slashdot (http://slashdot.org/comments.pl?sid=70875&cid=6428904):

> I have spent the better part of the last three to four years working on a lot of issues discussed here and with a lot of the technology. I advised Progeny Linux Systems—the Debian flavor, it gave me a lot of insight as to what open-source politics would be like and how the same principles could be applied. Really a lot of the same forces are at work if you think about it. Entrenched and flawed system, closed to everyone except the few that aim to keep control etc vs open dialogue, open collaboration, and a better solution emerging from the common actions of many. The country was not founded on the principle of self interest—it was founded on the principle of the common good. And it's fascinating to me how on every front it's the commons we need to build again.

Every death closes parentheses of time. Ian's were (1973–2015). That span is terribly short, and confused in the end by conflicting and bizarre accounts, including a police report and a series of tweets by Ian that were yanked by Twitter but saved by the Internet Archive. None of it reveals anything clear about Ian's final hours, other than trouble. No doubt more details will emerge over time, and perhaps something close to a true

account will be made public eventually. But the suppression of additional details, for whatever reason, hasn't been a bad thing. Better to celebrate a man who left the world a better place than just to talk about how he left it.

A thousand years from now Ian too likely will be among the billions forgotten—if there are still humans around to do the remembering. (I'm sure there will be, but that's still just a bet.) Meanwhile, we're here now, standing on his accomplishments. Appreciation is required.■

RESOURCES

Kevin Kelly: http://kk.org

Cory Booker: https://en.wikipedia.org/wiki/Cory_Booker

"Cruising the Kernel with Andrew, Ted and the Gang, Part I" by Doc Searls: http://www.linuxjournal.com/article/8607

Andrew Morton: https://en.wikipedia.org/wiki/Andrew_Morton_%28computer_programmer%29

The Debian Manifesto: https://www.debian.org/doc/manuals/project-history/ap-manifesto.en.html

"The Debian Distribution" by Ian Murdock in issue 1 of *Linux Journal* (March 1994): http://www.linuxjournal.com/article/2748?page=0,1

"The Open Development of Debian" by Ian Murdock in issue 3 of *Linux Journal* (June 1994): http://www.linuxjournal.com/article/2782

"Overview of the Debian GNU/Linux System" by Ian Murdock in issue 6 of *Linux Journal* (October 1994): http://www.linuxjournal.com/article/2841

Joe Trippi: https://en.wikipedia.org/wiki/Joe_Trippi

"Joe Trippi and Howard Dean" by Ian Murdock: http://ianmurdock.com/open-source/joe-trippi-and-howard-dean

Lawrence Lessig: http://www.lessig.org/about

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS