ADMIN

Network & Security

SPONSORED BY Linux Professional Institute

10 Terrific Tools

FOR THE BUSY ADMIN 2018 EDITION

Another Inspired Collection
of helpful tools for the network

Bonus articles
- bd, autojump, and Fasd
- Command-line backup apps

Find a free tool to help you
- Analyze the Linux startup process
- Sync your cloud drives
- Block annoying web ads
- Stress test your network

WWW.ADMIN-MAGAZINE.COM

US$ 7.95

ADMIN Network & Security

10 Terrific Tools

FOR THE BUSY ADMIN 2018 EDITION

ADMIN Special

Editor in Chief – Joe Casad

Copy Editor – Amy Pettle

Layout / Graphic Design – Dena Friesen, Lori White

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 3090 5128

Publisher – Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-866-247-2802
(toll-free from the US and Canada)

www.admin-magazine.com

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

ADMIN is published by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published in Europe by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany

Dear Readers:

Linux is built around the classic Unix principle of simple tools for a single purpose. Thousands of useful system administration utilities inhabit the Linux environment, and we highlight another 10 in this year's collection. Consider these candidates for your own sysadmin toolbox!

Table of Contents

As a special bonus, we're also including two more articles about useful tools for the busy admin:

01 Startup Scenes

In sys admin columnist Charly's case, no fumbled system startup goes undetected. This was already the case with SysVinit and has not changed with systemd. In terms of his analysis tools, no stopwatch goes unturned.

By Charly Kühnast

If a distribution shows its splash screen on startup – and almost all of them do – then I turn it off as of the second boot. I want to see the kernel messages rushing by when booting. Because more often than you might think, there's a problem with booting. When I see something questionable scurry by, I search with a tool like Bootchart [1] to find and fix a hanging process or similar issues quickly.

On systems with systemd, this is also possible in principle, but the tool is different: `systemd-analyze`. When called without parameters, `systemd-analyze` calculates the elapsed time until system startup is completed, broken down into kernel and userspace processes:

```
Startup finished in 3.507s (kernel) + ➥
  16.334s (userspace) = 19.842s
```

Of course, I want to know more about those processes now. The command `systemd-analyze blame` gives me a list of all processes started at boot time. They are sorted by the elapsed time before the process has fully launched. I usually only see the 10 biggest time hogs, as shown in Listing 1. In this particular case, I can ignore the candidates in lines 1 and 2 – they appear if the system performs unattended updates or upgrades on startup. Fortunately, they do not block any other processes and simply wriggle around a bit in the background. `systemd-analyze` does not calculate them for this reason, as can already be seen from the sums.

Listing 1: systemd-analyze blame | head

```
1min 3.753s apt-daily.service
    40.702s apt-daily-upgrade.service
     8.649s fail2ban.service
     4.246s networking.service
     3.327s smokeping.service
     2.622s apache2.service
     2.303s mysql.service
     2.206s postfix.service
     1.835s dev-sdb1.device
      993ms munin-node.service
```

Serial Plot

With the `plot` parameter, I can create a kind of timeline in SVG format, which shows the boot process in a very clear graphical way. For this to work, the *graphwiz* package must be installed. The command is then:

```
systemd-analyze plot > systemd-boot.svg
```

Figure 1 shows a small section of the results. The start time of the individual processes is symbolized here by a red bar. Not only the start order and times, but all the dependencies between the services, can be summarized by the tool in the form of an SVG graphic:

```
systemd-analyze dot | dot -Tsvg > ➥
  dependencies.svg
```

Caution: The result is really huge and probably only makes life easier for owners of A1-sized plotters. ■

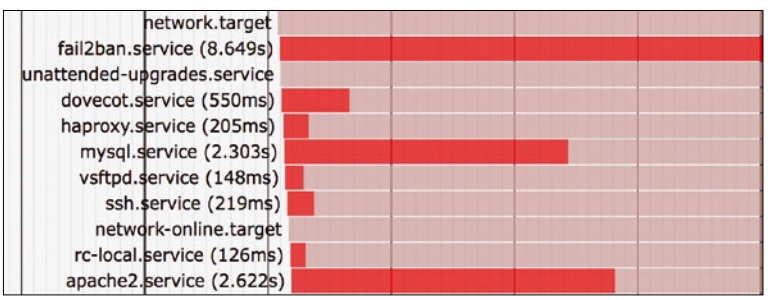

Figure 1: An SVG format timeline showing each service's system start times.

Info

[1] Bootchart: http://www.bootchart.org

The Author

Charly Kühnast manages Unix systems in the data center in the Lower Rhine region of Germany. His responsibilities include ensuring the security and availability of firewalls and the DMZ.

Info Tubbies

02

The name of the tool that columnist Charly Kühnast recommends this month may sound like the Teletubbies, but it is but far from infantile when it comes to functionality. In fact, inxi provides detailed and precisely controllable hardware and system usage information for the host computer. By Charly Kühnast

Every admin knows how to retrieve information about the system on which they are working. How many cores does the CPU have? `cat /proc/cpuinfo`! Is `eth3` a gigabit interface? `ip l sh`! But instead of many tools, you can just use one: inxi [1].

Suppose I need an overview of a machine with which I don't normally work. Then, I call inxi without any parameters and get some basic information about the hardware (CPU, clock speed, RAM, and disk size) and the system (kernel and shell processes).

If I want to see a few details, the `-F` parameter provides information on the video and audio hardware, partitioning, RAID, temperatures, and fan speeds (Figure 1).

If I'm only interested in a particular component, I can target this with specific parameters, such as `-C`, `-A`, and `-G`, which stand for information on the CPU, audio, and graphics, respectively. Information on the RAM is returned after a (lowercase!) `-m`, which takes some getting used to.

Memory Details

Running with root privileges, inxi tells me more about the RAMe: Apparently four 2GB DDR modules are plugged into my test machine and clocked at 1600MHz – yes, this is a fairly ancient beast (Figure 2). The `-c4` parameter shown in Figures 1 and 2 is responsible for the color scheme. The default color scheme is not easily legible on terminals with a light background, but thanks to the plethora of options from `-c1` to `-c32`, selectable sets are available to suit your taste.

I can even talk inxi into a spot of simple process monitoring. If I want to know which five (this is the default value) processes are currently hogging the most RAM, `inxi -t m` will help me find out. If I want to see the top 10 processes, I enter `-t m10`.

If I hear the CPU fan humming, on the other hand, I just need to replace the `m` with a `c` to view the processor load. You can also combine the two: `inxi -t cm 10` returns the top 10 RAM and CPU hogs.

At the end of this informative newscast on the computer, I'll take a quick look at the weather: `inxi -w Berlin, Germany` tells me what the situation looks like outside the server room. ■

```
charly@funghi:~$ inxi -F -c4 -x
System:    Host: funghi Kernel: 4.4.0-78-generic x86_64 (64 bit gcc: 5.4.0) Console: tty 0
           Distro: Ubuntu 16.04 xenial
Machine:   Mobo: ASUSTeK model: P8Z68-V PRO v: Rev 1.xx Bios: American Megatrends v: 3603 date: 11/09/2012
CPU:       Quad core Intel Core i7-2600K (-HT-MCP-) cache: 8192 KB
           flags: (lm nx sse sse2 sse3 sse4_1 sse4_2 ssse3 vmx) bmips: 28020
           clock speeds: max: 5900 MHz 1: 1629 MHz 2: 1599 MHz 3: 1607 MHz 4: 1608 MHz 5: 1606 MHz
           6: 1697 MHz 7: 1602 MHz 8: 1689 MHz
Graphics:  Card: NVIDIA Device 1c03 bus-ID: 01:00.0
           Display Server: X.org 1.18.4 drivers: nvidia (unloaded: fbdev,vesa,nouveau)
           tty size: N/A Advanced Data: N/A out of X
Audio:     Card-1 NVIDIA Device 10f1 driver: snd_hda_intel bus-ID: 01:00.1 Sound: ALSA v: k4.4.0-78-generic
           Card-2 Intel 6 Series/C200 Series Family High Definition Audio Controller
           driver: snd_hda_intel bus-ID: 00:1b.0
Network:   Card: Intel 82579V Gigabit Network Connection
           driver: e1000e v: 3.2.6-k port: f040 bus-ID: 00:19.0
           IF: eth0 state: up speed: 1000 Mbps duplex: full mac: 14:da:e9:4f:8c:cb
Drives:    HDD Total Size: 512.1GB (16.5% used) ID-1: /dev/sda model: TS256GSSD340 size: 256.1GB
           ID-2: /dev/sdb model: TS256GSSD340 size: 256.1GB
Partition: ID-1: / size: 227G used: 72G (34%) fs: ext4 dev: /dev/sdb1
           ID-2: swap-1 size: 8.55GB used: 0.00GB (0%) fs: swap dev: /dev/sdb5
RAID:      No RAID devices: /proc/mdstat, md_mod kernel module present
Sensors:   System Temperatures: cpu: 29.8C mobo: 27.8C gpu: 0.0:
           Fan Speeds (in rpm): cpu: 0
Info:      Processes: 195 Uptime: 17:24 Memory: 586.5/7950.4MB Init: systemd runlevel: 5 Gcc sys: 5.4.0
           Client: Shell (bash 4.3.481) inxi: 2.2.35
```

Figure 1: "Extensive" is probably the best description of what inxi bundles into its system overview here.

```
charly@funghi:~$ inxi -C -c4
CPU:       Quad core Intel Core i7-2600K (-HT-MCP-) cache: 8192 KB
           clock speeds: max: 5900 MHz 1: 1603 MHz 2: 1679 MHz 3: 1600 MHz 4: 1609 MHz 5: 1628 MHz
           6: 1668 MHz 7: 1977 MHz 8: 1674 MHz
charly@funghi:~$ sudo inxi -m  -c4
Memory:    Array-1 capacity: 32 GB devices: 4 EC: None
           Device-1: ChannelA-DIMM0 size: 2 GB speed: 1600 MHz type: DDR3
           Device-2: ChannelA-DIMM1 size: 2 GB speed: 1600 MHz type: DDR3
           Device-3: ChannelB-DIMM0 size: 2 GB speed: 1600 MHz type: DDR3
           Device-4: ChannelB-DIMM1 size: 2 GB speed: 1600 MHz type: DDR3
charly@funghi:~$
```

Figure 2: If you give inxi root privileges, you are rewarded with precise information on the computer's memory banks.

Info

[1] inxi: https://github.com/smxi/inxi

03 Great Shuttle Service

When he doesn't want to deal with OpenVPN version conflicts or congestion control problems during TCP tunneling, Charly catches a ride on sshuttle.

By Charly Kühnast

In untrustworthy networks, I let OpenVPN tunnel my laptop. There are certainly alternatives, and I would like to present a particularly simple one: sshuttle [1]. As the name suggests, the tool relies on SSH. The tunnel's endpoint is a leased root server, just like with OpenVPN. Sshuttle is very frugal. It only needs SSH access with user privileges on the server; root privileges are not necessary. Additionally, Python must be installed on the server – that's it.

This is because sshuttle loads and executes the required Python code on the server after the SSH connection is established. It also avoids version conflicts between server and client software. The following command is all it takes to set up the tunnel:

```
sudo sshuttle ➥
  -r <User>@<Server>:<Port> 0/0
```

You can leave out the port number if it is the SSH standard port 22. The `0/0` means that Linux should direct all connections into the tunnel. However, this means that I cannot reach other devices in the local network. To keep the local LAN still visible, I define it as an exception using the `-x` parameter:

```
sudo sshuttle ➥
  -r --dns <User>@<Server> 0/0 ➥
  -x 192.168.2.0/24
```

`--dns` is included here. This means that DNS queries also run through the tunnel, which does not happen automatically. This is sshuttle's Achilles heel: It only transports TCP; ICMP and UDP do not pass through the tunnel, apart from DNS.

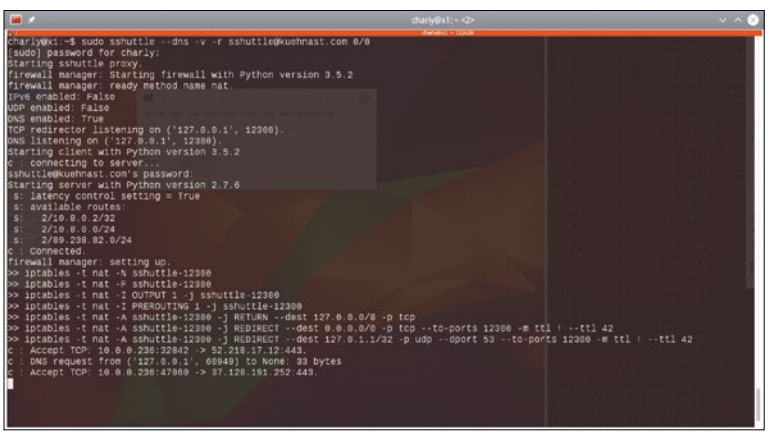

Figure 1: Sshuttle builds a VPN for a server. Because of `-v`, the messages are more extensive than without.

Congestion Alert

Whereas other VPN technologies work at packet level and rely on TUN/TAP devices, sshuttle works at session level. It assembles the TCP stream locally, multiplexes it over the SSH connection, while keeping the status, and splits it into packets again on the destination side.

This avoids the TCP-over-TCP problem which plagues other tools such as OpenVPN: TCP has an overload control (congestion control). The protocol defines a performance limit on the basis of dropped packets. If you tunnel TCP over TCP, you lose congestion control for the inner connection, which can lead to bizarre error patterns. Sshuttle is immune to the problem.

Verbose parameters can help if you do need to troubleshoot. Figure 1 shows a connection setup with `-v`. With the verbose option, sshuttle is very long-winded, so I recommend redirecting the output to a file that can be evaluated in peace. My conclusions: Sshuttle is an excellent and simple VPN for people who can do without UDP and ICMP. ■

Info

[1] sshuttle: https://github.com/apenwarr/sshuttle

Drive-In

04

Having a good backup is a matter of course for sys admin columnist Charly Kühnast, but devices could still fall victim to fire or theft some day. Because he has enough free space on Google Drive, he doesn't need to search long for a solution. The only thing missing is the right tool. By Charly Kühnast

I never wanted to have to lament lost data, which is why I have functioning backups at home. Nevertheless, having an offshore backup seemed to be a good idea in case of a chain of ridiculous circumstances (i.e., absolute disaster). As an Android user, Google gives me 15GB of disk space, which I don't currently use. But the right tool is still missing: Rclone. Completed binary packages for Linux, Windows, Mac OS, the BSDs, Plan 9, and Solaris can be downloaded from Rclone's website [1]. If you would prefer to build your own and you have a Go compiler on your system, you can proceed with the command

```
go get -u -v github.com/ncw/rclone
```

at the start. I launch the configuration of the storage back ends with `rclone config` and, with `n`, launch the path for *New remote*. I use `mygdrive` for the name now requested for the remote connection. Figure 1 makes it clear that there should be something for everyone in the supported storage services – even generic FTP/SFTP to set up your own backup corner on a rented server or web space. After choosing *Google Drive* and the automatic configuration, a browser opens so that I can let Rclone access my file storage on Google. This is the end of the configuration.

To check what I've saved on the remote side, the following command suffices:

```
rclone ls mygdrive:
```

The colon is mandatory. Because my drive is empty, there is no output. To test it, I copy my Apache config files into the drive:

```
rclone copy -L -v /etc/apache2/ ➥
  mygdrive:backup
```

This takes a second, because Google has relatively aggressive rate limiting (about two files per second). I check straightaway whether the files have arrived in Google's web interface (Figure 2).

```
root@funghi: ~ 67x45
root@funghi:~# rclone config
2017/07/16 15:09:01 NOTICE: Config file "/root/.config/rclone/rclo
e.conf" not found - using defaults
No remotes found - make a new one
n) New remote
s) Set configuration password
q) Quit config
n/s/q> n
name> mygdrive
Type of storage to configure.
Choose a number from below, or type in your own value
 1 / Amazon Drive
   \ "amazon cloud drive"
 2 / Amazon S3 (also Dreamhost, Ceph, Minio)
   \ "s3"
 3 / Backblaze B2
   \ "b2"
 4 / Dropbox
   \ "dropbox"
 5 / Encrypt/Decrypt a remote
   \ "crypt"
 6 / FTP Connection
   \ "ftp"
 7 / Google Cloud Storage (this is not Google Drive)
   \ "google cloud storage"
 8 / Google Drive
   \ "drive"
 9 / Hubic
   \ "hubic"
10 / Local Disk
   \ "local"
11 / Microsoft OneDrive
   \ "onedrive"
12 / Openstack Swift (Rackspace Cloud Files, Memset Memstore, OVH)
   \ "swift"
13 / SSH/SFTP Connection
   \ "sftp"
14 / Yandex Disk
   \ "yandex"
Storage>
```

Figure 1: Rclone helps you with all major cloud driver providers.

With History

Google provides a small versioning drive. If I upload changed Apache files again, the drive recognizes this and keep both files for a maximum of 30 days or 100 changes, whichever comes first. I now use the `sync` sub-command instead of `copy`:

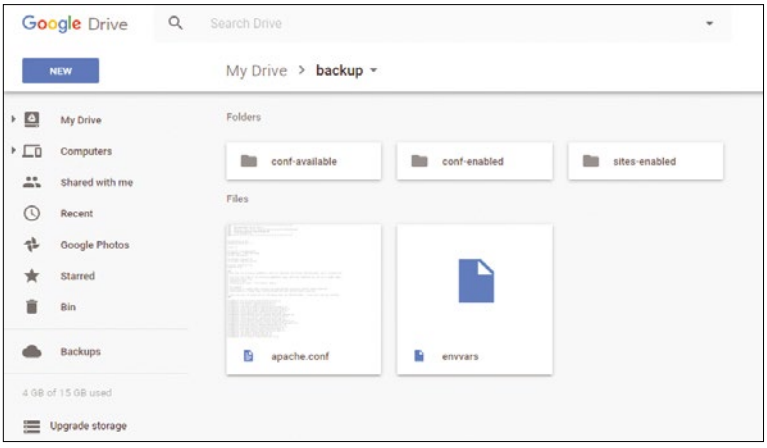

Figure 2: Google's web front end shows that the example files transmitted via Rclone have arrived safely.

```
rclone sync -L -v /etc/apache2/ ➥
  mygdrive:backup
```

Rclone synchronizes the local and remote pages and only copies the files that have changed. Files that you delete locally meet the same fate on Google Drive – not vice versa. Rclone cannot manage to synchronize both sides like Rsync. However, I can encrypt files for storage by choosing *Encrypt/Decrypt a remote* in `rclone config`.

Rclone is still in active development, so you can watch it closely at work. ■

Info

[1] rclone: https://rclone.org/downloads/

Alpha Beast

05

In this issue, sys admin columnist and tool veterinarian Charly Kühnast invites Sysdig, the jack-of-all-trades among system diagnostic tools, into his surgery for a quick checkup. The project promises to unite the functionality of lsof, iftop, netstat, tcpdump, and others. By Charly Kühnast

Where an alpha beast claims to replace an entire herd, the bar is naturally fairly high. Of course, the Wireshark authors, who are also the people behind the Sysdig [1] project, are no beginners. The software only performs well if you have root privileges; otherwise, it can't access all the required system areas. If you launch the tool without parameters, a steady stream of system messages scrolls by: It meticulously logs every single syscall. To thin out the thicket, Sysdig uses what it calls chisels. You can find out which chisels exist with the `sysdig -cl` command.
The chisels are sorted into categories (Net, IO, application, logs, and so on). For example, the Performance category has a chisel named `netlower`. I decided to pass in a time value of 10 milliseconds as a parameter:

```
sysdig -c netlower 10
```

Now Sysdig keeps listing processes whose network IO is slower than 10 milliseconds – on my home network, this means the SmokePing probes to the garden Raspberry Pis and some Munin connections.
You can output a list of the processes with the most frequent mass storage accesses by typing:

```
sysdig -c topprocs_file
```

The following reveals the entity causing the most network traffic:

```
sysdig -c topconns
```

A replacement for `top` can be found in:

```
sysdig -c topprocs_cpu
```

The built-in automatic analysis of bottlenecks is particularly informative. Typing

```
sysdig -c bottlenecks
```

generates a list of processes whose syscalls take a suspiciously long time. This is a great approach to searching for bottlenecks.

Depth on the Interface

If you like a more interactive approach, try `csysdig`. The tool displays the information provided by Sysdig in a continuously updated ncurses interface. Called without parameters, the start screen reminds one of `htop`, but pressing F2 takes you to a list of *Views* that correspond to the categories to which Sysdig assigns its chisels, and you can access them quickly and easily.
For example, if you choose the *Spectrogram-File* view, you are treated to a graphic like that shown in Figure 1: It shows the file access latency distribution, in which each line represents one second. At the time of grabbing the screenshot, an `apt dist-upgrade` was running, hence the high read and write load highlighted in red.
The *Views* overview showcases one of the specialities of Sysdig and Csysdig: You can restrict analyses to applications that run in containerized systems such as Docker or Kubernetes. Thus, admins can quickly and easily identify any performance fluctuations in containerized software.
My conclusions: Used only as a replacement for `top` and `netstat`, Sysdig is like taking a sledgehammer to crack a nut, but the many easily parameterized analyses of file and network latencies are a real help. If I have to dig down into individual syscalls, I can save a trace file and filter it until I find what I want. Here, at last, you can finally see the signature of the Wireshark makers. ■

Info

[1] Sysdig: https://www.sysdig.org

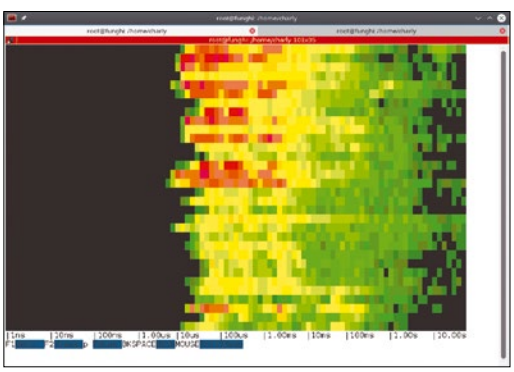

Figure 1: Abstract art? Nope: Concrete latencies during file access as a spectrogram, applied over time.

06 Health Spa

Charly and XML have never been best friends. However, because it was vital for him to have an excellent climate indoors, he plucked up his courage and considered XMLStarlet. By Charly Kühnast

A few smart home components from the HomeMatic system are permanent guests in my bathroom – they are for controlling the temperature in the room. A radiator thermostat and three sensors detect whether doors and windows are open. I use a central unit called CCU2 to set the climate I want and in which scenario. Their web interface seriously tests my patience. If, for example, I name a window contact `Bathroom_Window_North` instead of `NEQ1651969` – because that's what it's called – I can't rely on the web interface to use the new name in all views (Figure 1).

The issue of security is another low point: The user interface is not password protected by default. I may be able to set one, but because CCU2 does not speak HTTPS, the password would fall into the clutches of the next sniffer anyway. Access to the script engine on port 8181 is also completely insecure. Nevertheless, the benefit is great because it allows me to use HTTP requests to read out the status of HomeMatic components and even set values.

Given this information, however, it should be clear that nobody is allowed to simply put the CCU2 onto the Internet via port forwarding, at least not anyone who doesn't want to invite all hackers to play poltergeist in their own home. Anyone wanting to view their home heating system from Jamaica will need a good VPN.

XML Instead of a GUI

So, I have a tool at hand for operating the CCU2 without its web interface. To receive the status of a window contact, I type the following into the console:

```
wget http://10.0.0.231:/x.exe?Response=➥
  dom.GetObject(%27BidCos-➥
  RF.NEQ1651559:1.STATE%27).Value()
```

The CCU2 responds very quickly with (brrr!) XML:

```
<xml>
  <exec>/x.exe</exec>
  <sessionId/>
  <httpUserAgent>
    User Agent: wget
  </httpUserAgent>
  <Response>true</Response>
</xml>
```

I will suppress my XML aversion to help machines communicate with other machines – because that's just what XML is designed for. The `true` between the `Response` tags means that the window is closed. I can also retrieve the thermostat using similar commands and then set new temperature values. This means I can set up my own CCU2 GUI-free control using a few Bash scripts. Now I'm still missing an XML parser for the command line – with which I can finally angle for the XMLStarlet [1].

I previously had the output of the `wget` command above written to the `bathwindow.xml` file. In this way,

```
xmlstarlet sel -T -t -m xml ➥
  -v Response $WDIR/bathwindow.xml
```

I now get the value between the `Response` tags. Fancy a ride through the hell of parameters? Just read the 17-page (!) PDF document [2]. ■

Info

[1] XML Starlet: http://xmlstar.sourceforge.net

[2] Documentation: http://xmlstar.sourceforge.net/docs.php

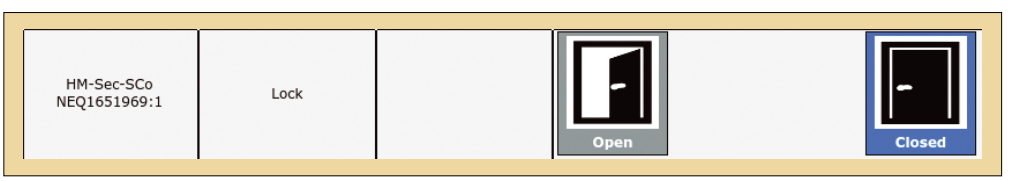

Figure 1: Window with a poor view: CCU2 cannot remember the name of window sensors.

The Hole Truth

07

A strange rule seems to dictate that the most useless products and services have the most annoying online advertising. Columnist Charly blocks the garish advertising for all computers on his network centrally with the Pi-hole tool, which is not only for Raspberry Pi devices. By Charly Kühnast

There are two irreconcilable camps in the discussion on the use of banners and skyscrapers on websites: One is populated by people who get annoyed by garish, flashing, fidgety advertising formats that remind them of neon signs from the 50s. An increasing number of these users simply reject advertising on the web as garbage. The opposing camp is occupied by website owners – amateur bloggers, to name just one example – for whom advertising is the only way to recoup their costs for servers and other things.

People who place ads on their websites usually source them from one of several large commercial networks and simply create placeholders on the sites, which are then later replaced with the ads. Most people do not know exactly what advertising their site is showing at any given time.

The ad networks, in turn, allow the ad creators a great amount of freedom. It is no longer only images that are used here, but also JavaScript and the like. Criminals exploit this to display manipulated advertisements that scan the visitor's browser for vulnerabilities and – if they find any – install malicious software or animate the user to download applications of dubious repute. It can thus happen that visiting a highly reputable website actually infects your own PC with malware.

Those who are aware of this "malvertising" – a word composed from malware and advertising – or are simply annoyed by the visual overkill can turn to an ad blocker in the form of a plugin for their browser. But because I have many computers, I need a centralized, easy-to-maintain instance that solves the

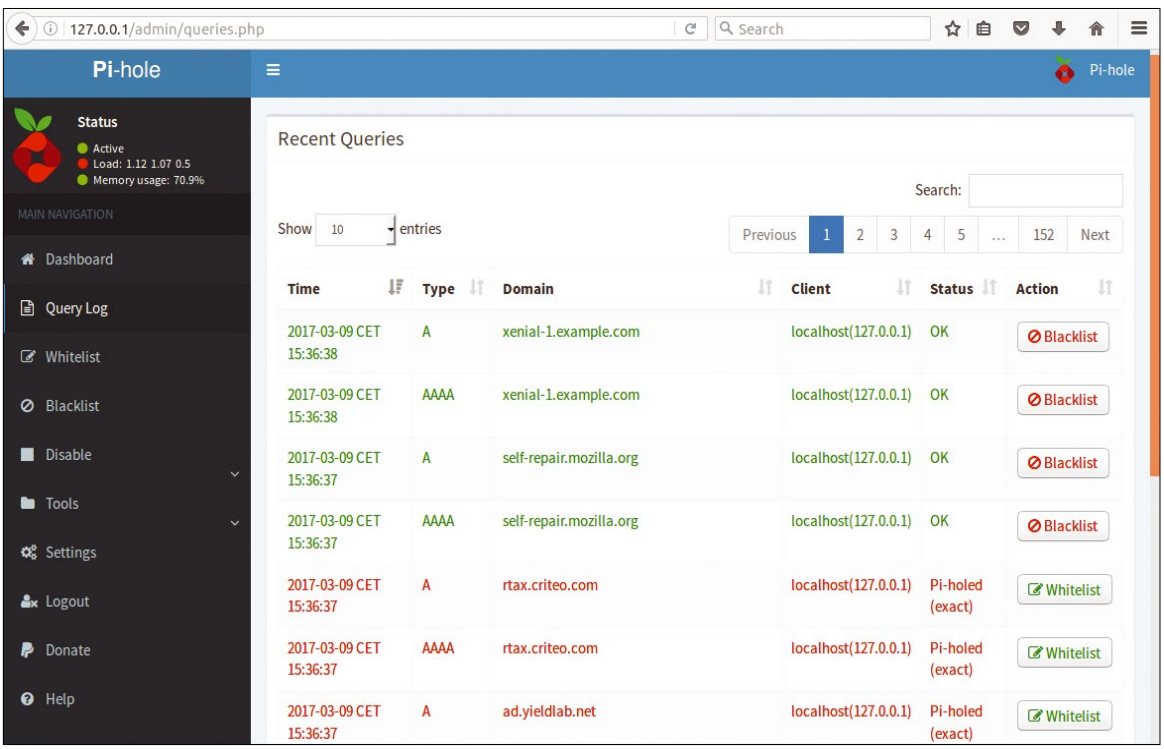

Figure 1: The Pi-hole UI, which is appealing both visually and in terms of content, presents various statistics and lists.

problem. It seems to me that Pi-hole [1] is extremely useful for this task. The tool got its name from the company that originally developed it for use on a Raspberry Pi, but it has long since been adapted for deployment on most standard Linux distributions.

Pi-hole is underpinned by the lean Dnsmasq DNS server with a special configuration. I entered Pi-hole as the DNS server on all my clients, and it now filters out the undesirable requests by the clients to ad networks and submits the remaining DNS requests to the regular DNS server.

Easy Install

The easiest way to install Pi-hole is with the following command:

```
curl -sSL ➥
  https://install.pi-hole.net | bash
```

Security-conscious admins might go into meltdown at the sight of this line, but the makers of Pi-hole have a way of calming them down. Of course, anyone can download the code, inspect it at their leisure, and then proceed with the install. Corresponding links and instructions can also be found online [1]. When done, the installer displays a randomly generated password for the web interface. You can access it on `http://<IP address>/admin`.

The web interface is visually appealing and offers a wealth of statistics (see Figure 1). You also can maintain your own blacklists and whitelists there. I make good use of this option, because I do not oppose advertising on the web as a matter of principle; I thus specifically add sites that I would like to support to the white list. In return, I punish sites that are badly behaved – because they install poster-sized pop-overs, for example – with a blacklist entry that filters their ads directly into a black hole.

Incidentally, there is no advertising at all on *pi-hole.net*. The project is free, and the code is open source. The authors simply ask you to donate an amount of your choosing. It would be nice if many people complied. ■

Info

[1] Pi-hole: https://pi-hole.net

More Lust for Load

08

How many users can the database take? When does a CMS throw in the towel? In order to explore performance limits, Charly Kühnast uses the Tsung load generator instead of human users as beta testers. By Charly Kühnast

If I want to test how much load a (perhaps even distributed) system can take, I launch a load generator. Some time ago, I praised Siege [1] in my column, which I still consider to be a good barrage tool. However, most load generators fire unrealistically from all barrels and do not simulate the behavior of a real user. Tsung [2] can do this better.

Tsung evolved in several evolutionary steps from a tool that ran load tests against Jabber/XMPP servers. Under the fear-inspiring name of idx-Tsunami, it was given multiprotocol capabilities. Since 2014, the development of idx-Tsunami has petered out. Tsung has simply taken the basis and continued developing Tsunami's codebase.

XMPP is still one of the services that Tsung can deploy to cause unrest on its test servers. On top of this, Tsung supports HTTP with and without TLS, WebDAV, SOAP, PostgreSQL, MySQL, AMQP, MQTT, and LDAP. All protocols are integrated via a plugin engine, so further protocols can follow at any time.

Planning the Attack Using XML

Using XML configuration files, the Tsung user designs their load test scenarios in detail. For example, you can stipulate that the requests should not only originate from one machine, but that several load generators (or clients) should play a key role. I can assign more or less work to clients with different performance characteristics by using weighting. I can also configure several back-end servers. IPv4 and IPv6 are allowed for the connections, also in mixed mode. The details of the requests that Tsung uses to stress the servers can be configured within a wide range.

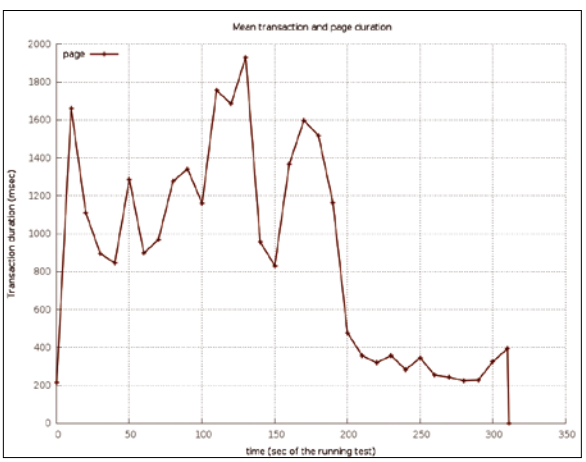

Figure 1: In this report, Tsung has the transaction time per call on the y axis and the sequence of the benchmark on the x axis.

In order to simulate realistic user behavior, the software does not torment the servers with constant fire on request, but instead makes well-planned pauses, just as a human user would if he or she were looking at the content of a website and then clicking on it.

Reception Center

If you want to make it even more realistic, use the supplied recorder: After starting, it records the behavior of one or more users, and Tsung replicates this session later. For example, variables can be brought into play when simulated users enter data in a search mask.

I can bundle a group of requests into one transaction. Tsung understands this term as a logically related request, for example: A user calls the website, authenticates themselves (say, using OAuth), then accesses the sub-page using the search function, and submits a search query.

Statistics Reveal All to the Administrator

In addition to the existing evaluations of the load behavior for the back-end servers, Tsung also generates reports on the performance of such transactions (Figure 1). These statistics are, as expected, more useful for the behavior of the systems in production than synthetic flak tests – and that's exactly what I like about Tsung. ■

Info

[1] Siege: https://www.joedog.org/siege-home/
[2] Tsung: https://github.com/processone/tsung/

09 Color Cast

The first time in our lives we got to a black-and-white Linux or Unix shell, most of us probably typed *ls* first. In a mixture of nostalgia and the knowledge that life is colorful, columnist Charly Kühnast plays a colorful trump card with *colorls*. By Charly Kühnast

`colorls` [1] is written in Ruby. If you don't have this language on your system yet, install it quickly:

```
sudo apt install ruby ruby-dev ↵
  ruby-colorize
```

Then you download a character set that you really like from Nerd Fonts [2] – say, Roboto Mono Nerd Font Regular. After unpacking the ZIP file, I moved the character set to the `/usr/share/fonts/truetype/roboto/` directory on my Ubuntu desktop; users of other distributions may need to change this path.

Why do I even get this font when there are a few dozen others preinstalled? Because Nerd Font's character sets are more extensive, containing more symbols, special characters, glyphs, and emojis than usual (Figure 1). Now I select the new font in my terminal's preferences. This fulfills the preconditions, and I can proceed to install `colorls` by typing:

```
sudo gem install colorls
```

The developers know that nobody types `colorls` 50 times a day. I recommend that you create an `lc` alias in your `~/.bashrc`:

```
alias lc='colorls'
```

If you use a light terminal background, you should always specify `--light` or, preferably, make it permanent by appending it to the `.bashrc` alias. The output then resembles that in Figure 2 – note the cute icons and bright colors. Light-shy workers can choose a variant optimized for dark terminals by specifying `--dark`.

No Blind Faith in Color

Speaking of the downside: `colorls` is a new implementation of `ls`, which does not implement all options identically and others not at all. My big favorites `-l` and `--sort=size` fortunately work. If you type `-f`, `colorls` only displays files; `-d` only displays directories.

If I want to see both, I have the choice between `--sd` (directories first – note the two dashes!) and `--sf` (files first).

If you would like a brightly colored `ls`, but have problems with `colorls` because of missing parameters, schedule a test run with `exa` [4], which doesn't offer any fancy icons but does support almost all the `ls` parameters and adds some on top. Especially with the defaults, `exa` impresses – for example, the `-h` parameter (human readable), which outputs file sizes in human-readable units instead of bytes, is implicit. ■

Info

[1] colorls: https://github.com/athityakumar/colorls

[2] Download character sets: https://nerdfonts.com/#downloads

[3] Nerd Font Cheat Sheet: https://nerdfonts.com/#cheat-sheet

[4] exa: https://the.exa.website

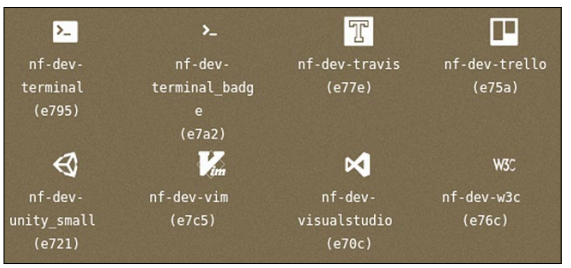

Figure 1: On Nerd Font's Cheat Sheet [3], you can see the extended character sets.

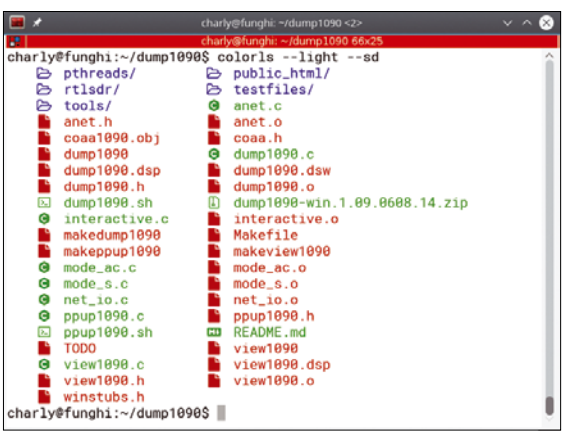

Figure 2: `colorls --light` prints files and directories in a light color scheme.

Always at Hand

10

Columnist Charly fights the fight for free SSL certificates with Let's Encrypt. He particularly likes the matching software client that takes care of everything – from certificate retrieval to web server integration. By Charly Kühnast

Although more and more web servers now use TLS, small businesses and amateur users are worried about the administrative overhead that accompanies SSL certificates. The Electronic Frontier Foundation, the University of Michigan, and the Mozilla foundation helped launch a project that addresses two pain points.

For one thing, it distributes Let's Encrypt [1] SSL certificates free of charge. For another – and this is where the project scores points compared with other free offerings – it delivers a Python client software tool that autonomously takes care of creating, validating, signing, and renewing certificates in good time. It even helps administrators integrate certificates with services running on their servers.

For Debian and its derivative distributions, you can pick up prebuilt packages. If you prefer to do it yourself – or are forced to do so – you can retrieve the Let's Encrypt client from GitHub with the following command:

```
git clone https://github.com/➥
  letsencrypt/letsencrypt
```

In the `letsencrypt` directory, you will now find a tool by the name of `letsencrypt-auto`. To use it to create a certificate for my example domain, `<mydomain>.com`, I would just type the following command:

```
sudo ./letsencrypt-auto certonly ➥
     --standalone -d <mydomain>.com
```

Here `certonly` creates a certificate but does not automatically integrate it with the web server configuration. I prefer to do this myself; after all, this part of the software is still beta and only works correctly for Apache installations that have not had their configuration manipulated by admins.

The `--standalone` parameter starts a web server for validation purposes – this is one of the reasons the tool needs root privileges. Next, `letsencrypt-auto` briefly stops the active web server; the server is only allowed to continue doing whatever it was doing after passing the SSL checks.

Web Server Integration

The certificates generated here – `cert.pem`, `chain.pem`, and `fullchain.pem`, which combines the previous two, and the private key `privkey.pem` – end up in the `/etc/letsencrypt/live/<mydomain>.com/` directory. To integrate this with an Apache web server, I need to pay attention to the Apache version number. Listing 1 shows the required instructions.

To renew the certificates, which are only valid for 90 days (Figure 1), automatically, you can simply run `letsencrypt-auto` with the `renew` parameter within this period. ■

Info

[1] Let's Encrypt: https://letsencrypt.org

Listing 1: Integrating Certificates

```
01 # Apache 2.4.8 and later:
02 SSLCertificateFile /etc/letsencrypt/live/mydomain.com/fullchain.pem
03 SSLCertificateKeyFile /etc/letsencrypt/live/mydomain.com/privkey.pem
04
05 # Pre-Apache 2.4.8:
06 SSLCertificateFile /etc/letsencrypt/live/mydomain.com/cert.pem
07 SSLCertificateChainFile /etc/letsencrypt/live/mydomain.com/chain.pem
08 SSLCertificateKeyFile /etc/letsencrypt/live/mydomain.com/privkey.pem
09
10 # Nginx:
11 ssl_certificate /etc/letsencrypt/live/mydomain.com/fullchain.pem;
12 ssl_certificate_key /etc/letsencrypt/live/mydomain.com/privkey.pem;
```

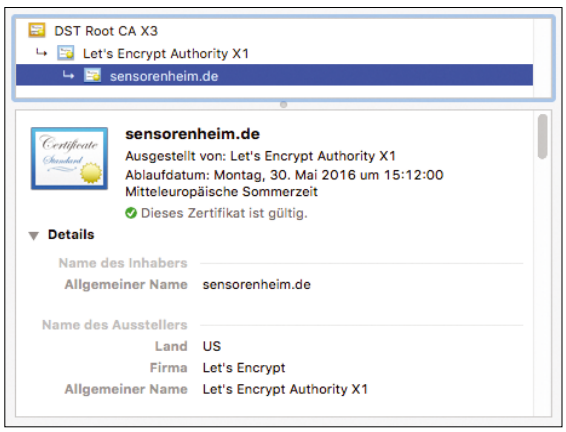

Figure 1: The browser reporting that the Let's Encrypt certificate is perfectly okay, but it expires May 30.

Quick directory change at the command line

Jump!

Bd, autojump, and Fasd improve the workflow for command-line aficionados thanks to quick navigation in the filesystem. By Ferdinand Thommes

With a little practice, working at the command line lets you feel the true power of Linux. However, the standard tools are not always as convenient as they are powerful, or as the user would hope.

If you frequently work at the command line, you most likely use the `cd` "change directory" command on a daily basis. In this article, I introduce a few helpers that promise greater convenience and speed, especially when changing directories in deeply nested paths. Bd, autojump, and Fasd are available as tools for different shells.

Inconvenient

The various Linux shells already provide some help with improved navigation in the command-line directory jungle, starting with finding out where you are. The `pwd` command helps by outputting your working directory's path. If you now want to jump up one or more levels, you can usually do this with:

```
cd ..
```

The two periods mean one step up in the direction of the root. You can jump up two steps by using:

```
cd ../..
```

However, frequent directory changes involve much typing, which can lead to errors. If you have to switch often between two directories in different places in the directory tree, the `cd -` command helps by taking you to the last directory you visited.

The directory stack provides further assistance: Multiple directories can be piled onto a stack. Bash provides the `pushd`, `popd`, and `dirs` commands to navigate in this stack [1]. If you replace `cd` with `pushd` for the directory change, the shell always places the directory to enter at the top of a stack. It remembers the stack – so you don't need to – and displays it after the command. Using `popd`, you can work your way back up the stack. For orientation, the `dirs -l -v` command displays the entire stack. To create the stack, you first have to access the directories with `pushd` to add them to the stack, which proves useful if you have to switch between a few directories in a working session: The tool consecutively numbers the stack, and you can jump to the folders with the use of these numbers.

However, bd, autojump, and Fasd offer a more convenient approach; all the major distributions have them in their repositories. The `whohas` command, installed via your package manager, tells you where to find them.

Typing `whohas autojump` gives you a list of distributions with the package, including the version and the respective branch.
You also can set up the bd [2] script for many distributions via a package manager. After installing, enter the two commands from Listing 1 to facilitate running bd. If bd is not available in the Linux derivative that you use, install it manually following notes from the project site on GitHub.

Shorter Return Path

Imagine that you have used `cd` to enter the `/home/fritz/foo/bar/bat/test/bd/` directory and now want to go back to `/home/fritz/foo/bar/`. You would usually enter:

```
cd ../../../
```

With the bd script you just installed, `bd bar` would do the trick: bd supports autocompletion, so you do not have to type out the directory names; usually you just need two or three letters.
Bd only operates backward, not forward. But it can also be used with other commands, for example: `ls`, `du`, `zip`, and `tar`. The command

```
ls -l `bd ba`
```

lists the content of the `/bar` directory, even though you are still in `/bd`. The call

```
du -cs `bd fr`
```

displays the size of `/fritz` (Figure 1).

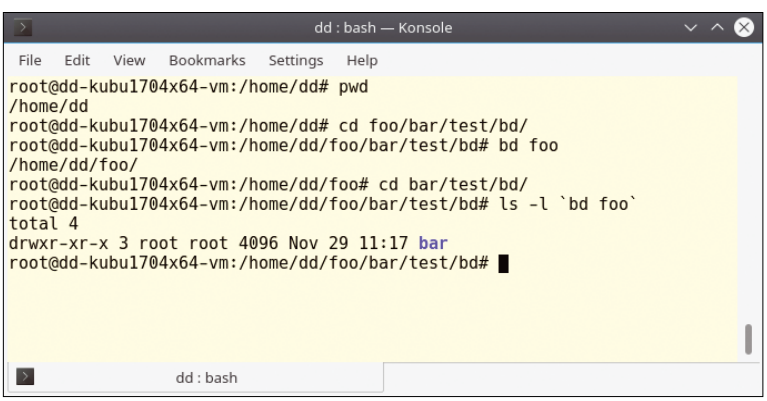

Figure 1: The bd script quickly takes you back to the directory tree and allows additional commands to be integrated.

autojump

Autojump [3] makes use of the basics of the directory stack described above and further expands on them. It also facilitates navigation in the directory tree in both directions, unlike bd.
Autojump can be used under Linux, Mac OS, and Windows with the Bash (from version 4.0), Zsh, and fish shells and experimentally with the tcsh and Clink shells.
Most distributions offer autojump packages. Under Debian and its subsidiaries; you can add the tool to the system using the command from Listing 2. Autojump works with a database that you initially have to fill by working with `cd` for a while. Alternatively, you can target folders that you use often.
You will find the database under `~/.local/share/autojump/autojump.txt`. You can query the contents of the database at any time using `j --stat`. The `j --purge` call removes directories that no longer exist from the database.
If you also want to test Fasd [4], the final helper discussed in this article, it makes sense to install it now, because it also benefits from the actions for filling the database. To do this, type the entries from Listing 3 after the install.

Listing 1: Running bd

```
$ echo alias bd='. bd -si' >> ~/.bashrc
$ source ~/.bashrc
```

Listing 2: Installing autojump

```
$ echo '. /usr/share/autojump/autojump.sh'  >> ~/.bashrc
$ source ~/.bashrc
```

Listing 3: Installing Fasd

```
$ echo eval '$(fasd --init auto)'  >> ~/.bashrc
$ source ~/.bashrc
```

Database Based

You can easily jump into directories that autojump keeps in its database using the `j` wrapper. For example, `j Dow` would take you to the `~/Downloads` directory. In the same way, `j loa` also takes you there. The depth of the directory hierarchy is not important; the main thing is that you visited the desired directory at least once using `cd`.
If you want to switch to your file manager for a better overview, `jo Dow` opens the directory for Plasma in Dolphin or for Gnome in Nautilus (Figure 2). If you just enter `j`, the command jumps to the most frequently visited directory, for which Autojump uses internal weighting.
If you call the content of the database with `j --stat`, you will see different numbers in front of the respective paths, for example:

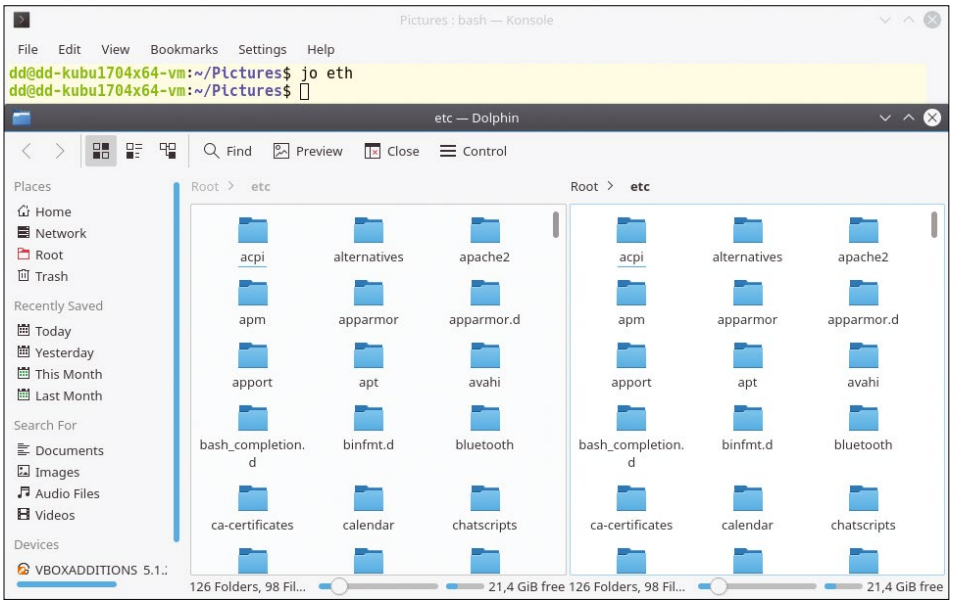

Figure 2: If you have lost track at the command line, autojump will also open the desired directory in the file manager.

```
22.4: /home/devil/Downloads/piwik
```

This sequence of digits is the internal weighting (Figure 3). As an example, the following directories are in the database:

```
30 /home/user/mail/inbox
10 /home/user/work/inbox
```

If you enter `j in`, high weighting causes autojump to switch to `/home/user/mail/inbox/`, whereas `j w in` would be the correct command to access the lower weighted directory. You can use the `--increase` (`-i`) and `--decrease` (`-d`) parameters to change the weighting, and `j --help` for additional information about syntax.

Alternatively, if you have several similar directories, you can display the numbered entries by means of tab-based autocompletion and then select the corresponding entry by number.

A helpful tip when using autojump: Sometimes the database suddenly empties, which can be extremely annoying, because you then have to start indexing from scratch. Sometimes it is better to back up the database manually.

Turbo

Fasd [4], the final tool covered in this article, was inspired by autojump. However, the software exceeds its role model: Fasd no longer restricts the navigation options to the directory tree, and it also indexes the data. Fasd works with the Ash, Bash, Zsh, mksh, pdksh, dash, and BusyBox shells under Linux, as well as `/bin/sh` in FreeBSD 9 and OpenBSD.

Fasd lets you open files by entering just a few characters in Vim or your default editor. You can view PDFs or play movies and music with MPlayer or your favorite player. The letters in the Fasd acronym stand for predefined aliases that process different types of data.

When you enter `f`, the tool shows you the visited files, including their weighting. Entering `d` provides the same result for the directories, and `a` lists both. `zz` takes you to a view in which you select an entry by the prefixed number (Figure 4).

Defining additional aliases lets you open various files: for example, PDFs (Figure 5). In turn,

```
devil@siductionbox:~$ j --stat
4.1:    /usr/share/doc
10.0:   /var/cache/apt/archives
10.0:   /home/devil/Downloads/_Series
17.3:   /home/devil/Downloads/thunderbird/defaults
17.3:   /home/devil/work/Linux-User
20.0:   /home/devil/Downloads/thunderbird/defaults/messenger
22.4:   /home/devil/Downloads/piwik
22.4:   /home/devil/Downloads/thunderbird
24.5:   /home/devil/Downloads/chrome
48.0:   /home/devil/Downloads
----------------------------------------

195:    total weight
10:     number of entries
0.00:   current directory weight

data:   /home/devil/.local/share/autojump/autojump.txt
devil@siductionbox:~$
```

Figure 3: The numbers in the database are used for internal weighting and state which directories were visited and how often.

```
devil@siductionbox:~$ z th
devil@siductionbox:~/Downloads/thunderbird$ cd
devil@siductionbox:~$ pwd
/home/devil
devil@siductionbox:~$
devil@siductionbox:~$ zz
5       6           /snap/core
4       15          /home/devil/Downloads/thunderbird
3       15          /snap
2       17.4        /var/cache/apt/archives
1       21.3181     /home/devil/Downloads/_Series
> 4
devil@siductionbox:~/Downloads/thunderbird$
```

Figure 4: In Fasd, you can select the desired directory using `z` and a few characters. You can select from the database by entering `zz`.

you can create the following proxy in `.bashrc`:

```
alias o='a -e xdg-open'
```

Now source the file, as shown in Listings 2 and 3. The following also applies: Like autojump, Fasd only detects objects that have already been indexed.
Movies and music also can be started with just a few keystrokes. For this to work, enter

```
alias m='f -e mplayer'
```

in `.bashrc`. Alternatively, use a different player or change the `m` call switch to another. Then, simply start the movies or music using `m<part of title>`.
Like autojump, Fasd also lets you use letters from the middle of the name. The aliases' structure is simple: `-e` stands for execute. For files, select `f`; for directories, `d`. If the alias includes both, as with xdg-open [5], select `a`. On GitHub, the project provides an overview of the tool's many other possibilities.

Conclusions

All three tools presented in this article have one thing in common: They do not run in drop-down terminals, such as Yakuake or Guake, just in consoles or terminals. However, you will not read this anywhere, which led to me occasionally doubting the tools' capabilities in the lab. All three helpers are meaningful, and they build on each other. If you do not constantly work at the command line, bd makes it easy to navigate without having to learn something new. Autojump offers much more and jumps into directories in random directions. This suits the workflow of users who frequently work on servers and like to handle administrative work at the command line. After the introduction phase, autojump saves some time here.
Fasd expands the feature set again, because it also indexes files and thus takes control of all the files on the computer. It also offers professional options to further customize the approach to suit your own needs. ■

Info

[1] Shell built-ins: https://www.howtoforge.com/tutorial/linux-command-line-tips-tricks-part-2/
[2] bd: https://github.com/vigneshwaranr/bd
[3] autojump: https://github.com/wting/autojump
[4] Fasd: https://github.com/clvv/fasd
[5] xdg-open: http://www.ubuntugeek.com/tag/xdg-open-example

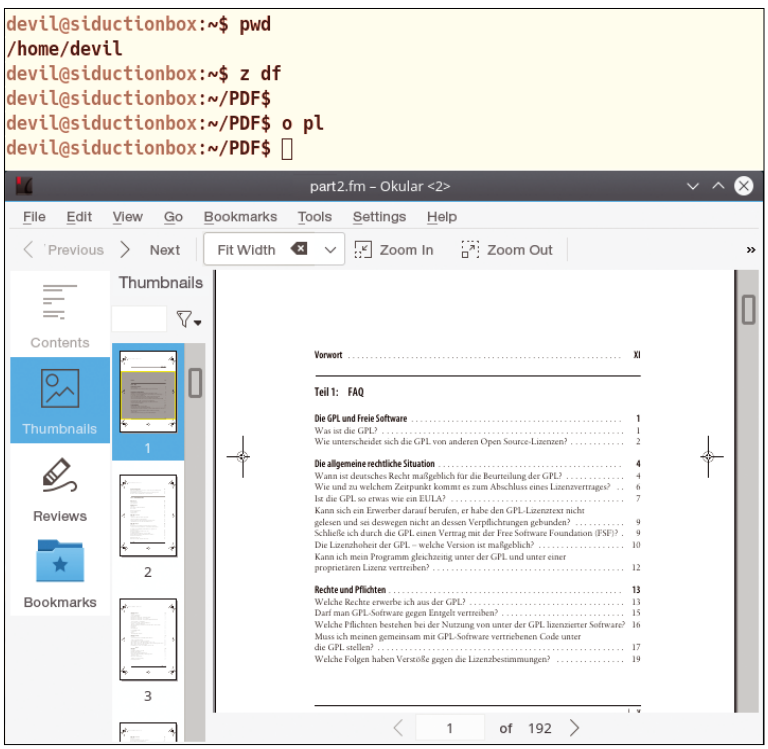

Figure 5: Defining additional aliases extends Fasd to display PDFs and other formats.

Automate data backup at the command line

Automatic Backup

Backing up data is an unpopular task that many users – and even some administrators – consider a chore, prompting us to take a look at some command-line automatic backup programs. By Erik Bärwaldt

Linux users have access to numerous backup tools. Administrators who like working with SSH appreciate that servers of any size and design can be backed up with command-line programs. However, the differences in terms of features are quite considerable (see Table 1 for an overview). Not every program is suitable for every application scenario. In this article, I investigate which tools work for which environments.

Server vs. Desktop

Home users often store large volumes of data on their computers, similar in volume to those found on servers in small businesses. High-definition video collections, as well as audio files with lossless compression and photo folders, are real memory hogs. New data is often added, but once stored, the data hardly ever changes.

On the other hand, you will also often find small files (such as correspondence, tables, presentations, and databases) on server systems. These data collections are constantly changing through modifications, such as newly created records or added documents. Accordingly, backup strategies must take existing data resources into account to guarantee rapid reconstruction in the event of data loss.

Differential vs. Incremental

Administrators distinguish three backup strategies: full backup, differential backup, and incremental backup. The full backup, a copy of the existing data, is always the first backup in any plan – subsequent backups follow as differential or incremental backups. Whereas differential backups always save changes since the last full backup, incremental backups only save modifications relative to the last backup of any kind.

The differential backup procedure requires more space for individual backups, but in an emergency, only the full backup and the last backup will help you recover the entire database. Although the incremental method uses less disk space, all incremental backups need to be re-installed in the correct sequence during the restore starting from the full backup. If a small backup is missed, the database is no longer reconstructible.

Before selecting a command-line backup software, I recommend initially performing a careful analysis of your data collection and data growth, so you do not accidentally select a program that is unsuitable for your specific IT environment. Differential backup strategies are more likely to be used for databases that have relatively few large files and moderate regular modifications, whereas incremental backups are better suited to typical office environments. Regardless of which you choose, you should always run at least one full backup per week.

Desktop users who want to back up their own databases without root privileges do not have a huge choice of backup software for the command line, which obviously requires some knowledge of the command syntax. For users, it is important to be able to run a backup as smoothly and reliably as possible. The end user will only use the backup software – and actually perform the backup – if it is quick and easy to use.

Ideally, the same software can be used for mixed environments with both a backup server and additional desktop backups by users. Using the same software saves you the hassle of having to know the syntax of two programs – and thus avoids any associated errors.

Attic

The Attic backup program, which is written in Python, can be found

in the repositories of some Linux distributions, such as Mageia, openSUSE, ROSA, or Slackware Linux; it can be installed conveniently using the respective package managers. The project page also provides the source code for download. Detailed documentation is also available [1].

Attic requires Python v3.2 or greater and openSSL in a version greater than 1.0.0. Because the software also lets you mount a backup set in user space, the *llfuse* package from the Python treasure trove has to be installed to provide this function.

After a successful installation, you first have to initialize a backup repository. This is achieved with the command:

```
attic init /<Repository-Path>/
  <Repository-Name>.attic
```

Several directories can then be backed up in an archive (which should be specially created) in this repository. Attic does not enable encryption by default. The names for these archives can be freely selected. The following command backs up the directories:

```
attic create /<Repository-Path>/
  <Repository-Name>.attic::
  <Archive-Name>/
  <Source-Directory-1>/
  <[...]>
  <Source-Directory-n>
```

If the data must be encrypted, then the

```
--encryption=passphrase|keyfile
```

parameter command must be added.

I recommend using the weekday as the archive name for regular backups of the same directories, which quickly gives you the correct sequence of backups during a restore. The first backup in a repository can take a long time to complete for large data volumes, but subsequent backups will be far quicker because Attic saves them incrementally (i.e., only modified or newly added data is included in the backup).

If you want to monitor the backup run, you can display the most important data for the backed up archive using the `--stats` parameter. Attic not only lists the directories and the required time for the backup run, but also the number of files backed up and the volume of data. It shows both the original and the compressed backed up data volumes, so you can keep track of data compression efficiency (Figure 1).

In contrast to many other backup tools, Attic provides a convenient approach to listing archive content. For this purpose, enter the following command at the prompt:

```
attic list -v /<Repository-Path>/
  <Repository-Name>.attic::<Archive-Name>
```

The software then lists all the content, including file size, owner, and file permissions. Subdirectories are automatically included, and it shows the absolute paths.

Verifying the Archive

In the same easy way, you can check data integrity with the Attic backup program. The command

```
attic check /<Repository-Path>/
  <Repository-Name>.attic
```

checks the repository and all its archives. The status is then output as a short message (Figure 2). If inconsistencies appear, you can perform repairs by repeating the command with the added `--repair` parameter.

Restore

To restore an archive, use the `extract` option by entering

Table 1: Command-Line Backup Tools

	Attic	bup	Duplicity	rdiff-backup	rsnapshot
Local backup	Yes	Yes	Yes	Yes	Yes
Backup via SSH	Yes	Yes	Yes	Yes	Yes
Verification	Yes	Yes	Yes	Yes	Yes (logfile)
Encryption	Yes	Yes	Yes	No	No
Cloud services	No	No	Yes (Amazon, Rackspace)	No	No
Include/exclude directory	Yes	Yes	Yes	Yes	Yes
Time-controlled	Yes*	Yes*	Yes*	Yes*	Yes*
Front ends available	No	Yes	Yes	No	Yes
Incremental backups	Yes	Yes	Yes	Yes	No
Differential backups	No	No	No	No	No
Manual full backup	Yes	Yes	Yes	Yes	Yes
FUSE-mount possible	Yes	Yes	No	Yes	No

*Backups scheduled with the cron daemon.

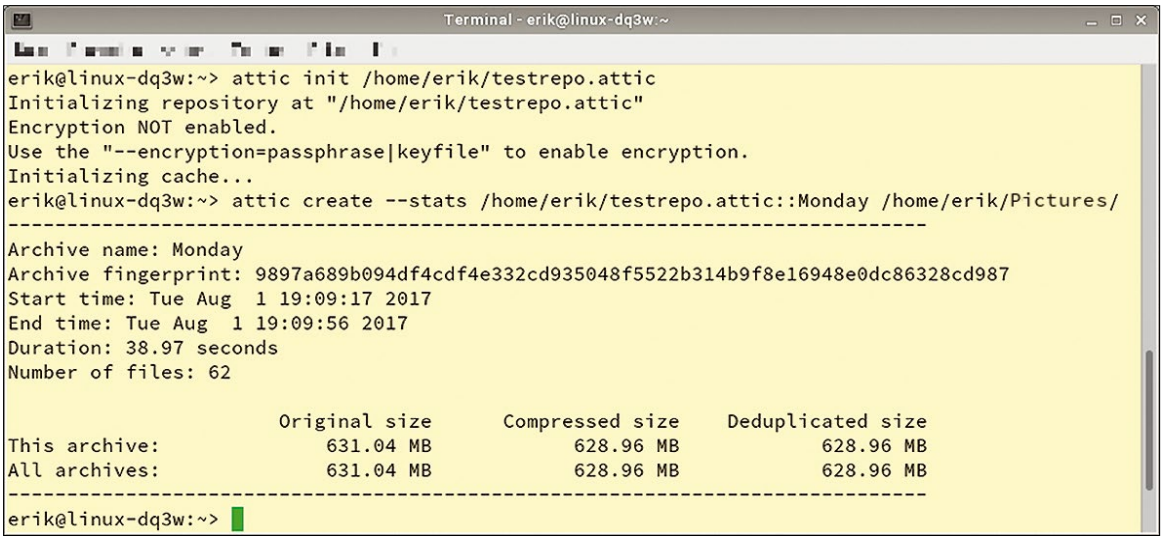

Figure 1: Attic provides clear-cut data for the latest backup.

```
attic extract /<Repository-Path>/?
  <Repository-Name>.attic::<Archive-Name>
```

which starts the recovery of the entire archive content to the original storage path. The software lists the individual files. The target path can be changed by an additional path specification, and you can exclude parts of the archive from the restore action.

Of course, Attic can also store backups on a remote server and retrieve them during a restore. It expects the same syntax as for a local backup; you address the server as follows:

```
<username>@<servername>:?
  <Repository-Name>.attic
```

However, enabling encryption is recommended when creating repositories on server systems. Attic uses 256-bit AES for encryption and HMAC-SHA256 for verification. Attic encrypts the data before storing it in the archive.

Automated

The software supports backup runs that are controlled and automated by cron jobs. To prevent the number of existing repositories from getting out of hand in the long run, Attic provides the `prune` parameter to let you define the storage life of older repositories. This option determines a maximum number of archives to be kept in the repository. You can define whether these are archives created hourly, daily, weekly, or monthly. However, they first must have been generated with the `date` parameter.

bup

The bup (short for backup) program has been in development and maintained for seven years, and it has established itself in small IT environments owing to its speed and efficiency [2]. The software is available from the repositories for immediate installation on many major Linux distributions.

Bup already differs from other backup tools in that it uses the Git packfile format instead of traditional TAR or ZIP archives. The software is very flexible in terms of handling: For example, bup archives can be mounted as filesystems in user space. Even backing up entire ISO images, which often contain several gigabytes of data, is no problem for the software. However, using the Git packfile format does require setting up a repository first.

For an overview of the software's numerous command parameters, zshort form. You can also find detailed documentation on the project site [3]. For local back-

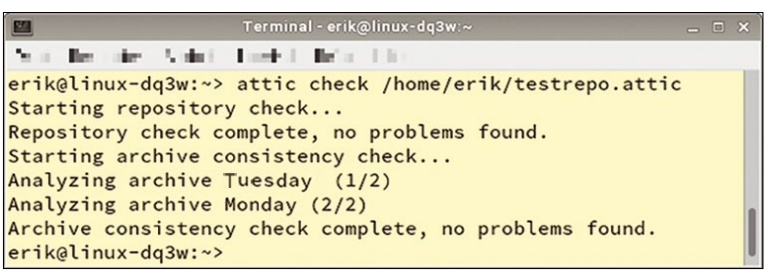

Figure 2: At a glance, you can see whether all the data was stored correctly.

ups on the desktop, the program has Gtk3 and Qt-based graphical front ends, which makes the backup process easier for inexperienced users. These are available from the bup homepage.

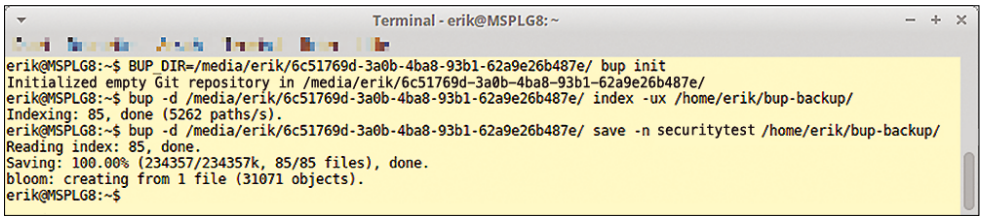

Figure 3: With bup, data is backed up in a few steps.

Usage

To create a backup, you first need to initialize the backup directory using the command sequence:

```
BUP_DIR=/<Backup-Set> bup init
```

Next, index the directory to be backed up with the command:

```
bup index -ux /<Directory>
```

If no corresponding directory variable is defined, the `-d` parameter must also be specified with the path for the backup set. Otherwise, bup saves the backup set in the hidden `.bup` subfolder in your home directory. After indexing, perform the backup with:

```
bup save -n <Set-Name> ➥
  /<Original-Directory>
```

In the event of missing directory variables, the path to the backup set must also be specified with a prefixed `-d` parameter. The set name can be optionally defined and is used to identify the correct set for recovery if several backups exist on the same target media.

During the first backup run, bup creates a full backup, which can take some time depending on the original data volume. The runs that follow only create incremental backups, which is much faster.

Keep in mind that before an additional backup run you have to re-index after each change to the original content for the software to detect changes.

If you want to store the backup on a remote server, in the simplest of cases, you would enter the following command sequence:

```
bup save -r <Username>@<Server-IP> ➥
  <Directory-Name> -n <Set-Name> ➥
  /<Original-Directory>
```

In this case, a full backup is run for the first pass, and the subsequent backups are incremental (Figure 3).

Bup also supports backing up a remote machine to a local machine. To do this, enter the `bup on` command followed by the server address and the target directory on the local system. Also, an indexing run must be performed beforehand.

To verify existing backup sets, use the `bup ls` command to list the individual files. The columns give you a clear overview (Figure 4).

Reverse Gear

Restoring backups is just as easy for users: After specifying the backup set path, use the `restore` parameter instead of `save`, followed by the set name and the path specification. Another option here is to restore individual subdirectories from the set.

Extras

Bup provides some little extras to check the integrity of backup sets. One of the most important goodies in practice is probably the option to mount a backup in user space like a conventional drive. The *python-fuse* package must be installed, and a mount directory must be created. The set is then integrated into the existing system by using:

```
bup -d <Path_to_Backupset> ➥
  fuse <Target-Directory>
```

You can then proceed to use the target directory and its content like any conventional drive.

Duplicity

Duplicity [4], which is available as a binary package for almost all common Linux distributions, is very flexible in terms of supported storage locations. Not only can you save backups locally, but also on FTP or SSH servers in the intranet. Additionally, Duplicity

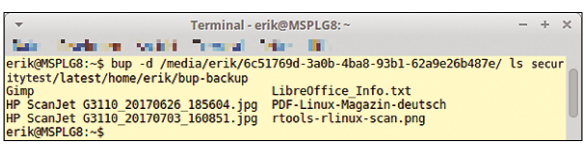

Figure 4: A simple display of the backed up data makes it easier for admins to check the backup.

supports Windows shares and WebDAV storage.

If such central storage options are not available, Duplicity can store the backups in the cloud. For this option, it supports Amazon's S3 cloud and Rackspace cloud solutions. One of the unique selling points of Duplicity is encryption: All files can be encrypted with GnuPG (GPG) and stored safely in a cloud, without curious outsiders being able to view them.

The software takes a very professional approach to creating backups: The first pass is a full backup, which it bundles into a TAR archive. The program then uses incremental backups, which saves not only storage space, but also time – in particular for larger databases. Signatures ensure data integrity, even in insecure environments.

That said, Duplicity does require a fair amount of user training because of its very complicated parameters [5]. Alternatively, graphical front ends (e.g., Déjà Dup [6]) that operate more intuitively are recommended, especially in smaller environments or for desktop backups, even though they fall well short of reproducing Duplicity's full functionality. The software does not support differential backups.

Duplicity is suitable for backing up individual folders, but not for creating complete system images. (Programs like Clonezilla [7] are better suited to this purpose.) In doing so, the program stores the backed-up data in volumes. For local backups, you also need to specify the absolute path, so the data is stored in the correct target folder.

Key Service

To let Duplicity play to its strengths (encrypted backups), you first have to generate a GPG key or already be in possession of one. Entering `gpg --gen-key` at the prompt generates a new key in just a few steps, and you can also define the key strength (Figure 5). The passphrase you specify when generating a key also provides increased security: If the key is lost, third parties cannot simply decrypt the archive because the passphrase is requested when restoring the database. Particularly paranoid users can also digitally sign their archives so that the data integrity of the backup volumes can be checked at a later time.

In the simplest case, specify the following command for a local backup:

```
duplicity /<Source-Directory> ↵
  file://<Target-Directory>
```

After prompting you for the passphrase twice, the software creates the backup in the target directory. To save data on an FTP server, call the software using the command:

```
duplicity /<Source-Directory> ↵
  ftp://<backupuser>@<Host.Name>/↵
  <Target-Directory>
```

The passphrase is also requested here before saving; you can work around the prompt by prefixing the sequence `FTP_PASSWORD=<Password>` to the backup command. In both cases, Duplicity automatically checks to see whether a full backup already exists in the target path. If this is not the case, it automatically creates the full backup during the first pass. With the second pass, it creates an incremental backup (Figure 6).

Manual Trigger

An increasing number of incremental backups accumulates over time, especially if you automate the backup runs. Because these need to be restored in sequence from the last full backup, it is a very good idea to create a new full backup regularly to minimize the number of incrementally backed up archives. Duplicity launches into a full manual backup when started with the `full` parameter. The `incremental` parameter manually triggers an incremental backup.

If specific directories are to be excluded from the backup, you can specify these using the `--exclude` parameter. Several directories that

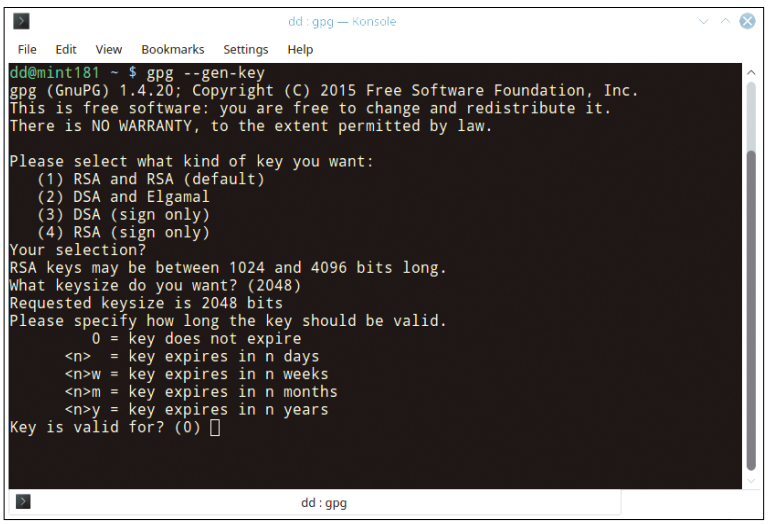

Figure 5: A key can be created in just a few minutes.

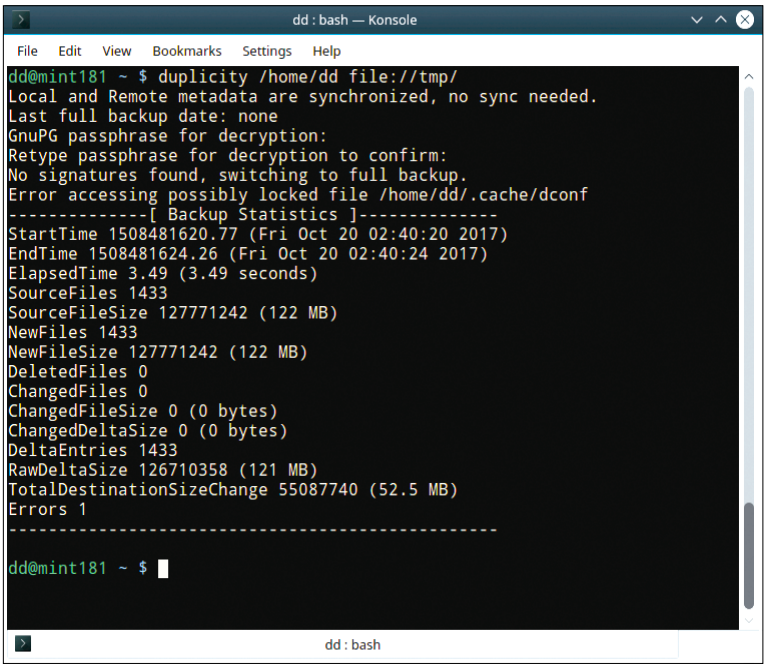

Figure 6: Duplicity also provides information about the backup.

you do not want backed up can be added in a space-separated list. Excluding individual directories is useful, especially for subfolders that contain, for example, temporary, cache, or logfiles. However, caution is required when backing up the root directory: The `/proc` directory must be excluded; otherwise, Duplicity will freeze. Conversely, the `--include` parameter can include additional folders in the backup.

Restore

Data recovery is just as easy: Duplicity is activated without any additional parameters; only the source and target paths need to be swapped. Alternatively, you can also specify a different target path. The archive is only unpacked after entering the passphrase, so unauthorized persons cannot access the data without knowing the password.
Using the `verify` and `--compare-data` parameters, you can test the integrity of your backups in a simple way. This test is possible even if the volumes are in a cloud. Particularly for documentation purposes, it is recommended to use the additional `-v<x>` parameter (verbosity level), and to replace the *x* placeholder with one of the numbers 4, 8, or 9 to obtain meaningful information. The data can then be saved in the specified file using the `--log-file <file name>` parameter.

rdiff-backup

Rdiff-backup [8] is another popular tool for backing up at the command line. The program is maintained in the repositories of virtually all major Linux distributions. The software is extremely easy to use: To produce a local backup of a directory or directory tree at the prompt, type:

```
rdiff-backup <Source-Directory> ➥
             <Target-Directory>
```

This creates a full backup that the administrator can recover without special tools just as easily as using a conventional folder hierarchy, using only on-board Linux tools. Subsequently backups made by rdiff-backup contain deleted or older versions of the changed files. The latest versions of the changed files all end up in the full backup, just like new files, so you always have the current status of the backup. Regular backups only save older versions of the database. The advantage of this special form of incremental backup, known as reverse delta, is that you do not have to fight your way through several incremental backups to restore a complete database.

Servers

Using rdiff-backup, you can also back up servers in the intranet. Note that you do need to install the application on the server. To run a backup over SSH, at the remote client prompt enter:

```
rdiff-backup <Username>@<Server-IP::➥
  <Source-Directory> <Backup-Directory>
```

This command backs up the source directory from the server on the local client. If the backup also includes files to which only root has access, you will need to log in with the appropriate privileges.

Displays

Rdiff-backup, like all other tools being reviewed here, has a collection of parameters for managing backups. To view the content of a backup, enter

```
rdiff-backup -l <Backup-Name>
```

at the prompt.

Because rdiff-backup does not output status messages during the backup runs, it also makes sense to enter the `-v6` parameter to switch to the highest verbosity level. The software then outputs all the processed files. Additionally, after launching the backup, various status messages appear about enabled and disabled parameters (Figure 7).
The `--compare` and `--list-increments` parameters provide information about the modified files and the different backup points. Using the command

```
rdiff-backup-statistics ➥
  <Backup Directory>
```

you can also retrieve statistical data for the backup directory (Figure 8).

Exceptions

As with other backup programs, you can also exclude files or directories from the backup. This can be beneficial, for example, if the backup path includes directories for temporary files that you don't want to save. In such cases, use the `--exclude` parameter to exclude individual files.
If you need to exclude complete directories from the backup run, you can use the `--exclude-file-list` parameter; you need to specify a file containing the paths of the files to be excluded. The file has to be created manually.

Restore

To restore backed-up files from the archives, you don't need to invoke rdiff-backup again: If you need to restore your data collection from the current archive, you can simply copy the data from the backup directory. The Linux `cp` command helps you do this; you need to specify the archive option. The easiest form is:

```
cp -a /<Backup-Directory>/<File-Name> ➥
  /<Restore-Directory>/<File-Name>
```

To access old backup files, the software offers two options: You can either access the incremental files from the `rdiff-backup` program, or you can use `rdiff-backup-fs`, which must be installed separately as an external package, to mount the backup archive as a conventional filesystem.
The *`rdiff-backup-fs`* [9] package (the name can differ among

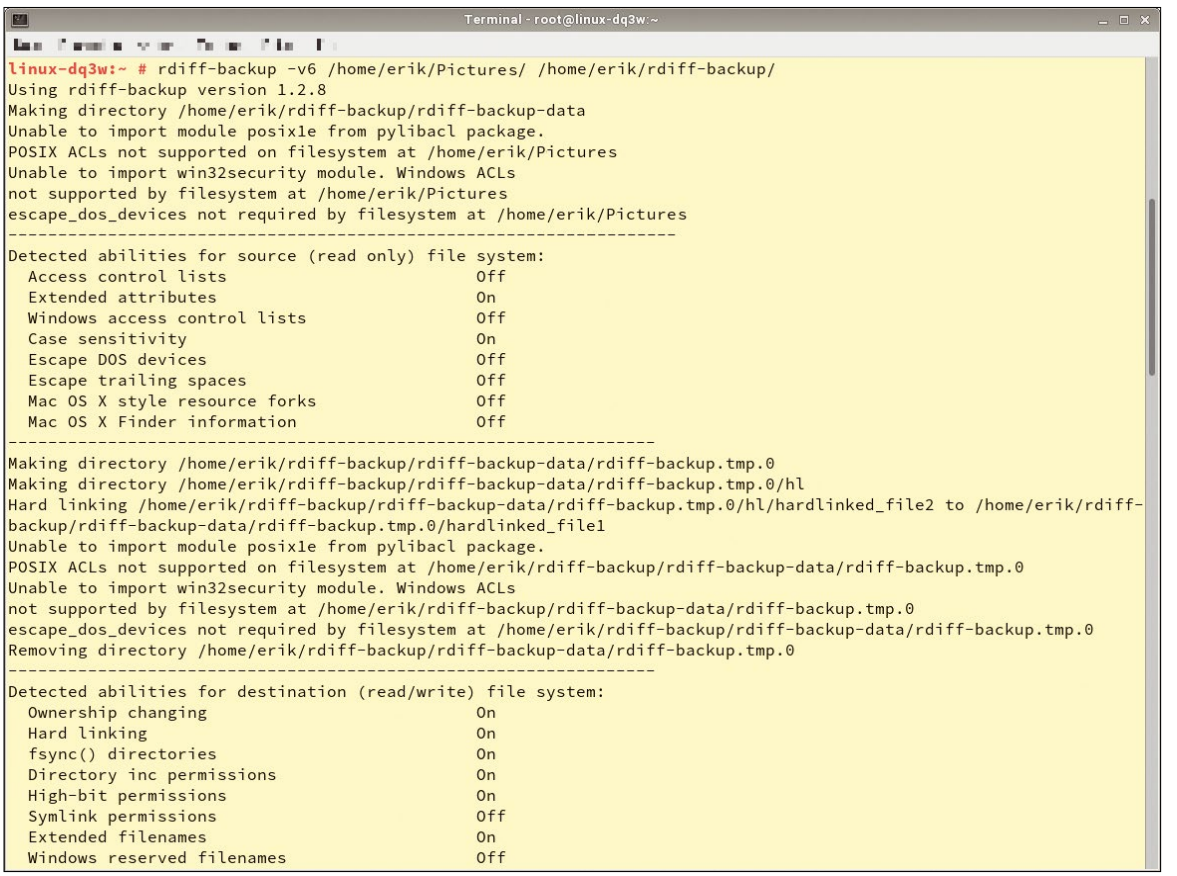

Figure 7: rdiff-backup gives you far more information than other backup programs – if you tell it to do so.

distros) is maintained in some software repositories of major Linux distributions, but it can also be picked up from the project website. For direct access, use the `-r` parameter (short for "restore as-of") followed by a date. For example, you need the following command sequence to restore a 10-day-old backup from a server to a local directory on a client:

```
rdiff-backup -r 10D ➥
  <Server-Name>::/<Source-Directory>/➥
  <Restore-Directory>
```

The detailed documentation [10] lists various application scenarios that makes the somewhat unusual nomenclature understandable.

rsnapshot

As the name suggests, rsnapshot is a tool for creating complete snapshots of a filesystem [11]. It can create both local snapshots and use SSH to create snapshots of remote systems. The Rsync tool creates backups in which hard links replace unchanged files. Rsnapshot actually only writes modified data in the backup. The cron daemon regularly initiates the backup. Manual backup runs are not intended. The software takes all its information from a configuration file.

Because rsnapshot works with hard links, the backups always have to end up on the same filesystem. Otherwise, the tool would have to create a space-consuming full backup. Therefore, rsnapshot is more suited to servers on small networks or to local workstations than to extensive storage setups. However, the tool, which is written in Perl, only stores a configurable number of backups, so the storage space remains manageable, even with short backup intervals.

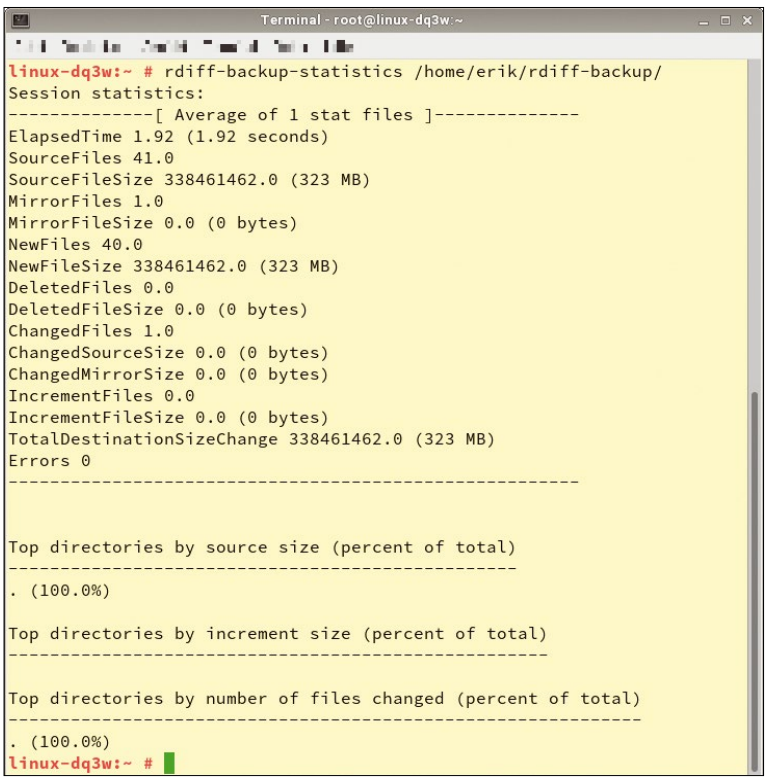

Figure 8: One of the various statistical data displays makes it easier for you to check the backups.

Configuration

After the installation, you need to configure the program, which is available from the repositories of practically all major Linux distributions. To do this, call the `/etc/rsnapshot.conf` file, which can be edited easily with any standard text editor.

Because the numerous options are easily accessible, thanks to detailed comments, the configuration does not pose unsolvable problems, even for inexperienced users. Note that the different options need to be separated by tabs. Also, path specifications must always end with a slash.

The configuration file not only lets you specify the source and target directories and list included or excluded files, it also contains a schedule for a backup plan. Moreover, settings can be defined to back up a remote server via SSH or handle an LVM network. Settings for the snapshot retention period (Figure 9) are also important.

The configuration file is quite extensive, so you can validate it after a modification. To do this, enter the `rsnapshot configtest` command at the prompt with administrative privileges. The *Syntax OK* output signals a consistent configuration. A cron or anacron job then launches the software automatically. A predefined sample configuration is available in the `/etc/cron.d/rsnapshot` file on some distributions, and its data can be adapted to suit your individual needs. A FAQ [12] provides more information.

Rotation Keeps Data Under Control

The rotation principle helps you keep the number of backups manageable in line with your needs. You can use the configuration file to define intervals at which backups are created and specify how many backups must be retained for each interval. The `retain` parameter in the `/etc/rsnapshot.conf` file uses time parameters such as `daily` or `hourly` followed by a number to configure both.

For example, `daily retain 5` means that a daily backup launches, and the last five daily backups should be kept. In further lines, additional intervals can be defined in the same way, each with its own number of backups to be retained, making rsnapshot very flexible in terms of use.

After completing the configuration, launch a first test run by entering the following command:

```
rsnapshot -v <Interval>
```

Use the `Interval` parameter to specify the defined interval in the configuration file (i.e., `hourly` or `daily`). You do not have to specify source and target directories, because rsnapshot takes this information from the configuration file.

Because rsnapshot does not store its backups in archives or its own formats, the data is directly accessible and can be easily copied back for recovery. After a successful test run, you can set up the required cron jobs.

Conclusions

The backup solutions explored here are all reliable and stable. They are suitable for backing up both local systems and servers. However, if you want to outsource your backup archives into the cloud, most programs will not be appropriate because of their lack of encryption support. In a public cloud, you will not want to store your backups without encryption. Additionally, some candidates have poor to simply catastrophic documentation. In part, the existing description does not list function parameters or sample applications, and the man pages are terse 10-liners. Consequently, developers should not expect their programs to be widely accepted by users whose time and patience are limited.

These backup solutions, most of which are developed for Unix-style operating systems, also exhibit some design weaknesses, in that they may not always work in heterogeneous environments. For example, hard links cannot be used on all platforms or with all filesystems. For a tool like rsnapshot, this factor limits the applications to Linux and related systems. ■

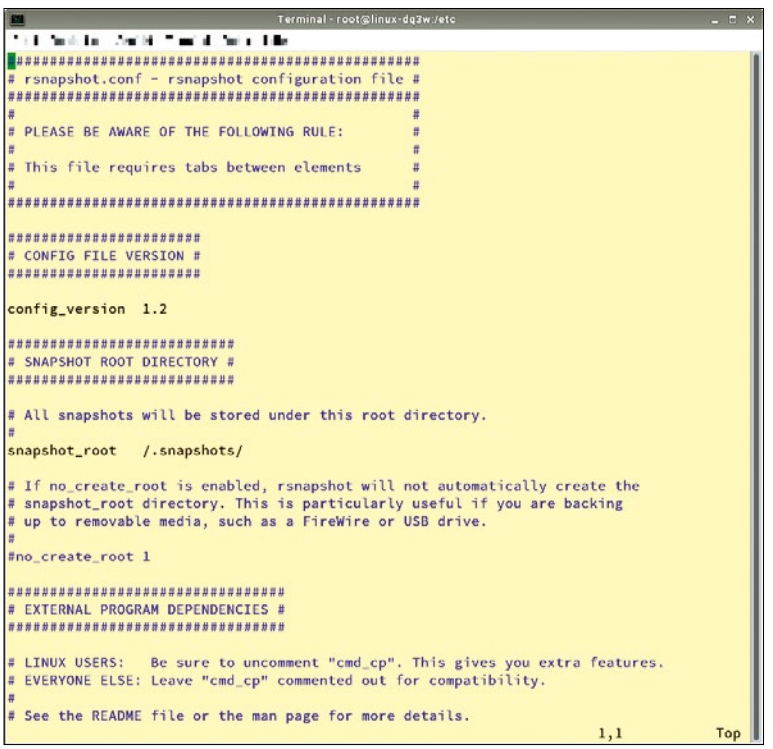

Figure 9: rsnapshot is set up with a configuration file.

Info

[1] Attic: https://attic-backup.org/installation.html#installation

[2] bup on GitHub: https://github.com/bup/bup

[3] bup documentation: https://bup.github.io/man.html

[4] Duplicity: http://duplicity.nongnu.org

[5] Duplicity documentation: http://duplicity.nongnu.org/duplicity.1.html

[6] Déjà Dup: https://launchpad.net/deja-dup

[7] Clonezilla: http://clonezilla.org

[8] rdiff-backup: http://www.nongnu.org/rdiff-backup/

[9] rdiff-backup-fs: https://github.com/rbrito/rdiff-backup-fs

[10] rdiff-backup documentation: http://www.nongnu.org/rdiff-backup/examples.html

[11] rsnapshot: http://rsnapshot.org

[12] rsnapshot FAQ: http://rsnapshot.org/faq.html