Uptimes

Mitgliederzeitschrift der
German Unix User Group

RANCID nicht nur für Cisco

Weihnachtliche Gadget-Wünsche

Uptimes: vom LaTeX zum EPUB

guug 2012-3

Inhaltsverzeichnis

Liebe Mitglieder, liebe Leser!
Gruß vom Vorstand

Die Winterausgabe gibt es frisch zu Weihnachten, ein EPUB ist am Start, das Frühjahrsfachgespräch 2013 profitiert von besonders vielen guten Beiträgen, und das Budget 2013 entsteht.

von **Wolfgang Stief, Vorsitzender des Vorstands**

Hello, merry GUUG
goodbye heart
sweet merry GUUG, I'm so in charge of you

Frei nach: Hello Mary Lou, ©Gene Pitney/Cayet Mangiaracina, 1960.

Das Jahr 2012 neigt sich dem Ende zu. Für die Aktiven in der GUUG kein Grund, sich auszuruhen. In den letzten Wochen wurde ein weiteres Mal mit Hochdruck an einer neuen Ausgabe der Uptimes gearbeitet, so dass die nächste zumindest rechtzeitig zu den Feiertagen bei Euch ankommt. Zum letzten Heft – der ersten PDF-Ausgabe – gab es an die Redaktion nur positive Rückmeldungen. Herzlichen Dank dafür, das spornt uns natürlich an. Auch mit dieser Uptimes gibt es eine Neuerung: Dank dem Einsatz vor allem von Mathias Weidner, aber auch Robin Schröder, einigen Betatestern und Anika Kehrer können wir jetzt aus den LaTeX-Sourcen quasi zeitgleich mit dem PDF auch eine EPUB-Ausgabe erstellen. Details könnt Ihr in einem Artikel von Mathias in dieser Ausgabe lesen.

Eine Vereinszeitschrift herauszugeben in Zeiten von Online-Medien, Blogs und Social Networks mag dem einen oder anderen altertümlich vorkommen. Auf einer GUUG-nahen IT-Veranstaltung im Herbst dieses Jahres sagte mir aber ein langjähriges Mitglied sinngemäß: Das Schöne an der GUUG sei, dass wir auch alte Unix-Werte und -Traditionen hoch halten. Für mich Ansporn genug, auch an einer Vereinszeitung festzuhalten. :-)

Neben der Uptimes gilt es, das nächste Frühjahrsfachgespräch zu organisieren. Das Programm steht, die Anmeldung ist ab sofort möglich. Weil das Programmkomitee dieses Jahr beachtliche über 100 Einreichungen hatte und davon selbst nach kritischer Sichtung noch ziemlich viele hochklassige übrig blieben, wird es in 2013 das erste Mal am ersten Konferenztag (Donnerstag) einen dritten Track geben. Der Vorstand, Die FFG-Orga und nicht zuletzt das Programmkomitee wünschen sich natürlich, das auch in 2014 fortzusetzen. Das gesamte Programm des FFG 2013 findet ihr in dieser Ausgabe. Anmeldungen bis Ende Januar kommen zusätzlich zum Mitgliederrabatt auch noch in den Genuss des Frühbucherrabatts. Details stehen auf den Webseiten zum FFG: http://www.guug.de/ffg.

Auf seiner letzten Sitzung Anfang Dezember 2012 in Berlin hat sich der Vorstand unter anderem mit dem Budget für das kommende Jahr beschäftigt. Auch in der nächsten Vorstandssitzung – Anfang Februar 2013 in Bochum – wird das Budget 2013 ein zentraler Diskussionspunkt sein, zusammen mit der dann fertigen Gewinn- und Verlust-Rechnung für 2012. Das fertige Budget stellen wir der Mitgliederversammlung Ende Februar vor und hoffen natürlich auf wohlwollende Diskussion und Genehmigung. Nach den positiven Erfahrungen bei der Unterstützung von Veranstaltungen rund um Open Source werden solche Veranstaltungen nächstes Jahr wieder großen Raum einnehmen. Ebenso wollen wir die GUUG in der Fachwelt bekannter machen, was sicher auch den einen oder anderen Euro an Investition braucht.

Genug der Vorrede! Ich wünsche euch viel Spaß mit der Lektüre dieses Heftes. Und natürlich wünsche ich allen Mitgliedern und Lesern sowie Euren Familien ein ruhiges, gesegnetes Weihnachtsfest und einen erfolgreichen Start in das Jahr 2013.

Über Wolfgang

Wolfgang Stief hat Linux während des E-Technik-Studiums kennen und schätzen gelernt (Kernel 0.99). Seit 1998 verdient er sein Geld überwiegend mit Solaris, Storage-Systemen und verteilten Dateisystemen. Er ist seit dem 30.12.2001 GUUG-Mitglied, seit 2005 im Vorstand und seit 2008 Vorsitzender. Wenn er gerade mal nicht beruflich oder für die GUUG unterwegs ist, züchtet er Gemüse, treibt sich in sozialen Netzwerken herum oder macht Musik.

Moin Moin
Lokalgruppe mit Hafenblick

In mehreren Städten treffen sich GUUG-Mitglieder und andere Unix-Interessierte regelmäßig in Person. Dafür muss einer die Organisation übernehmen.

In Hamburg macht das **Dirk Wetter**

Ick bin wat ik bün, kumm mi nich anne Plünn.
Doch komm fixma rum, um die de Norden antokieken.
Bi uns dor is jümmer wat los achter de Dieken.
Set di eerstmal dal, nimm 'n Kööm un 'n Aal
un smeckt di dat nich, is mi dat ok schietegal.

©Fettes Brot, Nordisch by Nature, 1995.

Wie kam es anno 2006 zur Hamburger Regionalgruppe?
Dirk: Ich habe mich 2005 beim seligen Linux-Kongress mit dem GUUG-Virus infiziert. Martin und Wolfgang waren bei der Gelegenheit in Hamburg. Ich kann mich erinnern, dass Letzterer erzählt hat, er hätte in München etwas Ähnliches angeschoben – also eine lokale Gruppe – und das habe Spaß gemacht. Martin hatte mich auf seine wohlwollende Art ebenfalls ermuntert.
Beschreib doch mal, wie die Hamburger so sind.
Dirk: Das sind Pappbrötchen mit 'ner Fleischfrikadelle dazwischen. :-) Na gut, im Ernst: Wer „die Hamburger"sind, das ist vom Thema und von Lust und Laune der Beteiligten abhängig. Bei Netzthemen sehe ich ganz andere Leute als bei Sicherheitsthemen. Heute ist IT auch so diversifiziert, dass Netzthema nicht unbedingt gleich Netzthema ist und Sicherheit nicht gleich Sicherheit.

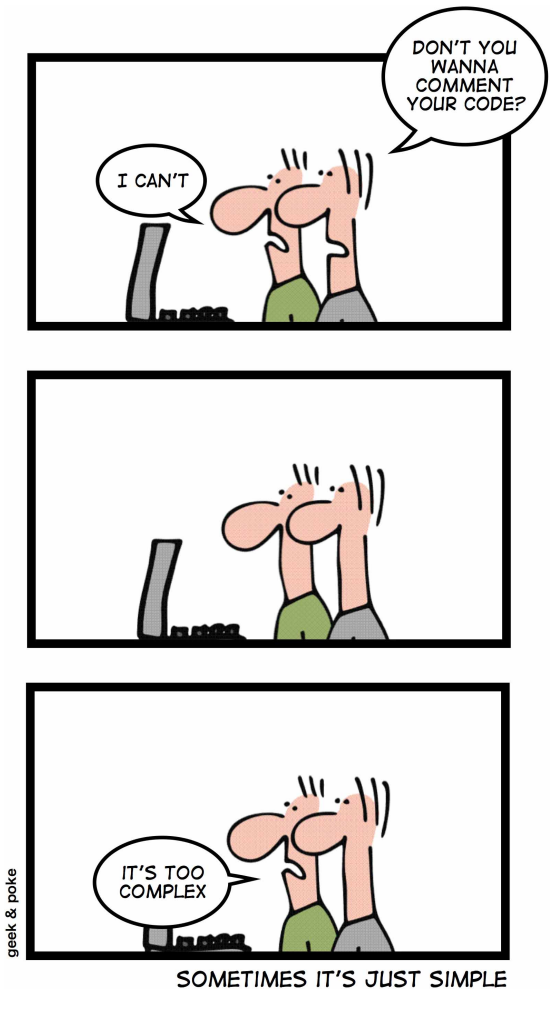
Beim Netzvortrag über das Border Gateway Protocol von Thorsten kamen Leute, die ich erwartete, aber überraschenderweise auch andere, deren Vergangenheit ich bis dato überhaupt nicht kannte – wie Lars MB, der mal für einen Internetzugangsprovider gearbeitet hat. Der Vortrag eines Sprechers, dessen Arbeitgeber Netzwerk-Equipment herstellt und der es geschafft hat, nicht ein einziges Mal seinen Namen zu nennen – ich tu dies hier nun auch nicht – behandelte Hardcore-Netzthemen. Da waren Leute dort, die ich nie zuvor gesehen hatte. Das kam vielleicht auch daher, dass Andreas LQ Kunden aus dem öffentlichen Bereich betreut und diese im Schlepptau hatte. Bei Dondes Forensik-Vortrag wiederum war gefühlt das halbe DFN-Cert anwesend, beim ISO/Verinice-Vortrag von Alex K waren's eher so die gesetzteren Herren wie ich, die die gewisse Reife für Informationssicherheit haben.

Kurz: Es gibt ein paar Treue, aber sonst sind die Hamburger recht promiskuitiv. ;-) Wer hätte das gedacht...
Wo in Hamburg trefft Ihr Euch, und warum gerade dort?
Dirk: Wir treffen uns in der Uni, weil Olo dort den Raum stellt. Das Bier danach gibt's meistens in dem auf türkisch gemachten Etablissement, wo wir unser Speaker Dinner während des Linux-Kongresses 2005 und 2008 hatten. Wenn wir weniger als zehn Leute sind, die noch Hunger und Durst mitbringen, gehen wir ins *Doris Diner*. Kleiner Schwank am Rande: Das ist mittlerweile fest in indischer Hand, aber den beliebten *Porn-Burger* gibt's immer noch. Erklärung: Beim Font der Karte ist das kleine *n* verdammt nah am *k*.
Sind Eure Treffen eher Stammtische, oder gibt es bei Euch auch Vorträge oder Workshops?
Dirk: Es gibt fast immer Vorträge. Vielleicht sind in anderen Städten die Leute genügsamer oder trinkfreudiger, aber mit nur einem Astra, Jever oder Becks kann ich hier kaum jemand hinterm Computer hervorlocken. Workshops, wo jeder Teilnehmer Hand anlegt, sind eher selten. Der gute Wolfgang L von der Ex-Firma S, nun O, hatte mal was zum Thema SMF arrangiert. Wenn ich mich richtig erinnere, hatte Olo noch seine Sun E4500 dafür geopfert. Spontan erinnere ich mich auch noch an Teilnehmer Donde, der zwar allein vorgetragen hat, aber das ging tierisch lange – das hatte eher Seminarcharakter.
Habt Ihr Lieblingsthemen oder Themen, die häufiger vorkommen?
Dirk: Nö. Ich nehme, was kommt.
Wieviele Teilnehmer kommen in der Regel zum Hamburger Treffen?
Dirk: Zwischen sieben und 30. Sieben heißt: Ich habe mit der Ankündigung etwas falsch gemacht. 30 ist aber auch wieder eine Ausnahme.
Was hast Du als Organisator in der Regel zu tun?
Dirk: Ich sorge für Sprecher und mache die Ankündigungen. Olo ist unser fürsorglicher Host in der Uni. Dann haben wir noch Propaganda-

Material zur Auslage – vulgo GUUG-Flyer – und ich moderiere auch die Treffen. Das war's schon. Das schwierigste ist, Sprecher zu finden, auch für mich. Ich empfehle jedem, der eine Gruppe auf die Beine stellen möchte, dafür einen guten Plan zu haben. Damit steht und fällt der Erfolg.

Über Dirk

Dirk Wetter beschäftigt sich mit Security und Websecurity. Wenn einem so jemand in einem Unix-Ballungsgebiet begegnet, dann kann man davon ausgehen: Das ist Dirk.



Vereinsleben
Aus MV und Vorstand Mitte bis Ende 2012

Die GUUG ist ein lebendiger Verein mit einer jährlichen Konferenz und Mitgliederversammlung, mehreren Vorstandssitzungen pro Jahr sowie diversen Aktivitäten hinter den Kulissen. Dieser Artikel soll auch diejenigen anschließen, die – aus welchem Grund auch immer – weniger am Geschehen teilnehmen.

von **Anika Kehrer**

Die Organe des Vereins
sind die Mitgliederversammlung
und der Vorstand.

§ 8 GUUG-Satzung

Einen großen Raum in den Vorstandssitzungen beansprucht regelmäßig das FFG. Eine Menge Ideen enstehen, etwa, die Proceedings als PDF auf einem USB-Stick abzugeben – woran sich stets die Frage anschließt, wie das denn konkret aussehen soll. Und wer sich – als anderes Beispiel – Gedanken um eine digitale Aufzeichnung der Vorträge macht, der muss auch die Rechteübertragung der Aufgezeichneten einplanen. Diverse kleine organisatorische Aufgaben hangeln sich auch rund um die Besetzung des Programmkomittees (wen sprechen wir an?), die interne organisatorische Abstimmung (wer macht was?) und die Frage, welche Werbemaßnahmen sinnvoll sind (was kostet die Anzeige, was gibt es da noch obendrauf? Wollen wir T-Shirts machen?).

Mit dem Ausblick auf das FFG 2013 rückt das zentrale Organ der GUUG in den Mittelpunkt: die Mitgliederversammlung. Für diejenigen, die dieses Jahr in München nicht dabei sein konnten, hier einige Ergebnisse der MV 2012.

Mitgliederversammlung!

Zum Ende des Jahres 2011 hatte die GUUG 678 Mitglieder, eine Steigerung um 17 im Vergleich zum Vorjahr. An der Mitgliederversammlung 2012 haben 51 Mitglieder teilgenommen, 3 davon Frauen. Neben einer solchen Statusmeldung ist der Kassenbericht für das vergangene Jahr ein zentraler Punkt jeder MV. So hat der Verein 2011 nach derzeitigem Kenntnisstand einen Überschuss von gut 18.000 Euro erwirtschaftet und besaß damit per 31. Dezember 2011 ein Vermögen in Höhe von rund 77.000 Euro. Die Einnahme-Überschussrechnung erstellt übrigens der Steuerberater des Vereins. Doch unabhängig davon bilanziert der Schatzmeister ebenfalls, damit frühzeitig und vor allem rechtzeitig zur Mitgliederversammlung ein Überblick vorliegt. Kassenbericht und Finanzlage sind in einer gemeinschaftlichen Organisation stets ein von den Mitgliedern gründlich inspizierter Bereich, der sich durch mehrere Tagesordnungspunkte hindurchzieht.

Auch der Bericht des Vorstands ist regulärer Teil der MV. Im Jahr 2011 hatten technische Weiterentwicklungen die Vorstandsarbeit bestimmt, etwa der Umzug des Vereinsservers auf eine neue Hardware und die Umstellung auf IPv6. Auch hat der Verein ein neues Corporate Design entwickeln lassen, das in das zukünftige Projekt einer neuen Webpräsenz münden werde. Für das Jahr 2012 standen vor allem mehr Öffentlichwirksamkeit auf dem Programm, implementiert durch Veranstaltungssponsoring und Social-Network-Aktivitäten.

Mäzenatentum

Kurz bevor die letzte Uptimes erschienen ist, traf sich der Vorstand in Köln zur zweiten Vorstandssitzung des Jahres. Hier jonglierte er mit allerlei Geldbeträgen: Die Sponsoringpakete etwa der Programmierkonferenz PyCon von November 2012 bewegten sich zwischen 1000 (Bronze) und 6000 Euro (Platin). Der Betrag von 1000 Euro bildet auch die Sponsoren-Untergrenze bei der DENOG-Konferenz (German Network Operators Group). Die OpenRheinRuhr konnte nicht in den Genuss von GUUG-Geldern kommen, weil sie 2012 gar nicht stattfand, wie nach einigem Hin und Her festgestellt wurde. Und für die Froscon hat sich die GUUG ein eigenes Paket geschnürt.

Bislang hat sich der Vorstand zu folgenden Gaben entschlossen: Die Froscon 2012 erhielt Besuchertaschen, die bis zu 2000 Euro kosten durf-

ten. Und die DENOG erhielt den Bronze-Betrag. Aber erst, nachdem eine kurzentschlossene telefonische Abstimmung mit den Organisatoren noch während der Vorstandssitzung zu der individuellen Abmachung geführt hat, dass die GUUG Flyer auslegen und ihr Logo auf den Rollouts am Empfang platzieren darf. Außerdem verloste die GUUG zwei Tickets unter den Mitgliedern. Im Austausch dafür verzichtete sie auf den eigenen Vortragsslot. Zum Jahresende hat sich der Vorstand schließlich noch kurzfristig entschlossen, die Berliner EHSM (Exceptionally Hard & Soft Meeting) mit 3.000 Euro unterstützen, eine Konferenz, die sich ausschließlich mit Hardware-Hacks aus allen möglichen Bereichen beschäftigt. Erst vor wenigen Tagen wurden auch hier drei Eintrittskarten unter den Mitgliedern verlost.

Hingegen fiel die Entscheidung, die Python-Konferenzen PyCon nicht zu unterstützen. Hier stellt sich der Vorstand im Namen der GUUG die Frage, wie weit in benachbarte Themen die Unterstützung denn eigentlich reichen solle – der Programmiersprachen zum Beispiel ist ja kein Ende. Die Diskussion der diversen Veranstaltungen führte auch zu der selbstkritischen Frage, was generell mit dem Sponsoring erreicht werden solle. Man müsse aufpassen, war man sich einig, dass man sich nicht in anderer Leute Veranstaltungen verzettele, denn Sinn der GUUG sei schließlich nicht die Rolle einer „deutschen Open-Source-Stiftung“. Einig war sich der Vorstand über die Frage, dass auf jeden Fall kleine, neue Veranstaltungen das Geld wirklich brauchen können.

Von solchen Incubator-Überlegungen auf einer Vorstandssitzung bis hin zur Präsentation auf der Mitgliederversammlung führt ein weiter Weg, der manchmal auch an der Bitte um Input via Newsletter an die Mitglieder entlangführen wird. Hingegen im Hier und Jetzt den Plan für GUUG-Aktivitäten im kommenden Jahr 2013 vorzuschlagen, ist Aufgabe des Vorstandes auf der nächsten Mitgliederversammlung – am 26. Februar 2013 auf dem FFG in Frankfurt.

Über Anika

Anika Kehrer stöbert als freie Journalistin in den Facetten der Informationstechnik. Sie ist seit der Wiederauflage der UpTimes im Sommer 2012 als stolze Chefredakteurin an Bord und wünscht sich vom Weihnachtsmann, dass im nächsten Jahr aus dem Kreis der GUUG-Mitglieder und Nahestehender genauso viele tolle Artikelideen in der Redaktion unter <redaktion@uptimes.de> eingehen – hint, hint –, wie es bei den weihnachtlichen Gadget-Wünschen weiter hinten im Heft der Fall war. :-)

Die UpTimes auf jeden E-Book-Reader! Von LaTeX zum EPUB

In der letzten UpTimes fragte Wolfgang nach jemandem, der sich in E-Book-Formaten zu Hause fühlt. Ich fühlte mich berufen, und habe mich um die erste UpTimes im EPUB-Format gekümmert. Ein Erfahrungsbericht.

von **Mathias Weidner**

Als in Ordnungen nun,
wer jener auch war von den Göttern,
abgeschichtet den Wust,
und die einzelnen Schichten gegliedert;
formt' er EPUB im Beginn,
und schuf, dass nirgend ihm ungleich
wär' ein Teil, die Gestalt des großartigen PDF.

Frei nach Publius Ovidius Naso, Metamorphosen

Ich habe bereits einige Bücher als EPUB herausgebracht. Also kam ich auf die Idee, ich könnte gemeint sein, als jemand nach EPUB-Erfahrungen fragte. Zwar hatte ich bisher das PDF für den Druck und das HTML für die EPUBs mit den Python DocUtils [1] erzeugt, während die Uptimes direkt in LaTeX gesetzt wird – aber zumindest wusste ich bereits, wie so ein EPUB aussehen muss und wie man formal die Qualität kontrolliert. Also nahm ich Kontakt zur Redaktion auf, bot meine Dienste an und erhielt Zugang zum Repository mit den LaTeX-Quellen der UpTimes. Die Reise konnte beginnen.

Das EPUB-Format

E-Books im EPUB-Format bestehen im wesentlichen aus XHTML-Dateien für den Text, aus Bilddateien und aus CSS-Dateien für das Layout. Zusammen mit ein paar Metadaten-Dateien liegen sie in einem Zip-Archiv. Einen schnellen Überblick über den Aufbau bekommt man mit dem *EPUB Format Construction Guide* von Harrison Ainsworth [2]. Noch detaillierter ist es in der Spezifikation [3] beschrieben.

Von Google gibt es das Programm *epubcheck* [4], womit auch Verlage prüfen, um eine vorliegende EPUB-Datei dem EPUB-Format entspricht. Das verwende ich für die Qualitätssicherung. Mit diesem Wissen ausgerüstet, machte ich mich an die Arbeit und teilte diese als erstes in drei Phasen ein:

1. Konvertieren der LaTeX-Artikel nach XHTML;
2. Erzeugen der Metadaten und Zusammenbau des E-Books;
3. Erarbeiten eines Stylesheets für das Aussehen.

HTML aus LaTeX

Ich hatte bereits erwähnt, dass ich bisher meistens mit den *Python Docutils* gearbeitet hatte. Mit diesen hatte ich aus dem Markup *reStructured Text* einerseits LaTeX für PDF und Druck, und andererseits XHTML für EPUB erzeugt.

Diesen Weg konnte ich hier nicht gehen, weil die Uptimes 2012-02 bereits vorlag, und das EPUB natürlich den gleichen Text und die gleichen Bilder enthalten sollte. Der Weg, den ich diesmal gehen wollte, musste also von LaTeX zu XHTML führen. Also suchte ich nach geeigneten Programmen für diese Konvertierung. Fünf Programme fand ich schließlich, die mir geeignet schienen und die ich mir mehr oder weniger intensiv daraufhin anschaute, inwiefern ich sie verwenden konnte.

Die aktuelle Version von ***LaTeX2HTML*** ist von 2008, also nicht mehr ganz frisch. Aber wenn es funktioniert, ist das kein Makel. Außerdem gibt es Latex2html als Softwarepaket für Ubuntu 12.04 – meine Entwicklungsumgebung. Somit war es schnell installiert, und ich wollte ja kein Yak rasieren, sondern nur ein paar LaTeX-Quellen nach HTML konvertieren. Leider erwies sich dieses Tool aber als etwas zu sperrig für die Uptimes (oder für meine LaTeX-Kenntnisse). Da ich die existierenden Quellen nach Möglichkeit nicht ändern wollte, probierte ich erstmal die anderen Kandidaten.

Der nächste war ***plasTeX***, in Python geschrieben, ziemlich komplex, aber mit bestechendem Konzept: Es generiert aus der Quelle einen Syntaxbaum, in den man seine eigene Ausgabefunktionen einhängen kann. Damit sollte es möglich sein, selbst die windigsten LaTeX-Makros abzubilden und daraus geeignetes XHTML zu generieren. Mit dem Paket *python-plastex* war es auch schnell installiert. Leider muss ich zu meiner Schande gestehen, das es mir nicht einmal gelang, das ein-

fache Beispiel aus dem Getting-Started-Dokument nachzuvollziehen.

Also ließ ich es bleiben und probierte den nächsten Anwärter: ***tth***. Vom ersten Test mit dem Kommando `tth main.tex` war ich überwältigt. Die gesamte Uptimes, mit sämtlichen Bildern, erschien in einer HTML-Datei, die zwar noch einige Ecken und Kanten hatte, aber doch schon deutlich in die richtige Richtung ging. Natürlich wollte ich nicht die gesamte Uptimes in einer XHTML-Datei, also versuchte ich als nächstes, die LaTeX-Quellen der einzelnen Artikel zu konvertieren – und da trat dann schon wieder Ernüchterung ein. Direkt gelang das nicht. Ich brauchte für jeden einzelnen Artikel ein eigene Hauptdatei `main.tex` mit den nötigsten Informationen. Ich fand keinen Weg, aus der Gesamt-LaTeX-Datei mehrere XHTML-Dateien zu generieren, die zum Beispiel jeweils einen eigenen Abschnitt enthalten (`section`). Also ließ ich auch *tth* erst einmal liegen und schaute mir den nächsten Kunden an.

ConTeXt ist laut eigener Aussage auf der Website in der Lage, EPUBs auszugeben. Aber dummerweise ist es nicht LaTeX-kompatibel, weil es ein eigenes, auf TeX aufbauendes Makrosystem darstellt. Da nutzte es auch nichts, dass Context als Softwarepaket zur einfachen Installation bereitsteht, denn ich wollte ja von den originalen Quellen der Uptimes ausgehen.

Jetzt blieb als letzter Kandidat, den ich untersuchen wollte, ***tex4ht*** übrig, ebenfalls sehr einfach zu installieren. Hier erwies sich ein Konflikt zwischen dem TeX-HTML-Konverter *tex4ht* und dem Hyperlink-Paket *hyperref.sty* [5] als Grund, das nicht weiter zu verfolgen. Meiner Meinung nach sollten Hyperlinks, die in einer Druckversion noch mühsam abzutippen sind, in einem E-Book durchaus mit Anklicken funktionieren.

Also alles zurück auf Anfang – und nochmal überlegen. Die drei Kandidaten *LaTeX2HTML*, *ConTeXt* und *Tex4ht* sortierte ich endgültig aus. Ein zweiter Versuch mit *plasTeX* nach einigen Tagen Pause scheiterte wiederum. Ich begann über Wege nachzudenken, wie ich die Unzulänglichkeiten von *tth* ausbügeln und seine Vorteile nutzen konnte, zum Beispiel die automatische Konvertierung von PDF-Bilddateien in PNG.

Jetzt geht's los mit *tth*

Eine LaTeX-Hauptdatei für jeden Artikel zu erstellen, nur um jeweils eine XHTML-Datei zu erzeugen, bedeutet 20 Dateien für 20 Artikel – die sich im wesentlichen nur in der `input`-Zeile unterscheiden. Das lässt sich automatisieren. Oder, noch besser, da ich nicht zu viele Dateien herumliegen haben möchte und *tth* sich in einer Pipe verwenden lässt: Ich generiere die *main.tex* für *tth* on-the-fly und schiebe sie zur Standardeingabe hinein. Soweit, so gut: Ein Skript muss sowieso her, und das LaTeX-Template, mit der das Skript arbeitet, kommt an sein Ende hinter `__DATA__`.

Die ersten werden es jetzt wohl erkannt haben, und einige stöhnen vielleicht: Das ist ein Perl-Skript. Entgegen hin und wieder vorgebrachten Meinungen kann man lesbaren Perl-Code schreiben, und ich habe es versucht. Um trotzdem aufkommendem Durcheinander im Hirn beim Betrachten des Codes entgegenzuwirken, will ich hier kurz beschreiben, was dieses Skript tut, ebenso ein weiteres, welches die Ausgabe von *tth* nachträglich etwas massiert.

Generell ruft das Makefile `make-epub.pl` die beiden Skripte sowie *tth* auf – ein Makefile war nötig, weil ich mir die vielen Optionen nicht merken kann:

```
01 epub/OEBPS/link_regio-berlin.html:\
02   link/link_regio-berlin.tex\
03   $(TTHHELPER)
04         [ -d epub/OEBPS ]\
05  || mkdir -p epub/OEBPS;\
06          bin/prepare-for-tth.pl\
07    --title 'sage@berlin'\
08        link/link_regio-berlin.tex\
09         | tth -e2 -u -w2 -r\
10         | bin/finish-after-tth.pl\
11         > $@
12
13 link/jens-link.png:\
14   epub/OEBPS/link_regio-berlin.html
```

Das erste Skript *prepare-for-tth.pl* (Zeile 06) bekommt mittels der Option `-title` den Titel des Artikels (Zeile 07) sowie den Dateinamen der LaTeX-Quelle mit (Zeile 08). Daraus erzeugt es den LaTeX-Code und ein paar Makrodefinitionen. Diese reicht es an die Standardeingabe von *tth* weiter (Zeile 09). Ich fand keinen Weg, den Titel zuverlässig aus den LaTeX-Quellen zu extrahieren, darum bin ich mit der Kommandozeilenoption den Weg des geringsten Widerstandes gegangen. Auf die Optionen von *tth* komme ich später zurück. *tth* arbeitet als Pipe, nimmt den LaTeX-Code von der Standardeingabe an und gibt das XHTML an der Standardausgabe aus. Das zweite Skript `finish-after-tth.pl` (Zeile 10) übernimmt den XHTML-Code von *tth*, poliert ihn etwas auf und schreibt ihn schließlich in die Zielda-

tei (Zeile 11). Die Make-Regel schiebt die XHTML-Datei in das richtige Verzeichnis für das EPUB, welches gegebenenfalls vorher angelegt wird.

In der zweiten Regel teile ich dem Makefile mit, dass es die XHTML-Datei erzeugen muss, um die PNG-Datei zu bekommen. Letztere fällt automatisch mit heraus, wenn *tth* die LaTeX-Quelle konvertiert.

Nachdem wir nun wissen, wie das Vorbereitungsskript, *tth* und das Abschlussskript zusammenspielen, schauen wir uns die einzelnen Programme näher an.

Vorbereiten: *prepare-for-tth.pl*

Das Template, mit dem das Skript arbeitet, steht am Ende des Skripts – wie bereits erwähnt. Da es außerdem mit dem Perl-Modul *Pod::Usage* seinen Hilfetext ausgibt, muss es diesen erstmal überspringen, um an das Template zu kommen. Das erledigen diese beiden Zeilen:

```
01 while (<DATA>) { last if /^=cut$/ }
02 while (<DATA>) { push @embed, $_; }
```

Vorher hat es in einer *while*-Schleife bereits die LaTeX-Eingabedatei eingelesen und dabei gleich ein paar Makro-Aufrufe entschärft, mit denen *tth* nicht klarkommt. So wird zum Beispiel `AddToShipOutPicture*` (mit Asterisk) zu `AddToShipOutPicture` (ohne Asterisk), und `put[]` wird zu `put`. Außerdem mache ich einen Kunstgriff, um Bilder und Tabellen zu nummerieren. LaTeX arbeitet mit dem Anker
`label{}` und der Referenz
`ref{}`, um Querverweise aufzulösen. In den Querverweisen steht ein eindeutiger Bezeichner, den LaTeX in der Ausgabedatei zu einer Nummer umrechnet. Ich übergebe den eindeutigen Bezeichner und das davorstehende
`label` beziehungsweise
`ref` an eine Funktion, die jedem Bezeichner eine fortlaufende Nummer zuordnet. Aus `label{xyz}` wird so beispielsweise `label{1}`.

Diese Methode hat einige Einschränkungen. Zum Beispiel funktionieren Querverweise nur innerhalb eines Artikels. Da sowohl Anker als auch Referenz für die Nummerierung verwendet werden, ist es möglich, dass eine Abbildung eine falsche Nummer besitzt, wenn sie ihre Referenz vor einer anderen erhält, aber erst nach dieser anderen Abbildung platziert wird. Schließlich, und das wiegt wohl schwerer, werden Tabellen und Bilder zusammen nummeriert. Das heißt, wenn es ein Bild 2 gibt, kann es nur eine Tabelle 1 und Tabelle 3 geben, aber keine Tabelle 2.

Das Template am Ende des Vorbereitungsskripts enthält Definitionen, die *tth* versteht und die sich aus seiner Dokumentation erschließen lassen. Ich habe lediglich eine Erweiterung eingebaut: Zeilen, die mit `prepare-for-tth` beginnen, werden von dem Skript selbst interpretiert, und zwar mittels der Funktion `prepare()`. Momentan versteht diese Funktion nur die Anweisung `print input`, woraufhin sie die bereits eingelesene LaTeX-Quelle an dieser Stelle einfügt. Ursprünglich war meine Intention, *tth* die Quelldatei via `include dateiname` selbst einlesen zu lassen. Aber im Laufe der Entwicklung stellte ich fest, dass Substitutionen im LaTeX-Quelltext nötig sind, um *tth* auf die Sprünge zu helfen.

Kommandozeilenoptionen für *tth*

Die Eingabe für das Konvertierungstool haben wir nun zusammen. Jetzt wählen wir die Optionen von dem Tool selbst:

```
tth -e2 -u -w2 -r
```

Die erste Option behandelt das Handling der Anweisung *epsfbox.sty*, also die Einbindung von Bildern, die als Postscript oder PDF vorliegen. Mit `-e2` konvertiert das Tool die Bilder mittels *ps2png* sowie *ps2gif* und bindet das Resultat inline ein. Das heißt, ich brauche mit diesen Bildern nichts mehr zu tun. Und nichts habe ich schon immer am liebsten gemacht.

Die Option `-u` schaltet auf die Zeichencodierung Unicode (UTF8). `-w2` beeinflusst den HTML-Stil, und zwar konkret so, dass XHTML erzeugt wird. Die letzte Option bewirkt schließlich, dass am Ende reines XHTML herausfällt, ohne Vor- und Nachspiel, damit es in anderen XHTML-Text Eingang finden kann. Das habe ich gewählt, um größere Kontrolle über den XHTML-Header zu bekommen. Dieser wiederum ist im Template für *prepare-for-tth.pl* definiert.

Damit ist die Arbeit von *tth* getan. Es gibt den Staffelstab weiter – oder vielmehr: Es gibt seine XHTML-Ausgabe weiter an das nächste Skript.

Nachbereiten: *finish-after-tth.pl*

An und für sich ist das gelieferte XHTML browserfest und – wenn man nicht so pingelig ist – gut genug. Allerdings sind einige HTML-Attribute, die *tth* verwendet, nicht für EPUB zulässig. Insbesondere *epubcheck*, der Qualitätssicherer, ist da durchaus sehr pingelig.

Glücklicherweise ließen sich alle Problemzonen, die der Check angemeckert hatte, mit einfachen regulären Ausdrücken beheben. So wird aus dem `<center>` von *tth* ein `<div class="center"`, für `<blockquote>` wird `<p>` eingefügt und einige weitere Konstrukte, etwa `<font ..>`, werden durch geeignete `<div class=..>`-Tags ersetzt. Letztere erlauben dann in Grenzen eine Anpassung des Layouts über CSS.

Damit sind wir durch mit der ersten Teilaufgabe, dem Erzeugen der XHTML-Dateien für das E-Book. Nachdem ich mich ausgiebig mit dem automatischen Erzeugen der XHTML-Dateien beschäftigt habe, wird es Zeit, ein paar Worte zu den Metadaten zu verlieren.

Metadaten für EPUB

EPUB-Metadaten sind zum Teil abhängig von den XHTML-Dateien, die dem EPUB zu Grunde liegen. Sie bleiben aber unverändert, solange keine Dateien hinzukommen oder wegfallen, und solange die Gliederung des Textes gleich bleibt. Also wäre es prinzipiell möglich, die Metadaten einmal pro EPUB zu schreiben, und gut is'. Das ist aber mühsam und eintönig. Es macht viel mehr Spaß, ein Skript dafür zu schreiben.

Zunächst ein paar Worte zu den verschiedenen Metadaten. Sie verteilen sich über die vier Dateien:

- *mimetype*
- *container.xml*
- *content.opf*
- *toc.ncx*

Der Medientyp: *mimetype*

Diese Datei liegt als allererste im Zip-Archiv, die ein E-Book ausmacht. Es enthält nur die Zeile:

```
application/epub+zip
```

Das ist so langweilig, dass ich es gleich hinschreibe und nicht aus dem Skript generiere. Vielleicht in einer späteren Version.

Die Kiste: *container.xml*

Diese Datei befindet sich im Verzeichnis `META-INF` und ist insofern interessant, als dass sie den relativen Pfad von der Wurzel des EPUB-Verzeichnisses zur Datei *content.opf* enthält. Das Skript *make-epub.pl* liest diese Datei ein und ermittelt so den Pfad, wo es die Content-Datei ablegen soll. Die Content-Datei lässt sich auch umbenennen oder in einem anderen Verzeichnis ablegen, solange das im Container notiert ist.

Momentan bricht *make-epub.pl* ab, wenn diese Datei fehlt. In einer späteren Version wird es sie selbst erzeugen.

Der Inhalt: *content.opf*

Das ist die erste Metadaten-Datei, die sich für den Einsatz des Skripts lohnt. Die XML-Datei enthält verschiedene Abschnitte, die das Makefile aus verschiedenen Quellen befüllt.

Für den ersten Abschnitt `<metadata..>` übergebe ich Daten in der Kommandozeile – die ersten drei sind Pflicht für ein EPUB-E-Book, die letzten drei fakultativ:

- `-title Titel` für `<dc:title>`, den Buchtitel);
- `-language de` für `<dc:language>`, die Sprache;
- `-identifier id` für `<dc:identifier>`, einen eindeutigen Identifier, der an verschiedenen Stellen zum Einsatz kommt (lasse ich diese Option weg, verwendet das Skript die Systemzeit als Hexadezimalzahl);
- `-creator autor` für `<dc:creator>`, den Autor;
- `-publisher verlag` für `<dc:publisher>`, den Herausgeber;
- `-rights cr` für `<dc:rights>`, den Urheberrechtsvermerk.

Der nächste Abschnitt in der XML-Datei, `<manifest..>`, enthält eine Liste aller Dateien bis auf `mimetype`, *container.xml* und *content.opf*, die im ZIP-Archiv des EPUBs enthalten sind. Wichtig ist, dass die Liste das Inhaltsverzeichnis *toc.ncx* enthalten muss – das ist die vierte Metadatendatei, zu der wir noch kommen. Die Liste erstellt das Make-Skript automatisch, indem es das Wurzelverzeichnis des EPUB und alle Unterverzeichnisse durchsucht. Für jede Datei gibt es einen `<item>`-Eintrag mit den folgenden Attributen:

- `id` ist ein innerhalb von *content.opf* eindeutiger Bezeichner;
- `href` ist der Pfad zur betreffenden Datei innerhalb des EPUB-Verzeichnisses;
- `media-type` ist der MIME-Typ, den das Skript anhand der Dateiendung bestimmt.

Der dritte Abschnitt heißt `<spine>`. Er enthält eine Liste aller HTML-Dokumente, und nichts

sonst. Die Listenelemente namens `<itemref>` verweisen über das Attribut `idref` auf die entsprechende ID im Abschnitt `<manifest>`. Die Reihenfolge der Listenelemente ist wichtig, da sie die Anzeigereihenfolge im E-Book festlegt.

Da sich die gewünschte Reihenfolge nicht ohne Weiteres beim Durchsuchen der Verzeichnisse erschließt, habe ich hierfür die Kommandozeilenoption `-spine` eingeführt, welche durch Komma getrennt eine Liste der XHTML-Dateien in der gewünschten Reihenfolge entgegennimmt (ohne Pfad und Dateiendung). Das heißt, wenn ich die durcheinander liegenden Dateien *OEBPS/ch1.html*, *OEBPS/epilog.html*, *OEBPS/ch2.html* und *OEBPS/intro.html* in die richtige Reihenfolge bringen will, verwende ich die Option `-spine intro,ch1,ch2,epilog`.

Der letzte Abschnitt , `<guide>`, enthält Verweise auf bestimmte Dateien im EPUB mit spezieller Bedeutung. Im Moment trägt das Make-Skript lediglich den mit Option `-cover` übergebenen vollständigen Dateipfad mit der Umschlagseite ein.

Das Inhaltsverzeichnis: *toc.ncx*

Die letzte Metadaten-Datei, eine XML-Datei, enthält ebenfalls mehrere Abschnitte:

- `head` für Kopfdaten;
- `docTitle` für den Buchtitel;
- `navMap` für das eigentliche Inhaltsverzeichnis.

Im Kopfbereich trägt das Make-Skript für `<uid>` die gleiche ID ein, die es schon für Content.opf verwendet hat. Mit `<depth>` wird die Verschachtelungstiefe des Inhaltsverzeichnisses festgelegt (momentan kann das Skript keine tiefere Verschachtelung als 1). Die Werte für `<totalPageCount>` und `<maxPageNumber>` setzt das Skript auf 0.

Der Bereich `<docTitle>` ist uninteressant, enthält lediglich den Buchtitel. Spannender ist der Teil `<navMap>`. Er listet `<navPoint>`-Einträge für das Inhaltsverzeichnis. Um diese Liste zu generieren, liest das Make-Skript die XHTML-Dateien in der Reihenfolge, wie sie mit der `--spine`-Option festgelegt wurden, und nimmt den Text für den Navigationseintrag aus dem HTML-Kopf, und zwar aus dem Abschnitt `<title>..</title>`. Das ist auch der Grund, warum Make-epub.pl zur Zeit nur die Verschachtelungsebene 1 beim Inhaltsverzeichnis beherrscht.

Damit sind wir mit der letzten Metadaten-Datei durch. Jetzt bauen wir das EPUB zusammen:

```
01 cd epub
02 zip -r -X ../uptimes-2012-02.epub \
03     mimetype META-INF content.opf \
04     toc.ncx OEBPS
```

So legte ich das erste *uptimes-2012-02.epub* (Abbildung 1) der Redaktion vor. Zurück bekam ich eine Liste mit Design-Änderungswünschen. Diese gingen fast ausnahmslos in das Stylesheet *uptimes.css* ein.

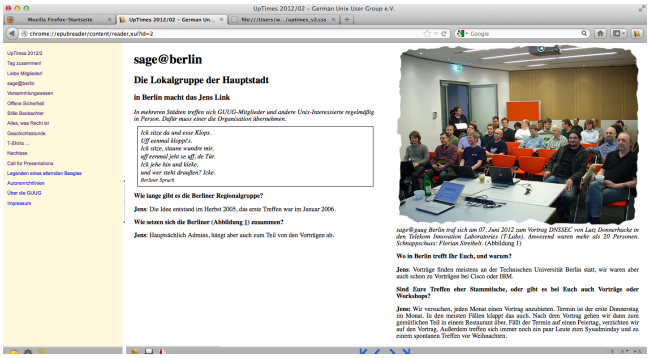


Abbildung 1: So sieht das Uptimes-EPUB in der aktuellen Version im Firefox aus.

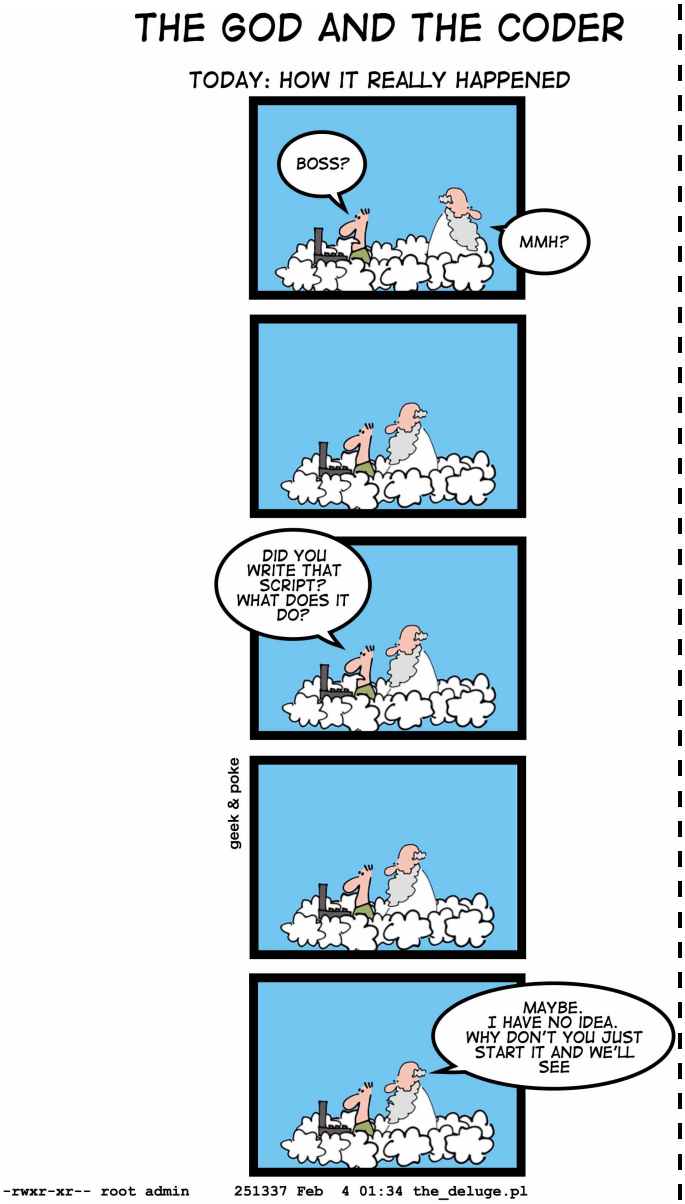
Das Ende der Reise: *uptimes.css*

Für EPUB ist ein Subset von CSS 2.1 zulässig, Details dazu stehen in der EPUB-Doku. Generell bin ich so vorgegangen, dass ich die HTML-Datei eines Artikels in den Browser Firefox geladen und mit dem Webdeveloper-Addon am Stylesheet experimentiert habe, bis es passte. Anschließend habe ich die Änderungen in die richtige Uptimes.css übernommen (Abbildung 2). Am Ende habe ich mit dem Make-File ein neues EPUB erzeugt, den Epubcheck drüber laufen lassen und das EPUB der Redaktion zur Begutachtung vorgelegt. Dafür waren kaum noch Programmierkenntnisse, sondern eher Webdesign-Skills erforderlich.

```
/* Stylesheet for UpTimes */

body {  margin-left:2%;
        margin-right:2%;
        margin-top:2%;
        margin-bottom:2%;
}

.article {
        line-height:1.5em;
}

.caption {
        font-style:italic;
}

.Exkursbox {
        background-color:#e0e0e0;
        border-style:solid;
        border-width:1px;
        font-size:smaller;
        line-height:1.2em;
        margin:.5em;
        padding:.5em;
}

.Exkursbox h1 {
        font-size:180%;
        text-align:left;
}

.fontsize-3 {
        font-size:smaller;
}

.headline4 {
        font-style:italic;
        text-align:left;
}

.headline5 {
        border-style:solid;
        border-width:1px;
        font-style:italic;
        margin:.5em;
        padding:.5em;
}

.headline5 .p { margin:.2em; }

.p { margin:1em; }
```

Abbildung 2: Die CSS-Regeln des Uptimes-EPUB in der aktuellen Version.

Damit ist die Reise zur EPUB-Erstellung der UpTimes beendet. Ich bedanke mich bei allen, die geholfen haben, sowie bei denen, die bis hier gelesen haben. Alle Leser der EPUB-Version sind aufgerufen, Probleme mit und Ideen für dieses Uptimes-Format der Redaktion unter <redaktion@uptimes.de> mitzuteilen.



Literatur

[1] Python-Docutils: http://docutils.sourceforge.net/
[2] EPUB Format Construction Guide: http://www.hxa.name/articles/content/epub-guide_hxa7241_2007.html
Auch als EPUB: http://www.hxa.name/articles/content/EpubGuide-hxa7241.epub
[3] EPUB-Spezifikation: http://idpf.org/epub
[4] *epubcheck*: https://code.google.com/p/epubcheck/
[5] „Apparent conflict between hyperref and tex4ht packages“: http://tex.stackexchange.com/questions/60615

Über Mathias

Mathias Weidner studierte Ende der 80er Jahre des vorigen Jahrhunderts Automatisierungstechnik in Leipzig. Nach verschiedenen Stellen in der Softwareentwicklung landete er bei einem kleinen Rechenzentrum in Wittenberg, wo er als Adminstrator für Unix/Linux und Netzwerke tätig ist. Seit einiger Zeit schreibt er hin und wieder Bücher zum Thema, die gedruckt, als PDF oder EPUB erhältlich sind. Die dabei gewonnenen Erfahrungen machten ihn glauben, dass er die Uptimes als EPUB formatieren könne.

Der Anfang vom Ende
Kluge warten nicht länger mit IPv6

Einige denken: „IPv6? Nicht schon wieder. Das Thema hatten wir schon vor Jahren. Macht doch eh keiner! Und wenn, dann haben wir noch Zeit.“Ja, stimmt. Einige Wenige predigen seit Jahren, dass man sich mit IPv6 beschäftigen solle. Doch es hat sich nicht viel getan.

von **Jens Link**

Besagte Prophezeiungen
wurden nicht von den Maya geschrieben.

Erklärung „Oxlajuj Baq'tun - Baktun 13 - 2012“
von den „Nachkommen der Mayazivilisation“
zu den Weltuntergangsprophezeiungen

IPv4 bietet 256 /8-er Netze. Die IANA (Internet Assigned Numbers Authority) hat im Februar 2011 die letzten fünf davon an die fünf regionalen IP-Registrierungsstellen vergeben [1]. Das für Europa zuständige RIPE (Réseaux IP Européens) in Amsterdam hat sein letztes /8-er Kontingent am 14. September 2012 angebrochen [2]. IPv4-Adressen sind jetzt Mangelware. Es gibt bereits einen Markt für sie. So hat Microsoft im Jahr 2011 Adressen aus der Konkursmasse von Nortel für 11,25 US-Dollar das Stück gekauft [3]. Und die ersten Anbieter erwägen, die Preise für IPv4 zu erhöhen [4]. Ist also wirklich noch Zeit?

Meine Antwort: Nein! Wer jetzt mit der Einführung von IPv6 anfängt, hat eventuell noch genug Zeit, dass Ganze in Ruhe zu tun. Wer erst dann anfängt, wenn es wirklich sein muss, wird an vielen Fronten gleichzeitig kämpfen.

Das Henne-Ei-Problem

IPv6 hat eine Henne-Ei-Problem. Zugangsanbieter bieten in der Regel noch kein IPv6 für Endkunden und führen als Begründung ins Feld, dass es kaum Inhalte gibt, die per IPv6 erreichbar sind. Und die Anbieter von Inhalten? Die begründen die Nichteinführung von IPv6 mit der Tatsache, dass es kaum Zugangsanbieter mit IPv6 gibt. Zwar hat man bereits vor vielen Jahren erkannt, dass der bei IPv4 genutzte Adressraum zuwenige Adressen für die wachsende Anzahl an Endgeräten besitzt. Aber die Einführung von *Network Adress Translation* (NAT) zögerte dieses Problem lange hinaus. Und auch heute dient NAT dazu, IPv4 künstlich am Leben zu erhalten: Zu dem NAT beim Nutzer zu Hause kommt ein zusätzliches NAT beim Provider, oft als *Carrier Grade NAT* bezeichnet.

Das führt natürlich zu Problemen. Die Latenz vergrößert sich – es wird unmöglich, Dienste auf dem heimischen Server laufen zu lassen, auch wenn man bereits an Workarounds tüftelt. Sehr wahrscheinlich wird es Probleme mit SIP und einigen Spielen geben. Für den Zugangsprovider bedeutet das außerdem neben zusätzlichen Kosten für Hardware und Betrieb der NAT-Gateways auch erhöhten Support-Aufwand. Neue Pläne liegen bei den meisten Zugangsanbietern schon in den Schubladen. Glücklicherweise scheinen die meisten erkannt zu haben, dass IPv4-NAT auf ihrer Seite nur eine Notlösung sein kann. Bei einigen Providern scheint der Notfall auch schon eingetreten zu sein: Unitymedia hat DualStack Lite eingeführt, stellt IPv6 nativ zur Verfügung, und tunnelt und nattet IPv4, wie sich einem Twitterdialog des UM-Supports entnehmen lässt [5].

Auch auf Seiten der Inhaltsnabieter führt das zu Problemen. Geolocation von IP-Adressen wird sehr wahrscheinlich ausgehebelt. Da moderne Webseiten viele parallele TCP-Verbindungen aufbauen, kann es dazu kommen, dass Webseiten nur unvollständig dargestellt werden, wenn der Zugangsanbieter zu viele User hinter einer öffentlichen IP versteckt. Einige Betreiber limitieren auch die Anzahl von Verbindungen von einer Quell-IP.

Schritt für Schritt

Es ist also wirklich an der Zeit, sich mit dem Thema IPv6 zu beschäftigen. Man muss ja nicht alles auf einmal machen. Es kann (und wird) noch lange IPv4-Inseln geben. Der Autor dieser Zeilen hat in den letzten drei Jahren noch OS/2, Windows NT und Novell-Server mit IPX-only gesehen: Er wettet darauf, dass ein Teil dieser Systeme in fünf Jahren immer noch aktiv sind (Abbildung 1).

Abbildung 1: Alte Geräte – fast vergessen? Oder ist die Doku auf den neuesten Stand? Der IPv6-Umstieg ist eine prima Gelegenheit, mal aufzuräumen. ©Rainer Sturm, Pixelio

Der wichtiges Punkt überhaupt ist Unterstützung durch die Geschäftsführung. Ohne, dass diese Geld zur Verfügung stellt und vor allem Zeit einräumt, ist das Projekt IPv6-Einführung zum Scheitern verurteilt. Der nächste wichtige Punkt ist Ausbildung. Und zwar nicht nur für ein oder zwei Administratoren, sondern auch Supportmitarbeiter und Entwickler müssen über IPv6-Wissen verfügen. Eine Testumgebung und ein Management, dass entsprechende Zeit einräumt, helfen hier ungemein.

Ohne Dokumentation sollte niemand mit der Einführung von IPv6 beginnen. Zum Beispiel müssen die Netzwerkpläne aktuell sein. Auch eine Liste der eingesetzten Hard- und Software ist extrem wichtig. Ein Beispiel aus der Praxis des Autors: Ein Kunde nutzte seit Jahren Cisco. In einer Außenstelle war noch ein Cisco CAT6500 mit einer Supervisor Engine 1a im Einsatz. Diese Supervisor Engine kann kein IPv6, und es gibt auch schon lange keinen Support mehr. In diesem Fall war neue Hardware unumgänglich. Bei den anderen Switchen des Kunden, ein Cisco 3750 und 3560, war eine Konfigurationsänderung mit anschließendem Reboot und teilweise ein Software-Update nötig. Die Einführung von IPv6 ist also eine gute Möglichkeit, das Netz aufzuräumen und die Dokumentation zu verbessern – beziehungsweise zu erstellen – und sich gegebenenfalls gleich nach passenden neuen Werkzeugen für Dokumentation und Monitoring umzuschauen.

Ein guter Weg ist dann, sich von außen nach innen vorzuarbeiten, also erst die DNS-Server, die Mailserver und den Webauftritt IPv6-fähig zu machen, und dann nach und nach das interne Netz folgen zu lassen. IPv6 einschalten bedeutet nicht, IPv4 auszuschalten. Beide Protokolle können parallel laufen, und sie müssen es sogar, um die reibungslose Kommunikation mit allen Nutzern zu gewährleisten. Natürlich sollte IPv6 auch auf den Arbeitsplätzen der Administratoren und des Supports funktionieren, damit diese testen beziehungsweise Fehler bei Kunden nachvollziehen können.

Prüfen und Planen

Ist grundlegendes Wissen über IPv6 bei allen Beteiligten vorhanden und die Dokumentation in Ordnung, gilt es zu prüfen, ob die eingesetze Hard- und Software IPv6-fähig ist. Unter Umständen stehen hier gerade bei Netzwerkkomponenten wie Layer-3-Switches, Routern, Firewalls und Load-Balancern Upgrades oder Neuanschaffungen ins Haus. Bei (Web-)Applikationen ist zu schauen, ob diese mit IPv6-Adressen zurecht kommen. Hier hilft nur testen, Quellcode lesen oder den Hersteller fragen.

Wenn bekannt ist, welche Änderungen an Hard- und Software anliegen, beginnt die entsprechende Ressourcenplanung. Von seinem Provider benötigt man einen passenden Addressbereich, der sinnvoll aufzuteilen ist [6]. Wenn die Hard- und Software dann IPv6-fähig ist, man Adressen von seinem Provider und einen entsprechenden Adressplan hat, kann man sich an die Einführung von IPv6 machen. Starten sollte man mit der externen Netzwerkanbindung und den Firewalls. Dann folgen die wichtigsten externen Dienste wie DNS, Mail und Web.

IPv6 ist nicht kompliziert, aber der Teufel liegt im Detail. Man muss sich darüber im klaren sein, dass es Probleme geben wird. Geht man aber schrittweise vor, lassen sie sich in den Griff kriegen. Und sie zu lösen wird umso leichter, je mehr Erfahrung man mit IPv6 hat.

Links zum Artikel

[1] Letzte IPv4-Adressen weltweit sind verteilt: http://heise.de/-1181351
[2] Letzter Europäischer IPv4-Vorrat ist angebrochen: http://www.ripe.net/internet-coordination/ipv4-exhaustion/reaching-the-last-8
[3] Microsoft kauft IPv4-Adressen aus Nortel-Konkursmasse: http://heise.de/-1214670
[4] Anbieter erwägen Preiserhöhung für IPv4-Adressen: http://heise.de/-1756209
[5] Twitterdialog des UM-Supports zu IPv6: https://twitter.com/UnitymediaHilfe/status/273395037184876544
[6] Sinnvolles Aufteilen von IPv6-Netzen: http://www.ipbcop.org/ratified-bcops/bcop-ipv6-subnetting/

Literaturtipps

Benedikt Stockebrand, IPv6 in Practice. A Unixer's Guide to the Next Generation Internet, Springer 2007:
Schon etwas älter, aber immer noch eine gute Einführung.
Ciprian Popoviciu/Eric Levy-Abegnoli/Patrick Grossetete, Deploying IPv6 Networks. Cisco Press 2006:
Gut zur IPv6 Einführung – wie der Verlag vermuten lässt, ist es Netzwerk- und Cisco-lastig, enthält aber auch zahlreiche andere Informationen.
Scott Hogg/Eric Vyncke, IPv6 Security. Cisco Press 2008:
Gute Ergänzung zu Deploying IPv6 – hier werden viele IPv6 Security Themen behandelt.
Silvia Hagen, IPv6. Grundlagen - Funktionalität - Integration, Sunny 2009:
Tiefgehende Einführung für jene, die sich näher auseinandersetzen wollen, ohne gleich die RFCs zu lesen.

Über Jens

Jens Link ist freiberuflicher Geek mit einem Fokus auf komplexen Netzwerken und hält seit Jahren Vorträge und Schulungen zu IPv6. Er hat auch schon ein paar Kunden bei der IPv6-Einführung unterstützt.

Nützlicher Veteran
Cisco-Tool Rancid verfolgt Änderungen im Netzwerk

Große Netzwerke erschweren es, alle Änderungen an Netzwerkkomponenten wie Routern, Switches oder Firewalls zu erkennen und zu dokumentieren. Das Tool *RANCID* (Really Awesome New Cisco conflg Differ) erledigt das seit mehr als zehn Jahren, ist aber im Enterprise-Umfeld recht unbekannt.

von **Jens Link**

Wer den Weg nicht kennt,
auf dem er zum Meer gelangen kann,
der sollte sich einen Fluss als Begleiter suchen.

Titus Maccius Plautus, Poenulus

Cisco im Namen ist irreführend: Rancid kann mehr, als nur Cisco-Konfigurationen sichern. Es unterstützt zum Beispiel auch Juniper, HP, F5 und Dell. Einzige Voraussetzung ist ein CLI, über das sich Daten auslesen lassen. Rancid selbst ist in Shell, Expect und Perl geschrieben und lässt sich mehr oder weniger leicht anpassen – dem Code sieht man sein Alter teilweise an.

Rancid verbindet sich über SSH oder Telnet mit einer Netzwerkkomponente, führt dort Befehle aus und speichert die gewonnen Daten in einem Versionskontrollsystem, und zwar CVS oder SVN. Darüber hinaus speichert das Tool die Konfiguration als Textdatei – nützlich als Backup – und verschickt Mails mit den Unterschieden zur Vorgängerkonfiguration an festgelegte Empfänger. Rancid erkennt auch den Austausch von Hardwarekomponenten, zum Beispiel ein neues SFP-Modul oder Netzteil. Voraussetzung ist, dass die Hardware, auf die Rancid zugreift, die entsprechenden Informationen liefert.

Die Verbindung zum Gerät realisiert ein Hardware-spezifisches Skript. Cisco IOS arbeitet zum Beispiel mit dem Skript *clogin* zusammen, für Junipers JunOS ist *jlogin* da. Einige Hersteller – wie auch Cisco – haben verschiedene Betriebssysteme im Umlauf, die sich zwar meist recht ähnlich sind, aber doch im Detail unterscheiden. Hierfür gibt es spezielle Skripte, etwa *xrlogin* für IOS-XR oder *nxlogin* für NX-OS (Cisco Nexus).

Der Aufruf von Rancid geschieht in der Regel via Cronjob. Mit etwas Handarbeit startet es auch Incident-gesteuert, etwa wenn im Logfile eine bestimmte Meldung auftaucht, oder wenn die Komponente einen entsprechenden SNMP-Trap schickt.

Konfigurieren

Am einfachsten lässt sich Rancid über das Paketmanagement der Distribution installieren. Kleiner Tipp zu Debian GNU/Linux: Wer dem User *rancid* eine Login-Shell in der `/etc/passwd` gibt, erleichtert sich die spätere Arbeit mit `su - rancid`. In der Konfigurationsdatei *rancid.conf* heißen die wichtigsten Variablen `CVSROOT`, `RCSSYS`, `LIST_OF_GROUPS` und `FILTER_PWDS`.

Die ersten beiden legen fest, ob CVS oder SVN zum Einsatz kommt. `LIST_OF_GROUPS` verzeichnet Gruppen, die in größeren Umgebungen ermöglichen, zum Beispiel zwischen LAN, WAN und Security zu trennen. `FILTER_PWDS` schließlich steuert, ob Rancid Passworte wegschmeißen oder lieber speichern soll. Standardmäßig steht der Wert auf `YES`: Passworte werden also herausgefiltert, die Konfiguration lässt sich dann nicht 1:1 als Sicherung zurückkopieren. Dafür verschickt das Tool dann aber auch keine Passworte im Klartext per E-Mail.

Damit Rancid Mails verschickt, bedarf die Datei `/etc/aliases` zwei weiterer Einträge je Gruppe: `rancid-gruppenname` und `rancid-admin-gruppenname`. An die erste Adresse verschickt Rancid Mails bei Änderungen an Geräten innerhalb der Gruppe. An die zweite Adresse gehen Mails bezüglich der Rancid-Administration.

Festzurren

Nachdem alles installiert und die globale Konfiguration abgeschlossen ist, sind noch ein paar Kleinigkeiten zu erledigen. Das Kommando `rancid-cvs` initialisiert das Versionskontrollsystem von Rancid – auch, wenn man SVN ge-

wählt hat. Das Beispiel geht davon aus, dass nur die Gruppe *lan* in der Konfigurationsdatei `/etc/rancid.conf` steht. Das Kommando erzeugt dann das korrespondierende Verzeichnis *lan* mit einigen Dateien und Verzeichnissen:

```
01 root@pc8:~/su - rancid
02 rancid@pc8:~/bin/rancid-cvs
03 root@pc8:~/cd lan
04 rancid@pc8:~/lan/ls -la
05 drwxr-x--- 3 rancid rancid 4096 Dec 10 21:23 \
       configs
06 -rw-r----- 1 rancid rancid 0 Dec 10 21:23 \
       router.db
07 -rw-r----- 1 rancid rancid 0 Dec 10 21:23 \
       routers.all
08 -rw-r----- 1 rancid rancid 0 Dec 10 21:23 \
       routers.down
09 -rw-r----- 1 rancid rancid 0 Dec 10 21:23 \
       routers.up
```

Das Verzeichnis `configs` beheimatet die gesicherten Konfigurationen. Die Datenbank `router.db` verzeichnet die einzelnen Geräte. Die Syntax dieser Datenbank ist sehr einfach, wie folgendes Beispiel für drei Cisco Geräte zeigt, zwei davon aktiv, eines abgeschaltet:

```
01 rancid@pc8:~/lan/cat router.db
02 192.0.2.1:cisco:up
03 switch1:cisco:up
04 192.0.2.20:cisco:down
```

Jeder Eintrag enthält Hostname oder IP-Adresse, gefolgt von Gerätetyp und Status (mögliche Zustände: up und down). Wer down ist, den fragt Rancid nicht ab. In der Version 3, als Alphaversion im November 2012 veröffentlicht, ersetzt ein Semikolon den Doppelpunkt als Trenner. Nach vielen Jahren hat sich auch hier IPv6 herumgesprochen. `:-)`

Damit sich die verschiedenen Skripte an den jeweiligen Geräten anmelden können, muss der Admin die Datei `.cloginrc` im Home des Rancid-Users anlegen. Diese Datei darf nur vom User *RANCID* lesbar sein. Die Datei definiert für einzelne Netzbereiche und Hosts Benutzer, Passworte und Methoden. Im Beispiel gelten für alle Geräte im Netz der Nutzer `Admin`, das Passwort `geheim` mit dem Enable-Password `secret` und als Methode SSH:

```
01 rancid@pc8:~/cat .cloginrc
02 add user * admin
03 add password * {geheim} {secret}
04 add method * ssh
```

Testen und zweckentfremden

Bevor man einen entsprechenden Cronjob einrichtet, empfiehlt sich ein Probelauf mit `rancid-run`. Das Programm selbst ist sehr schweigsam, legt aber für jeden Durchlauf ein Logfile an, in dem mögliche Fehler zu finden sind. Ist alles in Ordnung, findet man im Configs-Verzeichnis die entsprechende Konfiguration. Ist alles fertig eingerichtet, bekommt man bei Änderungen Mails. Im folgenden Beispiel wurde der Log-Server geändert:

```
01 no ip http server
02 - logging 192.168.73.3
03 + logging 192.168.73.9
04 snmp-server community foo RO
05  radius-server host 192.168.73.9 auth-port \
        1812 acct-port 1813 key Cisco
06  radius-server retransmit 3
```

Die Login-Skripte von Rancid lassen sich prima zweckentfremden, um Konfigurationsänderungen an mehreren Geräten vorzunehmen. Das Beispiel ändert den NTP-Sever auf drei Cisco-Geräten:

```
01 rancid@pc8:~/bin/clogin -c "conf t; \
      no ntp server 10.10.20.10; \
      ntp server 10.10.10.9; end; wr; " \
      192.0.2.1 192.0.2.15 192.0.2.17
```

Wenn der GAU kommt
Datensicherungsstrategie erarbeiten und umsetzen

Wenn alle Unternehmensrechner ihre Daten verlieren, kann nur ein kleiner, unbedeutender Systemadministrator mit einem Schrank voll Sicherungsbänder das Schicksal wenden. Wehe ihm, wenn dann nicht alle Werkzeuge bereitliegen. Dieser Artikel zeigt, welche Maßnahmen er ergreift und welche Fallen er umgeht, um eine gute Datensicherungsstrategie zu besitzen.

von **Stefan Schumacher**

Sicher ist,
dass nichts sicher ist.
Selbst das nicht.

Joachim Ringelnatz

Komplette Kinofilme auf einem Handy herumzutragen, ist heute kein Problem mehr. Mit den steigenden Datenmengen wachsen die Anforderungen an Datensicherungsmethoden. Die Datenmengen selbst sind nicht das Problem – umständlich ist nur, in so genannten organisch gewachsenen Netzen die Daten zu bestimmen, die zu sichern sind. Sobald sie in den Untiefen des Sicherungssystems verschwinden, sollten sie dann auch wieder herauszubekommen sein. Eine alte Administratorenweisheit besagt: Niemand will Backup. Aber alle wollen Restore.

Dieser Artikel gibt Tipps für eine Datensicherungsstrategie und ihre Umsetzung. Es geht nicht um die Technik, sondern ums Verwalten (lateinisch: administrare). Einige dieser Tipps hat der Autor mit Blut, Schweiß und Tränen erkauft.

Disaster Recovery Planning

Ein Notfallplan, neudeutsch *Disaster Recovery Plan*, dient zur Betriebswiederherstellung im Notfall. Er ist das Fundament der Sicherungsstrategie. Für die Organisation einer Sicherungsstrategie ist eine Anforderungsanalyse vorzunehmen und die gewonnenen Erkenntnisse umzusetzen. Schritte dahin sind:

- Inventur machen
- Bedrohungsszenarien analysieren
- Wichtige Daten identifizieren
- Systeme sichern
- Dokumentieren
- Testen

Bei der **Inventur** katalogisiert der Admin, welche Hardware, Betriebssysteme, Anwendungsprogramme und Dienste auf seinen Maschinen laufen. Das ermöglicht, Anforderungen an die Sicherungssoftware festzulegen. Existiert schon ein Inventar, zum Beispiel in einer Datenbank oder einem speziellen Inventurprogramm, ist dieses um die Sicherungsinformationen zu erweitern.

Bei den **Bedrohungsszenarien** geht es um die Fragen, wer oder was die Daten bedroht, und wie der Schutz davor aussehen kann. In der Regel gibt es bestimmte Fehlertypen:

Benutzerfehler: Der Benutzer löscht oder verfälscht Dateien. Hier muss die letzte vorhandene Version zurückgespielt werden. Dieser häufige Fehler erfordert eine Archivierung der Daten oder Sicherungsbänder.

Administratorenfehler: Wenn er vorkommt, hat er es in sich. Hierfür sollte das gesamte System zum Beispiel mit Snapshots gesichert werden.

Systemfehler: Bei Festplattendefekt schützen Sicherungen auf Wechselmedien, anderen Rechnern oder RAID-Systemen. Gegen Dateisystemkorruption helfen nur archivierte Sicherungen, da sich ein Fehler auf die Spiegel fortpflanzt.

Einbruch: Bei elektronischem Einbruch ist die Frage zu stellen, ob die Daten noch integer sind, sofern sie nicht gelöscht wurden. Datenvernichtung erfordert die Rücksicherung. Aber im Falle manipulierter Daten sind Archive notwendig. Gegen herkömmlichen Einruch, also Vandalismus oder Diebstahl, hilft die letzte Datensicherung nur, wenn sie der Einbrecher nicht finden konnte.

Naturkatastrophen: Bis zum Sommer 2002 haben einige Unternehmen nicht im Traum daran gedacht, das die Elbe mal in ihren Kellern vorbeischaut. In Deutschland sind also Fluten an man-

chen Standorten eine Gefahr. Weiterhin sind Flugzeugabstürze, Erdrutsche, Großbrände oder Unglücke in benachbarten Unternehmungen (Chemiewerk, E-Werk, Raffinierie) möglich. Was passiert also, wenn unser Gebäude oder Stadtteil ausradiert wird, unser Unternehmen aber die Daten noch benötigt? Existieren Sicherungen an anderen Orten? Sicherungsbänder sollten entsprechend dem Gefahrenpotential ausgelagert werden, zum Beispiel in andere Filialen. Ebenso möglich ist eine Spiegelung oder Replikation der Datenbestände auf entfernte Systeme.

Wie **identifiziert** man nun die wirklich wichtigen Daten – also die, deren Verlust nicht akzeptabel ist? Drei Kriterien helfen:

1. Die Daten sind bei Verlust nicht wiederherstellbar, da sie an einen bestimmten Zeitpunkt gebunden sind – wie Patientendaten, Logdateien, Versuchsprotokolle – oder an bestimmte Programme und Bearbeiter.
2. Die Daten sind zwar rekonstruierbar, aber der Aufwand dafür übersteigt den Aufwand der Sicherung. Oder die Nichtverfügbarkeit der Daten verursachte gleich einen wirtschaftlichen Totalschaden.
3. Es handelt sich um Betriebssysteme, Anwendungen oder Dienste. Kann oder möchte man diese nicht sichern, dann zumindest deren Konfigurationsdateien.

Herzstück: Datensicherung

Zur **Datensicherung** gehören neben den Home-Verzeichnissen der Benutzer im Allgemeinen die Datenbestände aus Anwendungen, etwa Datenbanken oder CAD-Systeme. Auch Konfigurationsdateien oder modifizierte Programme sollten nach jeder Veränderung gesichert werden. Systematische Dateiablage vereinfacht die Sicherung. Das ist ein Knackpunkt, wenn Heim-Betriebssysteme eingesetzt werden, die es dem Benutzer erlauben, seine Dateien wahllos im System zu verstreuen – es sei denn, man kopiert sowieso die komplette Platte. Verzeichnisstrukturen vereinfachen auch die Sicherung von Daten, die nicht oft verändert werden, etwa Bilder, Fotos, Audiodateien oder Filme.

Einige Sicherungssysteme verwenden einen Index, insbesondere solche für Netzwerksysteme wie *Amanda* oder *Bacula*. Amanda legt Logdateien an, Bacula verwendet eine Datenbank. In diesen Index wandern alle Metadaten zur Sicherung nach dem Schema: Wann wurde welche Datei von welchem Client in welcher Version auf welches Band gesichert. Dieser Index ist also wichtig, wenn man die Rücksicherung einer bestimmten Version durchführen muss. Ohne den Index kann man zwar in der Regel noch die Bänder von Hand zurückspielen, dies ist aber bei einer größen Anzahl mühselig. Daher sollte zwingend auch das Sicherungssystem selbst gegen Ausfälle und Verlust gesichert werden. Verwendet man eigene Skripte, die einen Katalog der gesicherten Dateien erstellen, sollten diese ebenfalls gesichert werden.

In Netzwerken, in denen nur Teile der Clients gesichert werden, ist es wichtig, die Benutzer der Systeme über die Strategie zu unterrichten. Denn die Anwender sollen ihre Daten nur in Verzeichnissen speichern, die auch gesichert werden. Dürfen die Benutzer auf den Clients selbständig Programme installieren, muss bereits im Vorfeld dafür gesorgt werden, dass deren Anwendungsdaten ebenfalls mitgesichert werden. Homogenität vereinfacht natürlich die Datensicherung. Meistens ist es aber nicht möglich, alle Systeme homogen aufzusetzen. Es ist hingegen möglich, die Systemdienste anzugleichen, Benutzerdaten zentral zu verwalten und gleiche Hardware einzusetzen.

Ebenso erleichtert Systematisierung im Tagesgeschäft die Sicherung. Viele erledigen bestimmte Handgriffe manuell, etwa neue Benutzer anlegen oder den Datenbankserver warten. Sie beherrschen die Befehle aus dem Effeff. Weitsichtiger ist es aber, diese Befehle in ein Shellskript zu gießen. Das hat den Vorteil, dass die Skripte mitgesichert werden, also nach einem Systemausfall wiederhergestellt werden können. Außerdem kann man ein Skript leicht kommentieren und somit dokumentieren. Im Zweifelsfall wissen also auch die Kollegen, was auf den Maschinen abläuft.

A propos Skript: Wer einzelne Systeme nachts per Cronjob erledigt, sollte seine Skripte robust halten, indem er zum Beispiel Toleranzen einplant. Kommt es nämlich dazu, das sich die Laufzeit der Skripte verschiebt, kann unter Umständen der gesamte Sicherungsalgorithmus fehlschlagen, wie das Beispiel im Kasten *Cronjob trifft auf Batchjob* zeigt.

Praxisbeispiel: Cronjob trifft auf Batchjob

Ein Unternehmen wollte vier kleine Arbeitsgruppenserver mit verschiedenen Unix-Varianten sichern. Dazu stand ein älterer Rechner mit Spoolingplatte und Streamer zur Verfügung. Nächtlich sollten die vier Server ihre Daten per `dump` und SSH-Tunnel auf den Spoolserver schieben, der diese danach auf ein Streamerband schreibt. Da alle Rechner im selben Raum standen, verkabelte sie ein einfacher Switch extra für die Sicherung mit TBase100. Ein Cronjob auf den vier Servern startete mit einer Stunde Abstand den Dump-Lauf und auf dem Spoolingserver die endgültige Sicherung auf Band.
Da das Backup-Netzwerk nur für das Backup genutzt wurde, bemerkte niemand, dass ein Port am Switch massive Hardwareprobleme entwickelte und daher die Sicherung über das Netzwerk extrem langsam wurde. Als der Spoolingserver per Cron begann, die Daten auf Band zu schreiben, lieferte der dritte Server seine Sicherung immer noch an die Spoolingplatte aus. Der Spoolingserver verwendete zwar `dump`, um das Band zu schreiben, sodass die anderen Sicherungen fehlerfrei waren – das Archiv vom dritten Server war aber defekt. Das Ganze flog erst auf, als ein Admin früher als gewöhnlich in der Firma auftauchte und bemerkte, dass der Sicherungsprozess noch läuft.
Nachdem der Netzwerkfehler behoben war, wurde das Skript abgeändert. Und zwar so, dass nächtlich per Cron auf dem Spoolingserver ein Shellskript anlief, das nacheinander die Sicherungsläufe auf den Servern anwarf und erst nach erfolgreichem Abschluss dieser Sicherungen die Archive auf Band schrieb. Abschließend wurden die Logdateien aller Dump-Läufe per Mail an die Administratoren weitergeleitet – und von diesen auch gelesen.

Der Admin ist der wichtigste Faktor

Die meisten Sicherungsprogramme erstellen Logdateien. Dies sollte man aktivieren – und die erzeugten Protokolle unbedingt lesen. Denn nur so ist sichergestellt, dass alle Sicherungsläufe auch wie erwartet funktionieren. Wann immer Daten übertragen oder bearbeitet werden, kann dabei etwas schief gehen. Daher sollte jedes Programm, das Daten verifizieren kann, dies auch tun. Bei den Kompressionsprogrammen Bzip2 und Gzip zum Beispiel dient dazu die Option `-t`. Ansonsten erledigen das Prüfsummen wie MD5 oder SHA1. Komplette Dateisysteme lassen sich mit *mtree(8)* überprüfen und vergleichen.

Wichtig ist, die Konfiguration aller Rechner und deren Software zu kennen, und zwar nicht nur als Datei, sondern auch als Ausdruck. Als Letztes benötigt man eine Live-CD oder eine separate Festplatte. Die Live-CD lässt sich mit `pkgsrcsysutilsmklivecd` selbst erstellen. Zu denken ist an Unterstützung für alle benötigten Geräte (Laufwerke, Netzwerkkarten, CGD, RAID, LVM) und Pakete. Alternativ kann man auch eine kleinere Festplatte mit NetBSD einrichten und alle benötigten Programme einbinden.

Zum Sicherungssystem gehört auch der Admin. Es darf nicht nur eine einzelne Person wissen, wie das Sicherungssystem funktioniert. Schließlich ist die auch auch mal im Urlaub. Oder im Krankenhaus. Es muss sichergestellt sein, dass auch andere Administratoren als der implementierende die Systemsicherung beherrschen. Daher ist es lebenswichtig, eine funktionierende Dokumentation des gesamten Systems zu erstellen. Idealerweise sollte hier der gesamte Entwurf und die gesamte Implementierung des Systems formal dargelegt werden – zusätzlich aber auch eine einfache und leicht verständliche Schritt-für-Schritt-Anleitung zur Rücksicherung von Daten vorhanden sein.

Nützlich ist es, die Einweisung in das Sicherungssystem mit der Dokumentation als Testlauf anzusetzen. Hierbei sollen die beteiligten Admins anhand der Dokumentation eine Rücksicherung auf ein Testsystem durchführen. Das stellt die Qualität sowohl der Dokumentation als auch des Sicherungssystems bequem auf die Probe: Versteht jeder, was beschrieben wurde? Fehlen Teile? Gelingt die Rücksicherung? Nach Abschluss der Rücksicherung dient eine gemeinsame Manöverkritik dazu, Schwächen und Fehler des Systems und der Dokumentation durchzuprechen und abzustellen.

Nicht nur der Administrator sollte in das Sicherungssystem eingewiesen werden. Insbesondere in kleinen bis mittleren Unternehmungen ist das ein Problem, weil die oft keine EDV-Abteilung besitzen, sondern nur einen Teilzeit-Admin beschäftigen. Ist der weg, weiß keiner bescheid. Lösungen lassen sich über einen externen Berater finden, der im System eingewiesen und vertraglich entsprechend in den Ablauf eingebunden wird.

Damit sich die ganze Mühe lohnt: Eine Inventur der Medienbestände ist regelmäßig angebracht, bei der neben der Vollständigkeit auch die Funktionsfähigkeit getestet wird. Bänder sollten mindestens einmal pro Jahr einmal durchgespult werden.

Testen, testen, testen

Ein Sicherungssystem, das nicht unter realen Bedingungen getestet wurde, kann niemand als funktionierend bezeichnen. Das hat mehrere Gründe. Zunächst müssen die Administratoren tatsächlich in der Lage sein, die Rücksicherung durchzuführen. In der Regel erzeugt ein Systemausfall mit Datenverlust Stress, denn der Chef will das Problem am besten gestern gelöst sehen. Daher ist es unerlässlich, derartige Situationen und Verfahren in Ruhe durchzuspielen, um Sicherheit im Ablauf zu bekommen. Das verringert den Stress und somit Fehlerquellen.

Außerdem ist ohne Tests nicht sichergestellt, dass die Sicherungsstrategie auch wirklich funktioniert. Man kann zwar versuchen, im Entwurf der Strategie Fehlerquellen zu minimieren, kann diese aber nicht ausschließen, wie in jedem Entwurfsprozess. Nur Testläufe stellen sicher, das die Prozedur funktioniert und auch wirklich die gewünschten Daten sichert. Folgende Testläufe spielen die gängigsten Situationen durch:

- Rücksicherung einzelner Dateien
- Sichern von Versionen einzelner Dateien
- Rückspielen eines Client-Dateisystems
- Neuaufsetzen eines Komplettsystems, also Betriebssystem mit Konfiguration, Anwendungssoftware und Daten
- Rücksicherung einer Archivversion zu einem bestimmten Zeitpunkt in der Vergangenheit (*Point-in-Time-Recovery*)
- Rücksicherung eines Systems, obwohl der Backup-Server ausgefallen ist

Woran man alles denken muss...

Aus Datenschutz-, rechtlichen oder technischen Gründen kann es notwenig sein, das Netzwerk in verschiedene Sicherungskreise zu unterteilen. So sollten Forschungsdaten anders behandelt werden, als das Datenverzeichnis des Webservers. Erstere sollten verschlüsselt im Tresor liegen, zweitere sind diesen Aufwand in der Regel nicht wert. Und so, wie der Gesetzgeber für einige Daten eine Mindestarchivierungszeit vorschreibt, legt er auch Obergrenzen fest. Beispielsweise sind Eintragungen in der Personalakte über Disziplinarmaßnahmen nach Eintritt des Verwertungsverbots meistens nach zwei Jahren von Amts wegen zu entfernen. Das heißt, dass binnen dieser Frist angelegte Sicherungskopien der Akte zu vernichten sind. Daher ist es sinnvoll, die Personalabteilung in einen eigenen Sicherungskreis zu gliedern und auch entsprechend zu archivieren, da man sonst juristisch belangt werden kann.

Unternehmen sichern oft nur an Werktagen. Wenn doch einmal jemand an einem Feiertag oder am Wochenende arbeitet, dann will es Murphy's Law, dass es dann garantiert zu einem Datenverlust kommt. Sind die entsprechenden Rechner also außerhalb der Werkszeiten an, können sie über das Netzwerk gesichert werden. Falls nicht, sollte die Netzwerkrichtlinien und Benutzerordnungen auf die fehlende Sicherung hinweisen. Bestimmte Maßnahmen ermöglichen auch eine Sicherung der Daten am Wochenende, zum Beispiel ein Fileserver, auf den die Mitarbeiter Kopien der zu sichernden Dateien ablegen können.

Tipps zum Komprimieren und Verschlüsseln

Wer Archive komprimiert, um Platz zu sparen, sollte bedenken, dass Archivfehler die Dekomprimierung oft unmöglich machen. Daher sollte man nach entsprechender Kosten-Nutzen-Abwägung die gesicherten Archive verifizieren. Ebenso ist es sinnvoll, nur gängige Komprimierungsprogramme zu verwenden, um auch später wieder an die Daten heranzukommen. Aktuelle Bandlaufwerke unterstützen auch Hardwarekomprimierung. Das heißt, dass das Laufwerk vor dem eigentlichen Schreiben der Daten diese in einem eigenen Chip komprimiert. Dieses Verfahren hat den Vorteil, die sichernde Maschine nicht weiter zu belasten.

Ob man Komprimierungsverfahren einsetzt, hängt von der entstehenden Last (Netzwerk, CPU, Bandbedarf) und Zusammensetzung der Daten ab. Besteht die Mehrheit aus natürlichsprachigen Texten wie Logdateien, Textdokumenten oder Text-Datenbanken, ist eine recht große Kompressionsrate gegeben. Bereits komprimierte Datentypen wie JPEG, MP3, AVI oder auch verschlüsselte Daten hingegen lassen sich nicht weiter komprimieren. Zur Komprimierung eignen sich die Standardprogramme *Gzip* und *Bzip2*, da sie weit verbreitet sind und bisher zuverlässig arbeiten. Bzip2 erzeugt zwar im Allgemeinen eine bessere Komprimierungsrate, erzeugt dabei aber auch höhere Systemlast und Laufzeiten.

Es ist sinnlos, ein verschlüsselndes Dateisystem einzusetzen und unverschlüsselte Backups im Schrank liegen zu haben. Daher sollten auch Backups verschlüsselt werden. Dies gilt insbesondere dann, wenn die Sicherungsbänder extern gelagert werden. Verschlüsseln kann man Archive sicher mit *OpenSSL*, *GnuPG* oder dem symmetrischen Verschlüsselungprogramm *mcrypt*, das verschiedene Algorithmen wie *Rijndael*, *3DES* oder

Panama unterstützt. Im Fall von GnuPG empfiehlt es sich, symmetrische Verschlüsselung einzusetzen. Denn für die Entschlüsselung benötigt man bei der asymmetrischen Verschlüsselung sonst wieder den GnuPG-Schlüssel – was das Archiv unbrauchbar macht, wenn der nicht mehr existiert.

Werden die Archive nicht mittels einer Pipe direkt verschlüsselt auf Datenträger geschrieben, müssen die unverschlüsselten Dateien sicher gelöscht (gewipet) werden. Dazu gibt es zwar die Option `-P` des *rm*-Kommandos, die Dateien vor dem Löschen dreimal überschreibt. Diese Option genügt aber nicht den allgemein anerkannten Regeln der Technik. Daher sollte dringend ein Programm verwendet werden, das zumindest den US-Sicherheitsstandard *5220.22-M(ECE)* oder Peter Gutmanns Standard implementiert. Im Verzeichnis *pkgsrc* befinden sich dazu die Tools `pkgsrcsecuritydestroy` oder `pkgsrc/sysutils/wipe`. Empfehlenswert ist im Fall von NetBSD, die unverschlüsselten Archive in eine mit CGD verschlüsselte Partition zu schreiben und danach an der Quelle zu wipen. Hierzu ist auf einzelnen Rechnern eine hinreichend große, mit CGD verschlüsselte *tmp*-Partition nützlich.

Folgendes Listing erzeugt einen Dump, der sofort per Bzip2 komprimiert wird. Anschließend wird der Dump mit GnuPG symmetrisch verschlüsselt und mit Destroy sicher gelöscht:

```
01 # dump -0a -f - /etc/ | \
            bzip2 > dump.bz2
02 # gpg -a -c dump.bz2 && \
            destroy -f -s 7 t dump.bz2
```

Das nächste Beispiel erzeugt einen Dump, der sofort per Pipe an Bzip2 zum Komprimieren und an OpenSSL zum Verschlüsseln geschickt wird:

```
01 # dump -0a -f - /etc/ | bzip2 | \
       openssl aes-256-ecb -out \
       dump.bz2.enc -e -salt
02 # openssl aes-256-ecb -in \
        dump.bz2.enc -d -salt | \
        bunzip2 - | restore -r -f -
```

Die zweite Zeile ist des Gegenstück zur ersten, denn hier wird der verschlüsselte und komprimierte Dump entschlüsselt, dekomprimiert und an Restore geschickt.

Eine Frage der Medien

Als Medium für die Datensicherung eignet sich prinzipiell alles, was am Markt erhältlich ist, jedoch sollte man einige Vorüberlegungen treffen.

Verlässlichkeit: Die Medien müssen manchmal jahrelange Archivierungsperioden überstehen können. Außerdem müssen Lesegeräte noch verfügbar sein. es bietet sich an, zum Beispiel zusammen mit der Inventur regelmäßig die ältesten Medien auf Lesbarkeit zu überprüfen, und gegebenenfalls das Medium zu wechseln. Alle acht bis zehn Jahre ist eine komplette Migration der Medien nötig, da sie und ihre Lesegeräte veralten.

Geschwindigkeit: Langsam sollte das Sicherungssystem nicht sein. Doch die Kapazitäten des Netzwerks und des Sicherungsservers sind zu beachten, denn ein 50-MBit-Streamer verliert in einem 10-MBit-Netz seinen Vorteil. Auch senkt ein Bandwechsel den Durchsatz des Systems, wenn sich Sicherungslauf auf mehrere Bänder erstreckt.

Dauer der Rücksicherung oder *Time to Data*: Die Zugriffszeit der Medien spielt genauso eine Rolle wie die Organisation der Archivierung selbst. Mehr als 90 Prozent aller Rücksicherungen betreffen nur einzelne Dateien, die in älteren Versionen restauriert werden müssen. Der gesamte Zeitaufwand der Operation besteht darin, die richtigen Medien mit der gewünschten Version zu finden, einzulegen und die Dateien zurückzuspielen. Verwendet man hierzu ein automatisiertes System (Index, Bandwechsler), geht das schneller, als wenn man die Daten von Hand zurückspielt. Ist man auf besonders schnelle Zugriffszeiten angewiesen, verwendet man hierarchische Speichersysteme, die verschiedene Medientypen mit unterschiedlichen Zugriffzeiten einsetzen. Häufig benötigte Dateien landen dann auf schnellen Medien wie Festplatten oder magneto-optischen Medien, während selten benötigte Daten auf entsprechend langsameren, aber größeren Medien wie Magnetbändern ruhen. Diese Methode war früher, als Festplatten noch in Megabyte vermessen wurden, sehr beliebt, und datenintensive Umgebungen wie Grafik- oder Videobearbeitung setzen sie noch heute ein.

Kapazität: Die anfallenden Datenmengen müssen auf möglichst wenige Medien passen. Verwendet man einen einfachen Streamer, ist besonders wichtig, alle Daten auf einem einzigen Band zu haben – denn niemand möchte nachts um drei neben dem Streamer Wache schieben. Beachten sollte man auch, dass steigende Kapazität die Zeit zur Rücksicherung anhebt, denn ein 10-GByte-Band lässt sich schneller durchspulen als ein 100-GByte-Band. Aus betriebswirtschaftlicher Sicht ist es zwar unnötig, einen 100-GByte-Streamer mit entsprechend großen und teuren Medien anzu-

schaffen, wenn täglich nur 5 GByte Daten anfallen. Andererseits sollte man auch nicht zu kurz planen, da die Hardware wenigstens die vier Jahre bis zur Abschreibung im Einsatz bleiben sollte.

Kosten: Das System darf üblicherweise nichts kosten. Wozu braucht man schon ein Backup-System, wenn doch alles prima läuft? ZIP-Medien, CD/DVD und Festplatten sind weit verbreitet, billig und von daher gut zur Rücksicherung geeignet. Nachteilig wirkt sich allerdings deren relativ geringe Integrität und kurze Lebensdauer aus. Kostspielig und langsam, aber dennoch überzeugend durch hohe Kapazitäten, exzellenter Integrität und langer Lebensdauer sind Magnetbänder. Es bietet sich an, verschiedene Medien zu mischen, zum Beispiel nur USB-Sticks oder DVD-RW für tägliche Sicherungen zu verwenden. Festplatten, die Daten sichern, sollten als Wechselplatte ausgeführt sein und offline gelagert werden.

Richtig lagern

Bänder müssen aufrecht stehend gelagert werden, da sich sonst das Magnetband lockern kann – und niemand will Bandsalat bei einer Rücksicherung erleben. Physikalische Sicherung gegen Einbruch und Brandschutzeinrichtungen sind ebenso obligatorisch wie eine restriktive Zugangspolitik.

Wer richtig große Datenmengen zu schultern hat, für den existieren automatische Bandwechseleinheiten, die die Medien und Hüllen mit einem Strichcode etikettieren. Für alle anderen gilt: Da die Anzahl der Medien mit der Zeit wächst und die Archivierung komplexer wird, bietet sich ein datenbankengestütztes Inventarisierungssystem an. Die Datenbank erfasst alle Eigenschaften, die helfen, das passende Sicherungsband zu finden. Folgende Eigenschaften sollten darin stehen:

1. Primärschlüssel
2. Name
3. Zweck
4. Medientyp
5. Lagerort
6. Gesicherte Dateien inklusive Datum, Version, Eigentümer, Attribute

Der Primärschlüssel dient der Identifikation des Mediums. Das kann eine laufende Nummer sein, sinnvoller ist aber, in ihm Informationen zu kodieren, etwa Datum, gesicherte Daten, IP-Adresse oder Level. Der Primärschlüssel steht auf jedem Medium und der entsprechenden Hülle. Die Eigenschaft *Name* ist für jene da, die die Medien nicht über den Primärschlüssel ansprechen möchten. Zweck, Medientyp und der Lagerort der Medien gehören ebenfalls in den Katalog. Das ist notwendig, um im Falle einer Rücksicherung das passende Medium zu finden.

Sicherungsvarianten

Die verschiedenen Arten, Daten zu sichern, sind zum Abschluss hier aufgeführt. Der anschließende Kasten enthält ein Beispiel für eine Sicherungsstrategie. Und für die Spielernaturen zeigt der letzte Kasten, wie man mit Dump-Levels *Türme von Hanoi* spielt. Da bekommt der Begriff **Diskjockey** doch gleich eine neue Bedeutung!

Ein **Komplettbackup** sichert alle Daten. Das ist einfach, ebenso wie das Zurückspielen der Daten. Nachteile sind die Dauer der Sicherung und der benötigte Platz, da auch Daten gesichert werden, die sich seit dem letzten Backup nicht verändert haben. Ein Komplettbackup wird normalerweise montags oder freitags erstellt.

Anders macht das das **differentielle Backup**. Es sichert nur Daten, die seit dem letzten Komplettbackup geändert wurden. Bei der Rücksicherung muss allerdings die Reihenfolge der Bänder beachtet werden. Im allgemeinen benötigt man hierzu die letzte Komplettsicherung und das letzte Tagesband.

Das **inkrementelle Backup** sichert nur die Dateien, die sich seit dem letzten Inkrementalbackup geändert haben. Bei der Rücksicherung muss ebenfalls streng die Reihenfolge beachtet werden, dafür ist bei der Sicherung noch weniger Platz und Zeit als beim Differentialbackup erforderlich.

Sicherungsprogramme unterstützen normalerweise den Einsatz so genannter **Dump-Level**, die mittels einer Zahl die zu sichernden Dateien angeben. Auf dem Level *n* werden also alle Dateien gesichert, die seit der Sicherung *n - 1* verändert wurden. Vorteil ist wieder die Zeit- und Platzersparnis, Nachteil ist die aufwändigere Rücksicherung.

Man kann aber auch die Methoden mischen. So ist es beispielsweise praktikabel, montags eine Komplettsicherung mit Level 0 durchzuführen, und werktags mit Level 1 die Arbeitsdaten der Woche zu sichern. Da es auch am Wochenende zu Arbeitseinsätzen kommen kann, könnten am Sonnabend und Sonntag mit Level 2 die Tagesdaten gesichert werden.

Beispiel für eine Datensicherungsstrategie

Das Beispiel betrachtet einen einfachen Rechner mit Leveln.

- Am Montag existiert 1 GByte an Daten.
- Jeden Tag kommen 100 MByte hinzu.
- Am Sonntag liegen 1,6 GByte Daten vor.
- Sicherungen erfolgen in der Nacht, vor Arbeitsbeginn.
- Sonntagmittag sei eine Rücksicherung vom Band notwendig.

Tabelle 1 zeigt die verwendeten Dump-Level. Die Tabellen 2, 3 und 4 zeigen die anfallende und gesicherte Datenmenge.

	Mo	Di	Mi	Do	Fr	Sa	So
Komplett	0	0	0	0	0	0	0
Differentiell	0	1	1	1	1	1	1
Inkrementell	0	1	2	3	4	5	6

Tabelle 1: Wochensicherung mit Dump-Leveln

	Mo	Di	Mi	Do	Fr	Sa	So
Komplett	0	0	0	0	0	0	0
Tagesration	1 GByte	1,1 GByte	1,2 GByte	1,3 GByte	1,4 GByte	1,5 GByte	1,6 GByte
Gesamt	1 GByte	2,1 GByte	3,3 GByte	4,6 GByte	6,0 GByte	7,5 GByte	**9,1 GByte**

Tabelle 2: Datenmenge einer Komplettsicherung

Die Komplettsicherung sichert jeden Tag alle Daten. Die Rücksicherung erfolgt vom letzten Band, also dem Sonntagsband.

	Mo	Di	Mi	Do	Fr	Sa	So
Differentiell	0	1	2	3	4	5	6
Tagesration	1 GByte	0,1 GByte	0,1 GByte	0,1 GByte	0,1 GByte	0,1 GByte	0,1 GByte
Gesamt	1 GByte	1,1 GByte	1,2 GByte	1,3 GByte	1,4 GByte	1,5 GByte	**1,6 GByte**

Tabelle 3: Datenmenge einer Differentialsicherung

Differentiell sichert man jeden Montag alle Daten und dienstags bis freitags auf Level 1. Die Rücksicherung erfolgt erst mit dem letzten Montagsband und anschließend dem letzten Tagesband.

	Mo	Di	Mi	Do	Fr	Sa	So
Inkrementell	0	1	1	1	1	1	1
Tagesration	1 GByte	0,1 GByte	0,2 GByte	0,3 GByte	0,4 GByte	0,5 GByte	0,6 GByte
Gesamt	1 GByte	1,1 GByte	1,3 GByte	1,6 GByte	2,0 GByte	2,5 GByte	**3,1 GByte**

Tabelle 4: Datenmenge einer Inkrementalsicherung

Die inkrementelle Sicherung greift sich jeden Montag alle Daten und wochentags die inkrementellen, also nur Veränderungen zum Vortag. Die Rücksicherung erfolgt in der Reihenfolge der Bänder von Montag bis Sonntag.

Für Diskjockeys: Türme von Hanoi

Der Nachteil der bisherigen Strategie ist, dass die Dateien in der Regel nur auf einem einzigen Band liegen, was bei Verlust des Bandes dem Verlust aller Daten eines Tages und der darauffolgenden Tage gleichkommt. Es gibt eine weitere, platzsparende Strategie, die an das Knobelspiel *Türme von Hanoi* angelehnt ist. Diese in Tabelle 5 gezeigte Strategie verteilt einzelne Dateien auf mehrere Bänder. Für einen gesamten Monat lässt sich der Algorithmus mit Level-1-Sicherungen an den folgenden Montagen erweitern. Vorteil dieser Strategie ist, dass die Dateien auf mehr als einem Medium liegen, ohne jedesmal alles sichern zu müssen. Die Verteilung der Dateien auf den Bändern illustriert Tabelle 6.

Mo	Di	Mi	Do	Fr	Sa	So
0	3	2	5	4	7	6
1	3	2	5	4	7	6
1	3	2	5	4	7	6
1	3	2	5	4	7	6

Tabelle 5: Türme-von-Hanoi-Algorithmus

Wochentag	Daten auf dem Tagesband
1. Montag	Montag
Dienstag	Montag & Dienstag
Mittwoch	Montag & Dienstag & Mittwoch
Donnerstag	Mittwoch & Donnerstag
Freitag	Mittwoch & Donnerstag & Freitag
Samstag	Freitag & Samstag
Sonntag	Freitag & Sonnabend & Sonntag
2. Montag	1. Montag – 2. Montag
2. Dienstag	2. Montag & 2. Dienstag
[...]	

Tabelle 6: Datenverteilung im Hanoi-Algorithmus

Über Stefan

Stefan Schumacher ist geschäftsführender Direktor des Magdeburger Instituts für Sicherheitsforschung und gibt zusammen mit Jan W. Meine das Magdeburger Journal zur Sicherheitsforschung heraus. Er befasst sich seit knapp 20 Jahren als Hacker mit Fragen der Informations- und Unternehmenssicherheit und erforscht Sicherheitsfragen aus pädagogisch-psychologischer Sicht. Seine Forschungsergebnisse stellt er auf Fachkongressen und in Publikationen vor. Seine Schwerpunkte liegen auf Social Engineering, Security Awareness, Organisationssicherheit, internationale Cyber-Security und Mensch-Maschine-Interaktion.

Geschichtsstunde II
1975 - 1990: Unix erobert die Welt

Unix' Jugend- und Mannjahre gehen fließend in das Internetzeitalter über. Jüngere mögen sich die Welt ohne Internet nicht vorstellen können, und tatsächlich tauschte man Software damals nicht per Downloadklick. Sondern auf Magnetbändern. Doch man tauschte. Und wie.

von **Jürgen Plate**

One of the advantages of using UNIX to teach an operating system's course:
the sources and documentation will easily fit into a student's briefcase.

John Lions, University of New South Wales, um 1976 über Version 6

Im ersten Teil dieser Artikelserie arbeiteten wir uns bis zum Jahr 1974 vor, als die Berkeley-Universität in Kalifornien eine Unix-Kopie bekam (Abbildung 1). Das hatte weitreichende Folgen. Die Uni erhielt die Distribution überhaupt erst deswegen, weil das amerikanische Justizministerium und die Bell Laboratories 1956 in einem so genannten Consent Decree rechtsverbindlich übereingekommen waren, dass die Telko-Monopolistin *Ma Bell* nicht auch noch außerhalb des Telefonmarktes kommerziell agieren durfte. *Ma Bell* kommt von Mother Bell, eine umgangssprachliche Referenz auf die Bell Laboratories. Wegen dieser Übereinkunft gab die Firma Unix-Quellen und -Binaries zum geringen Preis an Universitäten ab, und sie sollte bis 1979 gültig sein.

Abbildung 1: Geburtsstätte von BSD: die Universität Kalifornien in Berkeley. (©Steve McConnell, UC Berkeley.

1975 kehrte Ken Thompson für ein Jahr als Gastprofessor an die Universität Berkeley zurück, an der er zuvor studiert hatte. Dort entwickelten gerade die Studenten Chuck Haley und Bill Joy den Texteditor *em* (editor for mortals) zu dem Editor *ex* (extended editor) weiter. Die beiden begannen, Unix zu einer eigenen Version weiterzuprogrammieren. 1977 veröffentlichten sie ihre Berkeley-Erweiterungen als *Berkeley Software Distribution* (BSD).

Etwas später wurde *ex* zu einem Bildschirmeditor, der noch heute unter dem Namen *vi* (für „visual") die Unix-Anfänger in den Wahnsinn treibt. Trotzdem ist er – vielfach verbessert – immer noch der am häufigsten verwendete Unix-Editor (ruft da irgendwer „emacs"?). Bildschirmorientierte Editoren waren damals revolutionär. Bill Joy beschrieb die Sensation so:

> When using *vi*, the screen of your terminal acts as a window into the file which you are editing. Changes which you make to the file are reflected in what you see.

1978 erfolgte die zweite Release des BSD-Unix namens *2BSD*, die den *vi* enthielt. Im gleichen Jahr gründeten Doug und Larry Michels das erste Unternehmen, das sich hauptsächlich mit Unix befasste: *Santa Cruz Operations* (SCO) war ein Dienstleister, der Großrechner mit diesem System installierte und wartete. Später wurde SCO als Distributor einer Unix-Version für PC-Systeme mit i386er Prozessoren bekannt. Der gute Name ging verloren, als SCO von Caldera gekauft wurde. Dazu später mehr.

Die erste Unix User Group

Nicht nur in Santa Cruz und in Berkeley, auch bei AT&T lief die Entwicklung weiter. 1975 gab die Firma die sechste Edition heraus und begann, über

das Tochterunternehmen Western Electric Lizenzen an Unternehmen und Behörden zu verkaufen. Mike Lesk schrieb seine *Portable C Library*, eine Reihe mit Ein- und Ausgaberoutinen, die sich auf nahezu jeder Maschine mit C-Compiler implementieren ließen. Dies war ein großer Schritt in Richtung portablen C-Codes. Dennis Ritchie überarbeitete die Bibliothek später und nannte sie *Standard I/O Library*, und die heute jedem C-Programmierer als *stdio* geläufig ist.

In diese Zeit fielen die ersten Unix-Anwendertreffen. Das allererste Treffen in New York sah gerade einmal um die 40 Teilnehmer. Weitere Treffen erfolgten alle zwei Jahre. Ihr Ablauf war extrem locker: Wer etwas präsentieren wollte, hob die Hand und legte los. Man tauschte fleißig Bugreports, Fixes und neue Software. Üblicherweise hatte jeder Teilnehmer mindestens zwei Magnetbänder dabei – eines mit neuen Sachen für die anderen, und ein leeres, um die tollen Dinge der anderen mit nach Hause zu nehmen.

Die BSD-Versionen *3BSD* und *4BSD* dienten als Grundlage für die Entwicklung von TCPIP – und damit für das Internet. 1977 verkaufte das Unternehmen ISC (Interactive Systems Corporation) aus Santa Monica als erste Firma die siebte Edition von Unix an Endverbraucher. Nun war Unix ein normales Produkt mit entsprechendem Support.

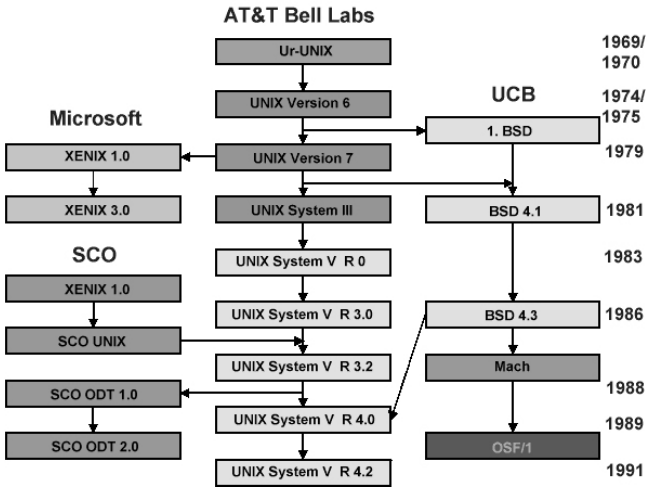


Abbildung 2: Übersicht der Versionsentwicklung von Unix.

1978 brachte die Digital Equipment Corporation (DEC) den Nachfolger ihres PDP-11-Systems auf den Markt: die VAX-Rechner. Obwohl sie mit dem DEC-Betriebssystem VMS ausgeliefert wurden, ließ sich Unix aufgrund der PDP-11-Kompatibilität leicht darauf portieren. Vor allem die Version 4.2 von BSD, die im Jahr 1983 erschien, sollte auf VAX-Maschinen sehr beliebt werden. Kommerzielle Unix-Versionen von Sun Microsystems (SunOS, später Solaris), von DEC (Ultrix) und anderen folgten. Neben AT&T und Berkely betraten damit weitere Akteure die Bühne. Abbildung 2 zeigt eine Übersicht.

Auch Universitätsprojekte migrierten in Berkeley zu Unix. Ein Beispiel dafür war das Ingres-Datenbankprojekt, für das man 1974 sogar einen zusätzlichen Rechner anschaffte. Ingres war so erfolgreich, das es kommerziell vertrieben wurde. Aus diesem Projekt entstand der Nachfolger Postgres und später das heute weiterhin sehr lebendige PostgreSQL-Datenbanksystem.

Die Bildschirmterminals kommen

Inzwischen gab es anstelle der ratternden Teletype-Geräte immer mehr Bildschirmterminals von unterschiedlichen Herstellern. Jedes arbeitete mit anderen Steuersequenzen etwa für Cursorbewegung und Darstellung. Legendäre Vertreter sind zum Beispiel das ADM-3A oder Televideo VT100. Damit ein existierendes Programm sofort auf einem neuen Terminal laufen konnte, entwickelte der BSD-Vater Bill Joy einen Interpreter, der die Charakteristik des angeschlossenen Terminals per Konfigurationseintrag in einer Datei entgegennimmt. Damit ermöglichte er den Programmen, über eine fest definierte Schnittstelle beliebige Terminals anzusprechen.

Dieser Interpreter namens *termcap*, später *terminfo*, ist noch heute Bestandteil von Unix, obwohl er längst nicht mehr die Rolle spielt wie in den 1970er und 80er Jahren. Aber auch heute noch löscht auf einer Linux-Konsole zum Beispiel das Kommando *tput clear* den Bildschirm oder positioniert den Cursor. Inzwischen nimmt die Bedeutung wieder zu. Immer mehr so genannte Appliances oder auch eingebettete Systeme basieren auf BSD oder Linux. Die Ansteuerung von modernen intelligenten LCD- oder OLED-Bildschirmchen dieser Geräte regelt oft schon ein neuer Eintrag in der Terminfo-Datenbank, ohne einen extra Treiber zu benötigen.

1979, nach der Freigabe von Unix V7, verlor das Consent Decree seine Gültigkeit. Jetzt begann AT&T selbst, Unix kommerziell zu nutzen. Ab diesem Zeitpunkt mussten die Nutzer also eine Lizenz erwerben. Im gleichen Jahr lizenzierte ein damals recht unbekanntes Unternehmen Unix und seine Portierungen auf Intels 8086-Prozessor und erschien 1980 unter dem Namen *Xenix OS*: einer der ersten kommerziellen Unix-Klone. Die Firma hieß Microsoft.

Obwohl Microsoft bereits 1987 Xenix an SCO verkaufte, hat dieses kurze Intermezzo Folgen. Die Unix-Erfahrungen bilden den Hintergrund des DOS-2.0-Dateisystems und die Grundlage für Windows NT. Auch die TCPIP-Implementierung von NT und anderen Windows-Versionen erinnert stark an BSD-Unix.

Durchbruch mit BSD 4.2

Im Frühling 1982 verließ Bill Joy die Berkeley-Universität und wechselte zu Sun Microsystems, wo er bis 2003 blieb. Im Gepäck hatte er ein Band mit der neuesten BSD-Version. Damit erhält der alte Slogan von Sun – „The Joy of UNIX"– eine ganz neue Bedeutung. Version 4.2 von 1983 erfuhr viele Verbesserungen und Neuerungen. Erstmals gab es die von dem australischen Netzwerk-Pionier Robert Elz entwickelten Disk Quotas, was bei den damaligen Platten mit ihrer aus heutiger Sicht winzigen Speicherkapazität dringend notwendig war. Auch wurde die gesamte Dokumentation renoviert. Hervorzuheben ist schließlich, dass *4.2BSD* das erste netzwerkfähige Unix war. Das dürfte wohl seinen Siegeszug als Internet-Serverplattform mitbegründet haben.

Die Version war ein voller Erfolg. Schon nach eineinhalb Jahren war die Tausendermarke der Installationen überschritten. Schon bald gab es mehr BSD-Systeme als die kommerzielle Version V von AT&T. Die Freude währte jedoch nicht lange, denn schon bald zog System V nach, auch auf der Netzwerkseite. Doch es war BSD, das in den Internet-Vorgänger *ARPANET* Einzug hielt. Zu dessen Zweck wurde unter Leitung von Professor Bob Fabry in Berkeley extra die *Computer Systems Research Group* (CSRG) gegründet. Bereits 1980 hatte diese ein E-Mail-System vorzuweisen. Die erste E-Mail verschickte aber schon neun Jahre zuvor Ray Tomlinson, der uns auch den Klammeraffen in Mailadressen beschert hat, über das Arpanet.

BSD-Unix wurde immer als Sourcecode verteilt – und hat so nicht nur viele Unix-Varianten oder Windows beeinflusst, sondern auch Linux. Die Verbreitung als Quellcode ermöglicht seither nicht nur die Portierung auf andere Hardware, sondern erleichtert auch die Weiterentwicklung, wie Source-Reviews, schnelle Bugfixes, Verbesserungen und Erweiterungen immer wieder bezeugen. Der Nachteil der Source-Distribution war jedoch, dass bis in das Jahr 1989 hinein jeder, der ein BSD-Unix nutzen wollte, zuerst eine AT&T-Lizenz erwerben musste, weil BSD-Unix ja auf einem Quelltext von AT&T aufsetzte.

Religiöse Sitten und Bräuche

Für viele Benutzer war Unix mehr als nur als ein Betriebssystem. Es war eine Art Weltanschauung. Aus diesem Grund gab und gibt es immer noch Dispute bis hin zum Kreuzzug, die teilweise bis in die Unix-Anfänge zurückreichen. Einer der Glaubenskrieg war der zwischen BSD und System V in der Release 4. Ein weiterer, der immer noch im Gange ist, heißt *vi* gegen *emacs* – diese Gegner sind sich immerhin darin einig, niemals einen grafisch orientierten Editor anzufassen. In den 1980er Jahren ging es dann weiter mit „X Window System vs. Konsole", bei der Grafikfraktion heißt es „KDE kontra Gnome", und beide formieren sich gegen XFCE. So richtig fetzen sich aber die Linux-Kernelentwickler. Wer will, kann mal zählen, wie oft das F-Wort im Kernelquelltext von Linux auftaucht.

Trotzdem wurzelt die Unix-Kultur in dem Brauch, untereinander C-Quellcode zu tauschen, ihn gemeinsam zu verbessern oder zu erweitern, und diese Veränderungen weiter zu verteilen. In vielen Fällen war die Dokumentation eines Programm lediglich der kommentierte Quellcode. Doch jeder, der einen C-Compiler besaß, konnte diese Programme implementieren.

In den Jugendjahren von Unix war 1983 ein bedeutendes Jahr, auch wenn damals wohl niemand die Tragweite erfasste. In diesem Jahr gründete Richard Stallman das Projekt *GNU* mit dem Ziel, einen komplett freien Unix-Klon zu entwickeln. Daher stammt das rekursive Akronym, das für „GNU is Not Unix"steht. Dieses Projekt schrieb alle Unix-Systemutilities neu und verbesserte sie oft, etwa im Fall von *less* als Erweiterung von *more*. Das Projekt brachte neben vielen Tools auch effiziente Compiler für die Sprachen C, C++, Fortran, Pascal und Lisp hervor. Inzwischen hat es neben den Compilern für Desktop- und Serversysteme sogar Crosscompiler für Mikrocontroller hervorgebracht, zum Beispiel für Atmel-Mikrocontroller oder die ARM-Plattform.

Die GNU-Tools trugen maßgeblich zum Aufstieg des neuen Unix-Klons namens *Linux* bei. Auch der legendäre, mit einem Lisp-Dialekt programmierbare Editor Emacs gehört zu den GNU-Werkzeugen, den es bereits vor Linux für das von Stallman im MIT benutzte ITS (Incompatible Timesharing System) gab. Außerdem existierte bereits die in Text gegossene Unix-Kultur, die Ge-

neral Public License (GPL), die untrennbar zur Open-Source-Bewegung gehört.

1990 begann Stallman im GNU-Projekt mit der Entwicklung eines Mikrokernels namens *The Hurd* auf Basis des *Mach*-Kernels von der Carnegie-Mellon-Universität Pennsylvania. Seine Entwicklung ist nicht abgeschlossen: Heute haben seine Versionsnummern immer noch eine dicke Null vor dem Punkt. Es existiert immerhin die Hurd-Distribution *Debian GNU/Hurd*, die jedoch noch nicht sehr produktiv ist.

The Clone Wars

1986 war abzusehen, dass die VAX-Architektur keine Zukunft hatte. Um die Portierung von BSD auf andere Plattformen zu erleichtern, begann man in Berkeley, den Quellcode zu bereinigen. Immer noch war für den kommerziellen Einsatz von BSD eine Lizenz von AT&T nötig. Also trennte man die Eigenentwicklungen in BSD vom AT&T-Unix und veröffentlichte sie 1988 unter dem Namen *Networking Release 1* mit einer einfachen und liberalen Lizenz.

Abbildung 3: Den allseits bekannten BSD-Daemon entwarf der BSD-Pionier und heutige FreeBSD-ler Marshall Kirk McKusick. (©Marshall Kirk McKusick, 1988. Alle Rechte vorbehalten.)

Aus dieser Zeit stammt der BSD-Daemon, das Maskottchen des BSD-Projekts (Abbildung 3). Es mussten hunderte Werkzeuge und Libraries neu implementiert werden, um aus der Networking Release 1 wieder ein lauffähiges Unix-System zu machen. Einige Dateien des Kerns konnten die Entwickler zunächst dennoch nicht ersetzen. Das weiterhin unvollständige System wurde 1991 als *Networking Release 2* freigegeben.

Erst Ende 1991 hatte der Programmierer Bill Jolitz schließlich die fehlenden Dateien ergänzt und veröffentlichte das vervollständigte System als *386/BSD*. Da Jolitz das Projekt nicht allein weiterführen konnte, gründeten einige Benutzer das Projekt *NetBSD*, um die Entwicklung ihres BSD-basierten Unix' fortzuführen. Kurz danach formte sich auch das Projekt *FreeBSD*. Mitte der 1990er Jahre entstand außerdem noch *OpenBSD*, das ein besonders sicheres und einfach zu handhabendes System sein sollte. Schließlich entstand auch noch das Unternehmen Berkeley Software Design, Inc. (BSDi), das sich mit Entwicklung und Vertrieb eines kommerziellen BSD-Systems auf Grundlage von *386/BSD* befasste.

Leider funkte AT&T dazwischen, sodass keine der Gruppen ihr Betriebssystem *Unix* nennen durfte. Die folgenden, endlosen Rechtsstreitigkeiten wurden bekannt als *The BSD Wars*. Die letztliche Einigung erbrachte, dass *4.4BSD-Lite* als offiziell einzig legale BSD-Release erschien (und eben nicht Unix hieß).

Der PC und SCO

Während sich 1981 die Unix-Landschaft zu formieren begann und Microsoft seine ersten Betriebssystem-Schritte unternahm, schickte IBM den so genannten Personal Computer in ein siegreiches Rennen. Er bestand aus bunt zusammengewürfelten Komponenten: Der Prozessor 8088 mit 4,77 MHz Taktfrequenz kam von Intel, der Grafikcontroller 6845 von Motorola (heute Freescale), der I/O-Baustein 8255 stammt wieder von Intel, die 5,25-Zoll-Speicherdisketten von gottweißwo, und die Druckerschnittstelle wurde aus einigen Logikbausteinen zusammengebastelt. Am 12. August 1981 erblickte der IBM-PC das Licht des Weltmarkts und kostete je nach Ausstattung zwischen 3000 und 6000 Dollar.

Der große Vorteil bestand darin, dass IBM keine eigenen Chips einbaute. Praktisch jeder konnte den Computer nachgebauen. Nur das grundlegende Ein-/Ausgabe-System (Basic I/O System, BIOS) war von Interessenten aus Copyright-Gründen neu zu schreiben. Der IBM-PC wurde ausschließlich mit dem Betriebssystem PC-DOS vertrieben, und zwar bis 1995. IBM hatte es von Microsoft lizenziert. Microsoft wiederum vertrieb das Betriebssystem MS-DOS nur für IBM-PC-kompatible Computer. Aber: Mit dem PC und sei-

nen Nachfolgern gab es auch eine neue Plattform für Unix.

Nachdem sich die Santa Cruz Operation mit Xenix für die bis 1985 erschienenen 286er und 386er Intel-Prozessoren große Marktanteile im PC-Unix-Sektor gesichert hatte, stellte sie im Jahr 1989 die erste Version von *SCO UNIX* vor. Dies war das erste von AT&T lizensierte Unix-Derivat und durfte damit – im Gegensatz zu BSD – den Markennamen *UNIX* in der Produktbezeichnung führen. Es lief auf i386-PCs und basierte auf dem System V Release 3.2. *SCO UNIX* war robust, und viele Hardware-Hersteller unterstützten es. Bis 1995 lieferte SCO sein Unix einzeln als Runtime-System, sowie in den Produkten OpenServer und OpenDesktop mit integrierten Netzwerkprodukten. Für die Anbindung an Netzwerke standen SCO IPX/SPX, SCO TCP/IP und SCO NFS zur Verfügung.

Die heutige *SCO Group* hat übrigens nichts mit der hier erwähnten Firma SCO zu tun. Es handelt sich vielmehr um eine ziemlich glücklos operierende Novell-Ausgründung namens Caldera, die in den 1990er Jahren SCO gekauft und den Namen übernommen hat. Inzwischen ist auch die SCO Group am Rande der Liquidation und hat sich einen zweifelhaften Ruf dadurch erworben, dass sie gegen diverse Linux-Unternehmen erfolglose Copyright-Klagen führt.

Exoten der Unix-Landschaft

Der Reigen der Unix-Anbieter wäre unvollständig, wenn einige Exoten ungenannt blieben. Dazu gehört *MKS Inc* (Mortice Kern Systems). Einige Studenten der University of Waterloo gründeten MKS 1984 als Beratungsunternehmen. Das erste Produkt von MKS war 1985 das MKS Toolkit, eine Sammmlung von Unix-Kommandos für MS-DOS. So bekam der DOS-User zumindest eine Shell und all die kleinen Tools, die wir an Unix so schätzen (mehr oder weniger alles, was bei Unix in `/bin` und `/usr/bin` zu finden war). Später wandte sich MKS anderen Tools zu, aber das MKS Toolkit ist als Entwickler-Plattform auch heute noch für Windows erhältlich.

Ein echtes Unix-artiges Betriebssystem für PC-kompatible Computer stellte die Mark Williams Company erstmals 1983 unter dem Namen *Coherent* vor. Es lief später auf den meisten PCs mit i286-, i386- oder i486-Prozessoren und war mit einem Preis von umgerechnet etwa 100 DM eine höchst preiswerte Alternative zum SCO-System. Geradezu überwältigend war die gedruckte Dokumentation aus Tutorien und Manual-Pages. Aber die Company vermarktete erst Version 3.0 von 1990 intensiv. Auch für Entwickler war Coherent durchaus interessant, waren doch *vi*, *make* und der C-Compiler enthalten. Die Mark Williams Company starb 1995, nicht zuletzt, weil sein *Coherent* keine Netzwerke unterstützte und das aufstrebende Linux kostenlos war. Die letzte Version von Coherent war 4.2.

Einem weiteren Spezialfall, *Cygwin*, widmet sich unter anderem der dritte Teil.

Chronik

- 1975 Unix Version 6, Portable C Library
- 1977 Berkeley Software Distribution (BSD), UNIX Version 7 von AT&T
- 1978 2BSD mit *vi*, VAX-Rechner von DEC
- 1980 Xenix OS von Microsoft
- 1982 4.2BSD
- 1983 GNU-Projekt (GNU is not Unix)
- 1988 Networking Release 1
- 1991 Networking Release 2

Das Letzte

Bill Joy wollte den *vi* tatsächlich einmal so erweitern, dass er mehrere Textfenster unterstützt. Während der Programmierarbeiten streikte das Bandgerät – der Quellcode war weg und es gab kein Backup. Bill ließ sich nicht entmutigen und machte weiter – bis eines Tages auch die Festplatte den Geist aufgab. Wie das Schicksal so spielt, gab es auch diesmal weder Backup noch ein aktuelles Listing. Daraufhin stellte Bill nur noch das Handbuch der fensterlosen Version fertig und es blieb beim *vi*, wie wir ihn heute kennen und schätzen.

Das Bild des BSD-Daemons hat sein Schöpfer Marshall Kirk McKusick für private und nichtkommerzielle Nutzung frei gegeben, und zwar „innerhalb der Grenzen des guten Geschmacks“. Als Beispiel für schlechten Geschmack nennt er die Abbildung eines Daemons, der das Solaris-Logo anzündet. Der Künstler bietet nach Rücksprache Lizenzen für T-Shirt-Druck oder ähnliche Vorhaben an. Einzelheiten enthält die Webseite http://www.mckusick.com/beastie/mainpage/copyright.html.

Literatur

Dennis Ritchie, The Evolution of the Unix Time-sharing System:
http://cm.bell-labs.com/cm/cs/who/dmr/hist.html

The History of AT&T:
http://www.corp.att.com/history/

Marshall Kirk McKusick, Twenty Years of Berkeley Unix. From AT&T-Owned to Freely Redistributable:
http://www.oreilly.com/catalog/opensources/book/kirkmck.html

Eric S. Raymond, The Art of UNIX Programming:
http://www.faqs.org/docs/artu/

Eric S. Raymond, A Brief History of Hackerdom:
http://www.tuxedo.org/~esr/writings/hacker-history/

Free Software Foundation, Overview of the GNU Project:
http://www.gnu.org/gnu/gnu-history.html

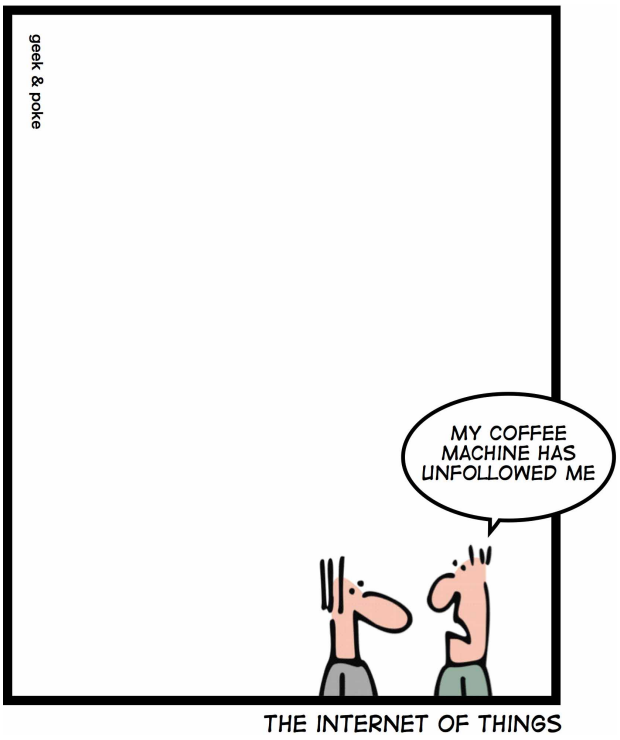
Richard Stallman, The GNU Operating System and the Free Software Movement:
http://www.oreilly.com/catalog/opensources/book/stallman.html

Die Geschichte von BSD-Unix:
https://www.bsdwiki.de/Spezial:Verweisliste/Geschichte?title=Geschichte

SCO vs. Linux, Die unendliche Geschichte:
http://www.heise.de/ct/artikel/SCO-vs-Linux-Die-unendliche-Geschichte-302076.html

MKS-Toolkit und Kommandoliste:
http://www.mkssoftware.com/products/tk/commands.asp

C. Barks: Junior Woodchuck's Guidebook and Reservoir of Inexhaustible Knowledge, Duckburg Press Inc.



Über Jürgen

Jürgen Plate ist Professor für Elektro- und Informationstechnik an der Hochschule München. Er beschäftigt sich seit 1980 mit Datenfernübertragung und war, bevor der Internetanschluss für Privatpersonen möglich wurde, in der Mailboxszene aktiv. Unter anderem hat er eine der ersten öffentlichen Mailboxen – TEDAS der mc-Redaktion – programmiert und 1984 in Betrieb genommen.

Sightseeing für Technikfans
Buchbesprechung: Geek-Atlas

Der Geek-Atlas ist eine Art alternativer Reiseführer, nicht nur rein geografisch gesehen, sondern auch, indem er allerlei Wissenslandschaften vorstellt.

von **Franz Krojer**

Wer nicht auszieht,
kommt nicht heim.

Deutsches Sprichwort

Mit den Geeks und Nerds verhält es sich ähnlich wie früher mit Hackern und Crackern: Sie sind plötzlich in vieler Munde. Worin sie sich genau unterscheiden, wissen die meisten nur vage, während andere es genau zu wissen glauben. Ich versuche das mal so: Ein Geek – auf gut deutsch ein Geck – ist ein im Internet-Zeitalter aufgewachsener Technik- und Wissenschaftsbegeisterter mit einem gewissen sozialen EQ, während ein Nerd vorm Computer emotional und intellektuell verkümmert und noch stolz ist auf seine Fachidiotie.

Die Kluft zwischen der schöngeistig-sanften und der technisch-harten Kultur, wie sie Charles Percy Snow vor rund 50 Jahren in seiner Zwei-Kulturen-These diagnostizierte, scheint mir nach wie vor ähnlich breit und tief zu sein. Wer von Kultur redet, findet auch der Geek-Atlas-Autor in seiner Einführung, bezieht das meistens nur auf Dichter, Maler oder Schriftsteller. Seine Erfahrung: „Rufen Sie ein beliebiges Fremdenverkehrsbüro irgendwo auf der Welt an und fragen Sie nach naturwissenschaftlichen, mathematischen oder technischen Sehenswürdigkeiten. Sie werden ein langes Schweigen ernten, oder eine kurze Liste der einschlägigen Wissenschaftsmuseen erhalten.“(Seite XI)

Neue-Welt-lastig

Der Geek-Atlas beschreibt 128 wissenschaftlich und technisch interessante Orte, sofern diese mehr als nur eine Gedenktafel oder ein IMAX-Kino vorzuweisen haben. Er beschreibt bevorzugt die Wirkungsstätten von Naturwissenschaftlern, Mathematikern und Ingenieuren. Zu jedem Museum, Technikzentrum, Friedhof und jeder Gedächtnisstätte gibt es einen Kasten, der auf ein bis zwei Seiten ein wissenschaftlich-technisches Thema erörtert und dazu ein paar praktische Hinweise reicht, wie Anfahrt, Internetauftritt oder Öffnungszeiten. Ein ausführlicher Sach- und Personenindex erleichtert die Orientierung.

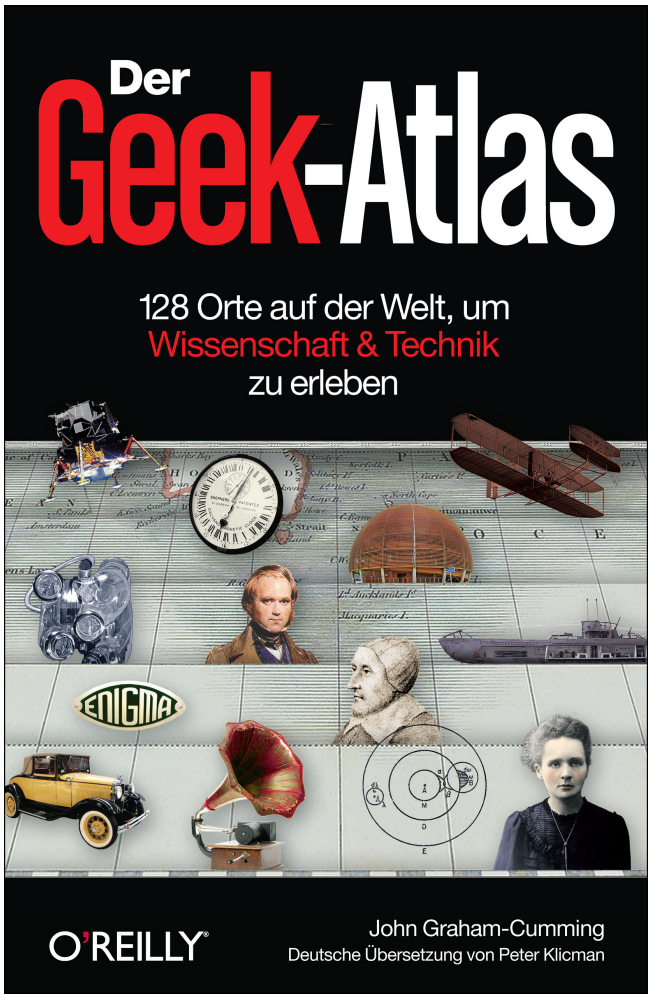


Abbildung 1: John Graham-Cumming:
Der Geek-Atlas. 128 Orte auf der Welt, um Wissenschaft & Technik zu erleben.
KölnO'Reilly 2010.
Print: 592 Seiten, 24,90 Euro, ISBN 978-3-89721-933-5.
E-Book: 20 Euro, ISBN 978-3-89721- 936-6.

Das Buch verspricht vieles, hält auch vieles, aber nicht alles. Von den 128 Orten auf der Welt stammen 91 aus dem Vereinigten Königreich und

den Vereinigten Staaten. Deutschland ist mit fünf Orten noch relativ gut vertreten (Deutsches Museum München; Informationszentrum Peenemünde; Röntgen-Museum Remscheid; Stadtfriedhof Göttingen; Gutenberg-Museum Mainz), Afrika hingegen gar nicht.

Diese starke Englisch-Lastigkeit sehe ich aber nicht nur negativ, ist damit doch auch gezeigt, dass noch ein Potential für weitere Bände besteht. Für den süddeutschen Raum fallen mir spontan das Kepler-Gedächtnishaus in Regensburg und das Rieskrater-Museum in Nördlingen ein. Übrigens: Wer nach dem Sinn der Reihenfolge im Inhaltsverzeichnis fragt – auf Australien folgt Österreich, dann Belgien und Kanada – ahnt, dass das in der Quellsprache sinnvoller aussah: Australia, Austria, ...

Geekiges Gegenlesen

Angesichts der Themenvielfalt hätte es dem Buch gut getan, wenn es von einigen Geeks vorab gelesen worden wäre. Es finden sich immer wieder leicht vermeidbare Fehler und Ungenauigkeiten. So wird von dem 27. Längengrad von Jaipur (Indien) gesprochen (Seite 99), wo es sich doch um den Breitengrad handelt, und eine Seite später steht: „Auf der Erde wird der Breitengrad vom Greenwich-Meridian aus gemessen“, wo es diesmal Längengrad heißen müsste. Oder wenn auf Seite 67 als Einheit für die Lichtgeschwindigkeit „Kmh“angegeben sind – bei einem Wert von um die 300.000 –, statt richtig „kms“: Dann rechne man mir das nicht als kleinliche Nörgelei an, denn ich will ja nur beispielhaft zeigen, welche Arten von Fehlern vorkommen, die umso leichter übersehen werden, je weniger man von einem bestimmten Thema weiß.

Ich gestehe gern, dass ich das Buch mit großem Gewinn gelesen habe. Es hat mir Spaß gemacht, von Dingen, die ich schon wusste, wieder einmal zu hören, und sie nun besser zu verstehen. Von einigen anderen, besser gesagt: nicht wenigen Dingen habe ich sogar zum ersten Mal gehört, oder sie endlich begriffen. Jetzt behaupte ich ganz geckenhaft: Wer es nicht gelesen hat, ist kein echter Geek.

Über Franz

Franz Krojer, geboren 1958, arbeitet am Institut für Informatik an der LMU München als EDV-Systemadministrator. Darüber hinaus beschäftigt er sich mit Astronomie-Geschichte und setzt sich dabei besonders mit der Chronologie-Kritik (und zwar der so genannten mittelalterlichen Phantomzeit) auseinander, wozu von ihm der Differenz-Verlag gegründet wurde.

Volles Programm
Übersicht des FFG 2013

Über 100 Vorschläge erreichten das Programmteam, von denen auch nach kritischer Sichtung ziemlich viele hochklassige übrig blieben. Am ersten Konferenztag (Donnerstag) gibt es daher erstmals einen dritten Track. Das Frühjahrsfachgespräch (FFG) der GUUG findet vom 26.2. bis zum 1.3.2013 an der Fachhochschule Frankfurt am Main statt.

vom **FFG-Team**

Der Mensch entwirft die Pläne im Herzen,
doch vom Herrn kommt die Antwort
auf der Zunge.

Altes Testament, Sprüche 16, 1

Zeit	Dienstag, 26.02.2013
9 - 18 Uhr	Anmeldung
10 - 18 Uhr	Tag 1: Monitoring von IPv6-Netzwerken (Robin Schröder, Timo Altenfeld, Wilhelm Dolle, Christoph Wegener)
10 - 18 Uhr	Tag 1: Plattformübergreifende Dateidienste sicher anbieten (Daniel Kobras, Michael Weiser)
10 - 18 Uhr	OpenNMS: Make the net work :-) (Alexander Finger)
10 - 18 Uhr	Wireshark – Anwendungsmöglicheiten und Erweiterungen (Martin Kaiser)
10 - 18 Uhr	ISO 2700x und IT-Grundschutz mit dem Open-Source-Tool *verinice* (Alexander Koderman)
10 - 18 Uhr	SAP Business Warehouse Accelerator – Linux in RZ-Umfeld (Jochen Hein)

Zeit	Mittwoch, 27.02.2013
9 - 18 Uhr	Anmeldung
10 - 18 Uhr	Tag 2: Monitoring von IPv6-Netzwerken (Robin Schröder, Timo Altenfeld, Wilhelm Dolle, Christoph Wegener)
10 - 18 Uhr	Tag 2: Plattformübergreifende Dateidienste sicher anbieten (Daniel Kobras, Michael Weiser)
10 - 18 Uhr	Monitoring mit Zabbix – erfolgreich und schnell ein- oder umsteigen (Thorsten Kramm)
10 - 18 Uhr	Metasploit Kung Fu für Systemadministratoren (Michael Messner)
10 - 18 Uhr	OS Lifecycle Management in Oracle Solaris 11 (Detlef Drewanz)
10 - 18 Uhr	Wie setze ich neueste IT-Technologie in einer HA-Umgebung um? (Dirk Reuper)
10 - 18 Uhr	*anykey0x* (Tim Becker, Matthias Krauß)
19 - 21 Uhr	GUUG-Mitgliederversammlung

Zeit	Donnerstag, 28.02.2013		
9 - 18 Uhr	Anmeldung		
9.15 - 10.15 Uhr	Keynote		
10.15 - 10.45 Uhr	Kaffeepause		
10.45 - 11.30 Uhr	MySQL HA: Galera in the house (Erkan Yanar)	Ich habe eine WAF – Hilfe, sie loggt! (Christian Bockermann)	PDF Debugging (vorwiegend) auf der Kommandozeile (Kurt Pfeifle)
11.30 - 12.15 Uhr	Oracle Migration – Hürden und Best Practices in einer hochverfügbaren Umgebung (Andrea Held)	Admins schlagen zurück: SSH-Angreifern mit Honeypots über die Schulter schauen (Andreas Bunten et al.)	*strace* für BASH-Versteher (Harald König)
12.15 - 13.45 Uhr	Mittagspause		
13.45 - 14.30 Uhr	Puppet, klar. Und dann? (Thomas Gelf)	DNS Rate Limiting (Matthijs Mekking, Vortrag auf Englisch)	TacNET – Grafische Verwaltung komplexer virtueller KVM-Netze (Ralf Spenneberg)
14.30 - 15.15 Uhr	Ansible – Verteilen, konfigurieren, machen (Philipp Grau)	Home Network Horror Stories (Michael Messner)	Alternativen zur klassischen Virtualisierung (Oliver Rath)
15.15 - 15.45 Uhr	Kaffeepause		
15.45 - 16.30 Uhr	Aktuelle Entwicklungen bei Icinga und ein Ausblick auf Icinga2 (Bernd Erk)	Best of Audit 2012, oder: "IT-Sicherheit? Evaluieren wir gerade.„ (Alexander Koderman)	Sicheres und unabhängiges Datensharing mit ownCloud (Christian Schneemann)
16.30 - 17.15 Uhr	Netzmanagement mit HIRN (Robin Schröder)	IPv6 und Datenschutz – eine technische und juristische Bewertung (C. Wegener, J. Heidrich)	Die eigene Cloud mit OpenStack Folsom (Martin Gerhard Loschwitz)
17.15 - 18 Uhr	System- und Netzwerküberwachung mit Zabbix (Stefan Gazdag)	Die größten IPv6-Marketinglügen hinterfragt (Marc Haber)	Aufbau einer IaaS-Umgebung mit OpenStack (Christian Berendt)
Ab 19 Uhr	Geselliger Abend		

Zeit	Freitag, 01.03.2013	
9 - 14 Uhr	Anmeldung	
9.15 - 10 Uhr	Samba 4.0: Active Directory – und viel mehr (Michael Adam)	Best Practice für stressfreie Mailserver (Peer Heinlein)
10 - 10.45 Uhr	TBA	Strukturiertes Logging mit *rsyslog* (Rainer Gerhards)
10.45 - 11.15 Uhr	Kaffeepause	
11.15 - 12 Uhr	Lustre/ZFS – verteiltes Dateisystem auf neuen Sohlen (Daniel Kobras)	Reboot reloaded – Linux-Kernel patchen ohne Reboot (Udo Seidel)
12 - 12.45 Uhr	Nahtlos skalierbares Storage mit Ceph (Martin Gerhard Loschwitz)	Kernelhänger mit System, oder: Systemüberwachung testen durch Fehlerinjektion (Stefan Seyfried)
12.45 - 14 Uhr	Mittagspause	
14 - 14.45 Uhr	Btrfs und ZFS: eine Gegenüberstellung von Dateisystemen der neuen Generation (L. Grimmer, U. Gräf)	Interrupt-Routinen: Mit Störungen umgehen (Karsten Schulz)
14.45 - 15.30 Uhr	Illumos, SmartOS, OpenIndiana: Morgenröte für OpenSolaris oder Sonnenfinsternis? (Volker A. Brandt)	Sei kein Troll – Über den Umgang mit Flame Wars (Thomas Rose)
15.30 - 16 Uhr	Kaffeepause	
16 - 16.45 Uhr	Web-Performance-Optimierung mit Varnish (Stefan Neufeind)	Erfolgreich selbständig sein in IT-Projekten oder: Wie frei ist ein Freelancer? (Martina Diel)
16.45 - 17.30 Uhr	Managing Loadbalancers on an Enterprise Level (Jan Walzer)	Linux-ARM-Server im Unternehmenseinsatz (Daniel Gollub)

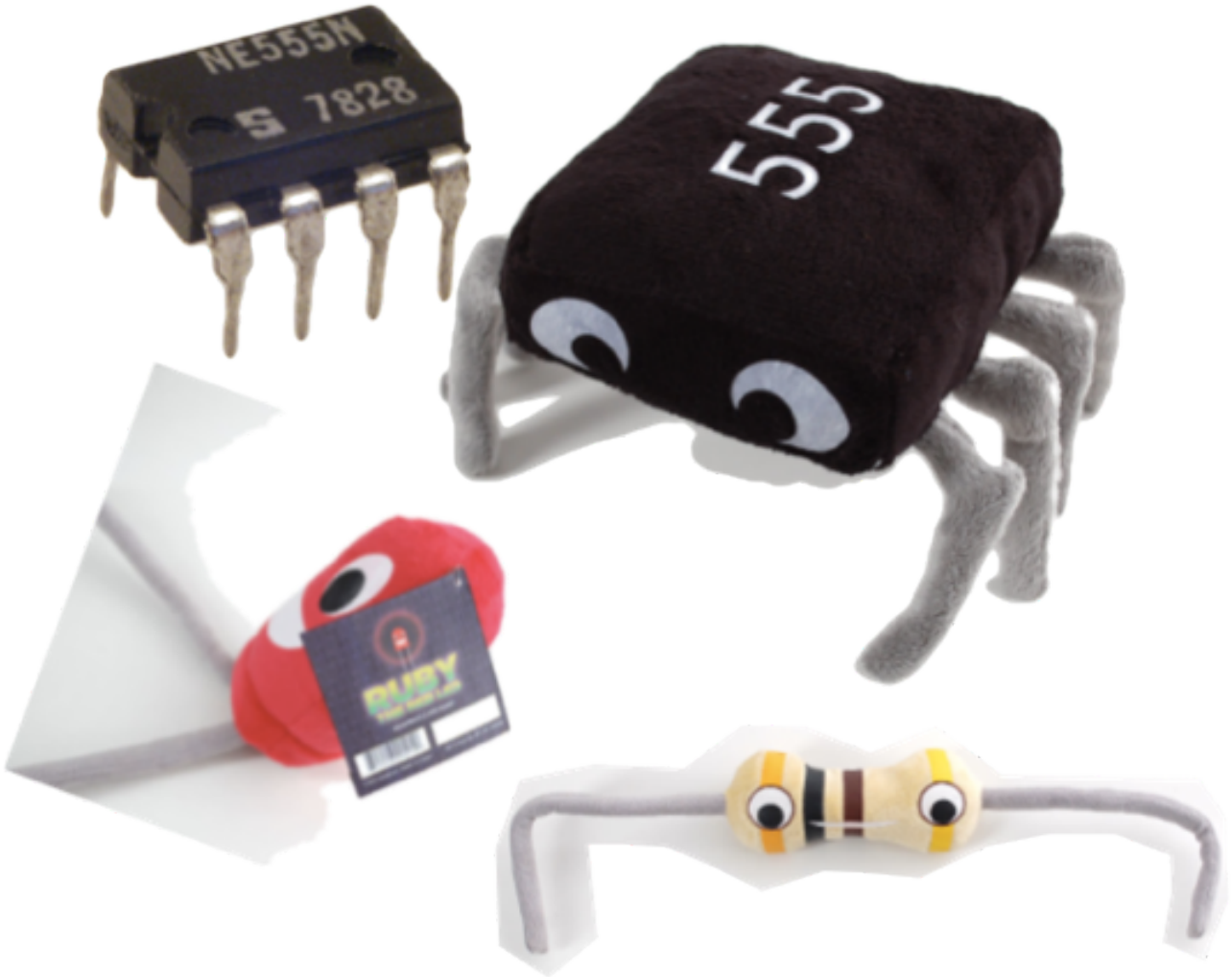
Die aktuellste Version sowie Einzelheiten zu den Vorträgen und Sprechern sind in der Online-Version des Programms auf den GUUG-Webseiten zu finden: http://www.guug.de/veranstaltungen/ffg2013/programm.html .

24 Türchen
Was wir gern unterm Weihnachtsbaum fänden

Einem Aufruf auf GUUG-Mailinglisten und im sozialen Netz folgend, erreichten Tipps der Sorte *Haben will!* die Redaktion. So heißt es wieder: Lassen wir Bilder sprechen.

von **Anika Kehrer**

Knüller: **Plüsch-Elektronik** [1]. Der größte Kandidat misst ungefähr 20 cm x 15 cm x 6 cm und wiegt 90 Gramm. Er heißt *Hans*, denn sein Erfinder – zumindest der Erfinder des Original-Bauteils – heißt Hans Camenzind. *Ruby die LED* ist 30 cm hoch, ansonsten eher schlank, und *Mho der Widerstand* bringt es auch 50 cm in der Breite. Der kleine Preis von 10 US-Dollar pro Software – wo natürlich noch Versandkosten hinzukommen – nützt leider nicht viel: Die Kuscheltiere sind samt und sondern vergriffen. Beim Versender Adafruit kann man aber seine Mailadresse hinterlegen, um bei Eintreffen neuer Objekte informiert zu werden. (Abbildungen: Adafruit.com, Bauteil: stefan506, Wikipedia, CC-by-sa)



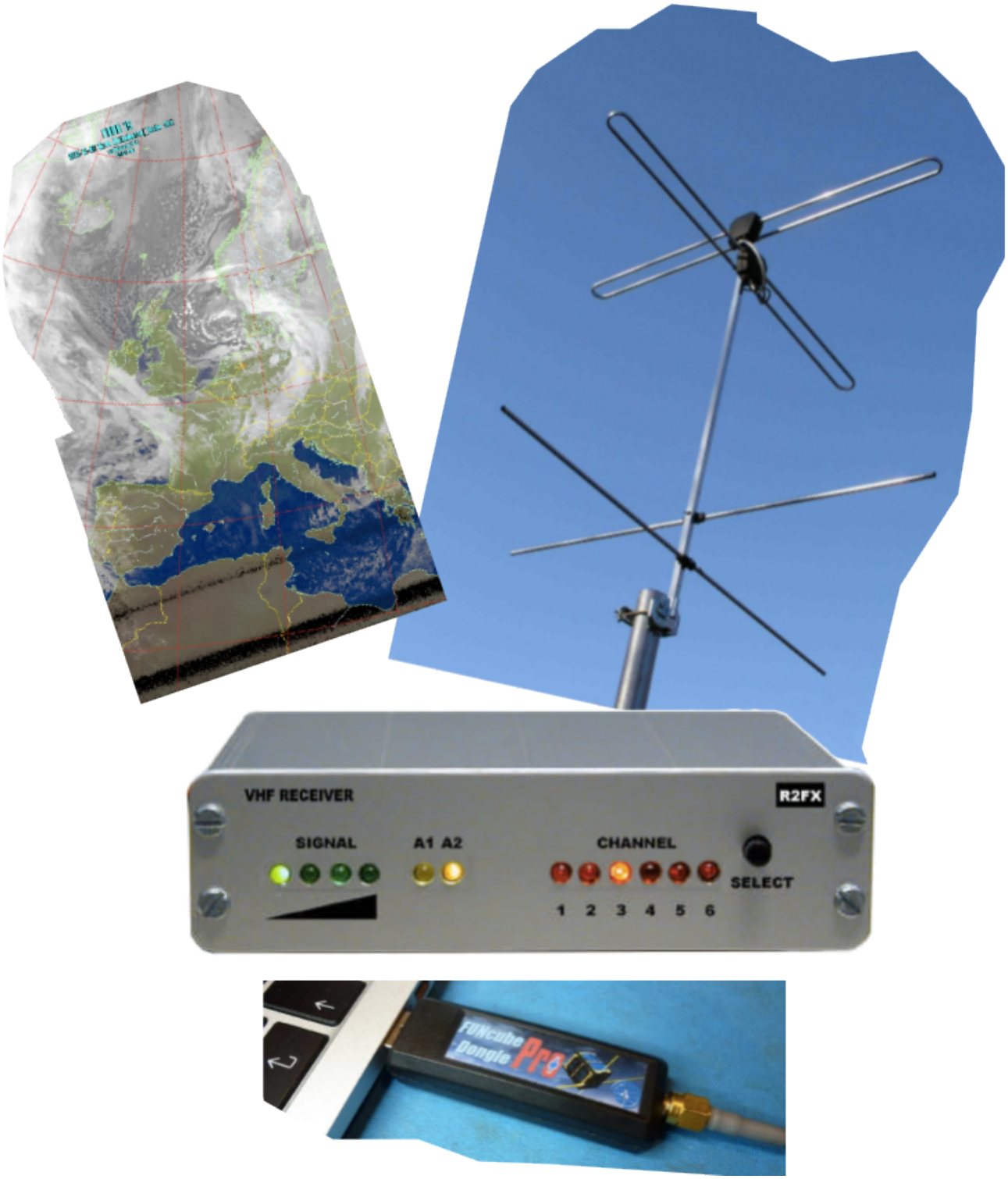
Stefan Lesser empfiehlt Starterkits für **Arduino**: „Seit letztem Wochenende habe ich so ein Board und ein paar I2C-Gimmicks„, berichtet er. “Es macht einfach Spaß, innerhalb von fünf Minuten etwas in Hardware zu realisieren, wo man früher deutlich länger mit dem Lötkolben kämpfte. Die Tutorials sind gut,

es gibt Sensoren, Module und Libraries für so ziemlich jede Anwendung, inklusive Bluetooth, ZigBee und WLAN.„In Deutschland ist zum Beispiel bei der Berliner Open-Hardware-Initiative *Fritzing* für 66 Euro ein Starterkit mitsamt Kabeln, LEDs und Ähnlichem erhältlich, kompakt verpackt im Plastikkoffer [2]. Fritzing ist eine Initiative der Designagentur Ixds und als eingetragener Verein angelegt. Drumherum docken diverse Projekte und ein Lab an. Für 80 Euro bietet auch der Berliner Elektronikhändler *Segor* ein Starterkit an [3]. Es enthält zusätzlich das ziemlich aktuelle, rund 140 Seiten fassende Buch „Arduino für Einsteiger"(April 2012) von O'Reilly im Wert von 10 Euro. Bei dieser Art von Massenpublikum-Büchern ist allerdings von Nachteil, dass sie zur Installation der Entwicklungsumgebung nur Windows- und Mac-Anwender berücksichtigen. Wer herausfinden will, welches Kit sich mehr lohnt, vergleicht die enthaltenen Komponenten. Und wer sich sein Kit lieber selbst zusammenstellt, findet im Webshop der Elektronikladen-Kette *Elmicro* die einzelnen Module ab 20 Euro [4]. Stefan Lesser hat übrigens gleich einen jahreszeitgemäßen Anwendungsfall parat: Weihnachtslichterketten mittels Ansprache jedes Lämpchens über den Bus programmieren [5]. (Abbildungen: Unternehmen)

Bernhard Schneck hat einen Tipp für Funkinteressierte, das **Funcube Dongle** [6]. Er beschreibt: „Das ist ein schmalbandiger (192kHz sampling rate) Software-defined Radiofunkempfänger (SDR) von zirka 150 kHZ bis 1.9 GHz, den man zum Beispiel mit Gnuradio betreiben kann. Er geht nicht zum Analysieren oder Pentesten von WLANs und Bluetooth, andererseits kann man damit einige interessante digitale Daten empfangen, etwa die ADS-B-Telemetriedaten von Verkehrsflugzeugen oder den Datenverkehr von der ISS. Außerdem ist es nur ein Empfänger. Ohne Zulassung und Lizenz sollte man normalerweise eh nicht

funken. Das Dongle ist auch nicht vergleichbar mit professionellen SDR-Radios. Ich kenne aber ein paar Leute, die eines haben und davon begeistert sind.“Kostenpunkt: 184 Euro plus 5 Euro Versandkosten bei Vorkasse.

Zu Bernhard Schnecks Dongle kommentiert Bernd Hennig, dass man nicht die Antenne vergessen sollte. Das bringt ihn auf die Idee eines **Wettersatellitenempfängers** als Weihnachtsgeschenk: Das Gerät *R2FX* ist für 180 Euro direkt beim Bastler erhältlich, dazu eine passende Antenne für 80 Euro [7, 8]. Die passende Software *WXtiImg*, die die Daten in Bilder umwandelt, gibt es als Freeware von einem neuseeländischen Anbieter [9]. Wie die Ergebnisse aussehen, zeigt Bernd auf seiner eigenen Seite [10]. (Abbildungen: Unternehmen, Bernd Hennig)



Von den etlichen **T-Shirts** von Getdigital (plus 4 - 6 Euro Versand) heben wir das Shirt *Widerstand ist zwecklos* hervor, zum Beispiel für 17 Euro im körperangepassten Frauen-Format [11]. Bei Sowaswillichauch.de – ein Linktipp von Jürgen Plate – gibt's das *E-Gitarren-Shirt* (35 Euro plus 5 Euro Versand) [12]. Es soll echte Akkorde hervorbringen, die Elektronik-Elemente sollen zum Waschen abnehmbar sein, und es kommt mit Magnetplektron und am Gürtel befestigbarem Verstärker daher. Keine Garantie, ob's funktioniert: Zu diesem Produkt gab es keine Kommentare oder Bewertungen. Bei gleicher Quelle gibt es dafür etwas Unriskantes, Niedliches für Leute, die gern kochen. Die Silikon-Figürchen **Lid Sit** (4 cm x 4 cm) halten den Topfdeckel hoch, damit nichts so schnell überkocht (12 Euro im Zweierpack plus 5 Euro Versand) [13].

Für die **Küche** hätte Thomas Bille außerdem gern die *Dart-Vader-Backform* (28 cm x 24 cm x 5 cm, 11 Euro), erhältlich via Amazon (plus 3 Euro Versand) von einem Händler namens Close Up in Ostfildern [14]. Er hat auch *Darth-Vader-Eiswürfelformen* für 17 Euro (ohne Abbildung) sowie einen *Pizzaschneider* in Form

des Star-Trek-Raumschiffs Enterprise. Der kostet 30 Euro, aber dann immerhin keinen Versand mehr. Direkt von Amazon als Händler ist Darth Vader auch als *Geburtstagskerze* erhältlich (9 Euro), falls der Nachwuchs die Neigung geerbt haben sollte [15].

Doch zurück zu den Großen. Es gibt nämlich Neues für Kaffee-Junkies. Das Kickstarter-Projekt **Coffee Joulies** verkauft seit einem knappen Jahr fünf kaffeebohnenförmige Stahlgebilde, die jeweils etwa die Größe eines Hühnereis haben [16]. Sie kosten – schluck – 50 Dollar. Von diesen Dingern legt man zwei in seinen Kaffeepot und kippt die brühendheißeLieblingsflüssigkeit drüber. Der Witz: Die Stahlbohnen enthalten ein spezielles Material, womit sie dem Kaffee Hitze entziehen – und speichern. Frischer Kaffee wird also schneller trinkbar, werben die Startupper, und er bleibt länger heiß, weil die Bohnen die Wärme an den Kaffee zurückgeben, wenn er beginnt, kalt zu werden. Schluck! Achtung: Da die Joulies aus den USA kommen, ist mit weiteren 10 bis 20 Dollar Versandkosten zu rechnen. (Abbildungen: Unternehmen, Coffee Joulies LLC)

Bernd Sommerfeld empfiehlt aus seinem eigenen Laden Lehmanns zum Beispiel die **Hex Bugs** – das sind Insekten als Miniroboter. Zum Beispiel die *Spinne* [17]: Für sie gibt es sogar eine Fernbedienung (7 cm x 11 cm, 25 Euro). Ihr Bruder, der Hex Bug *Skarabäus* (6 cm lang, 16 Euro), sticht dadurch hervor, dass er sich mit einem „Affenzahn“fortbewegt und unbeirrt wieder aufrappelt, wenn er umfällt, so verspricht die Beschreibung [18]. Den größten Ekelfaktor verspricht die Hex Bug *Larve* (10 cm x 3 cm, 15 Euro), die wacker ihrer Wege kriecht und die der Verkäufer als Scherzartikel für fremde Küchen einordnet [19]. Die Roboter sind bereits zusammengesetzt und verfügen über Abstands- und teilweise auch Lichtsensoren, damit sie sich in dunklen Ecken verkriechen (das wäre die Hex Bug *Krabbe* zu 17 Euro, ohne Abbildung und importiert aus China). (Abbildungen: Unternehmen)

Der Flaschenöffner in Form der **Klein'schen Flasche**, den Wolfgang Stief gern hätte, ist aus Stahl und Bronze gefertigt [22]. Er sieht wie eine Kleinigkeit aus, kostet aber inklusive 20 Dollar Versand stolze 90 US-Dollar. Für weitere 25 gibt's auch eine Gravur. Und wenn wir schon bei kostspielig sind: Werner Wiethege hätte sich gern selbst die Enigma geschenkt, die das altehrwürdige Londoner Auktionshaus Bornhams im November versteigerte [23]. Allerdings hat er nicht früh genug angefangen zu sparen, wie er einsehen muss: Das Museumsstück fand einen Käufer, der gut 85.000 Pfund hinblätterte. Werner tröstet sich aber: „Eine allein ist natuerlich auch nicht so interessant."(Abbildungen: Bathsheba.com, Bornhams)

Thomas Bille mag nicht nur Darth Vader, sondern auch Disketten. Die **Floppy-Post-its** stammen von dem türkischen Wahl-Kanadier und Designer Burak Kaynak. Angefertigt hat er sie nach eigenen Angaben für den englischen Geschenke-Großhandel SUCK UK. Hier kosten sie 7,50 Pfund, plus allerdings 12 Pfund Versandkosten nach Deutschland [20]. Bleiben wir bei den Floppys. Die gibts nämlich auch als Tisch – von Volker Lendecke ausgegraben [21]. Das silberne Metallstück auf der Oberseite des **Floppytable** lässt sich zur Seite schieben und verhüllt Dinge des täglichen Gebrauchs, etwa eine Fernbedienung, Schokoladenkekse – oder Floppy-Post-its. Nach Angaben des Herstellers Neluant van Exel – ein Berliner Designstudio – wird jeder Tisch aus Stahl auf Anfrage handgefertigt. Kostenpunkt: 700 bis 800 Euro. Nun denn. (Abbildungen: Unternehmen)

****** FLOPPYTABLE 3.5" HOT-ROLLED STEEL ******

Material:

- Hot-rolled steel (welded)
- Stainless steel (welded)

Measures:

- 27.56" width x 25.59" height x 17.72" depth
- 70cm width x 45cm height x 65cm depth

Extras:

- Secret space
- Unique lasered serial number
- Produced on-demand
- Hand-made

Price:

- X-MAS-SPECIAL
- 720 EURO (930 USD) + Shipping

Order your floppytable via 3.5@floppytable.com

Im 24. Türchen würdigt Hans Franke den Zuwachs im Haus Hardkernel. Das neue **Odroid-Entwicklerboard** U2 mit Android 4 oder Ubuntu 12.10 ist kleiner als eine Kreditkarte (4,8 cm x 5,2 cm) und wird aus Südkorea verschickt (Lieferzeit: 6 Wochen) [24]. Kostenpunkt: 90 Dollar plus 30 Dollar Versandkosten. Das Kühlgehäuse (5,9 cm x 5,7 cm x 6 cm) ist inbegriffen, der Stromadapter allerdings nicht (Stromzufuhr über USB ist nicht möglich) und kostet weitere 9 Dollar. Wer einen der zwei USB-Ausgänge mit Wlan ausbauen will, zahlt für die optional erhältliche Antenne (802.11n) 12 Dollar. eMMC-Module beginnen bei 8 GByte und 25 Dollar. Ein zugeschnittener Lüfter, der ins Gehäuse passt, kostet 8 Dollar. Auf dem Odroid U2 sitzt ein Samsung Exynos4412 Prime Cortex-A9 Quad Core 1.7Ghz, dazu kommt der Vierkern-Grafikchip Mali-400 zu 440MHz, 1 GByte RAM. Ausgänge: Megabit-Ethernet, Audio, HDMI, MicroSD-Schlitz und eMMC-Slot. (Abbildungen: Hardkernel)

Bezugsquellen

[1] Plüsch-Elektronik: https://www.adafruit.com/category/116 [2] Arduino von Fritzing: http://shop.fritzing.org/products/fritzing-starter-kit-with-arduino-uno [3] Arduino von Segor: http://www.segor.de/#/arduino-a-co/arduino-kits/arduino-starterset [4] Arduino-Einzelteile von Elmicro: http://elmicro.com/de/arduino.html [5] Lichterketten programmieren: http://www.deepdarc.com/2010/11/27/hacking-christmas-lights/ [6] Funcube-Dongle: http://www.thiecom.de/amsat-uk-funcube-pro-plus.html [7] Wetterdatenempfänger: http://www.df2fq.de/produkte/r2fx.html [8] Satellitenantenne: http://www.df2fq.de/produkte/Turnstile.html [9] Wetterdaten-Software: http://www.wxtoimg.com [10] Wetterdaten-Anwendungsbeispiel: http://www.segelflug.de/segelflieger/bernd.hennig/NOAA/ [11] T-Shirt *Widerstand ist zwecklos*: http://www.getdigital.de/products/Widerstand_ist_zwecklos [12] *Lid Sit* – Männchen zum Topfdeckelhalten: http://www.sowaswillichauch.de/lid-sid-schafft-zwischenraum-zwischen-topf-und-deckel [13] E-Gitarren-Shirt: http://www.sowaswillichauch.de/elektrogitarren-shirt [14] Darth-Vader-Kuchenform: http://www.amazon.de/Star-Wars-65922-Backform/dp/B003KV7L4Y/ref=sr_1_2?ie=UTF8&qid=1355486050&sr=8-2 [15] Darth-Vader-Geburtstagskerze: http://www.amazon.de/Dekoback-04-10-00169-Kuchenkerze-Darth-Vader/dp/B006FIL0ZU/ref=sr_1_9?ie=UTF8&qid=1355486964&sr=8-9 [16] Kaffeewarmhaltebohnen: http://www.joulies.com [17] Minirobo Spinne: http://www.lehmanns.de/shop/nocategory/25436058-2220000006385-hex-bug-spider [18] Minirobo Skarabäus: http://www.lehmanns.de/shop/nocategory/25436059-2220000006392-hex-bug-scarab [19] Minirobo Made: http://www.lehmanns.de/shop/nocategory/25436060-2220000006408-hexbug-larva [20] Floppy-Post-its: http://www.suck.uk.com/products/floppydiskstickynote [21] Floppytable: http://www.floppytable.com [22] Klein'scher Flaschenöffner: http://www.bathsheba.com/math/klein/ [23] Versteigerte Enigma: http://www.bonhams.com/auctions/19799/lot/74/ [24] Odroid: http://www.hardkernel.com/renewal_2011/products/prdt_info.php

Autorenrichtlinien
Hilfreiches für alle Beteiligten

Selbst etwas für die UpTimes schreiben? Gern. Jedes Thema ist willkommen, das ein GUUG-Mitglied interessiert und im Themenbereich der GUUG liegt. Was sonst noch zu beachten ist, steht in diesen Autorenrichtlinien.

Der Schriftsteller ragt zu den Sternen empor,
Mit ausgefranstem T-Shirt.
Er raunt seiner Zeit ihre Wonnen ins Ohr,
Mit ausgefranstem T-Shirt.

Frei nach Frank Wedekind

Wir sind an Beiträgen immer interessiert. Wir – das ist diejenige Gruppe innerhalb der GUUG, die dafür sorgt, dass die UpTimes entsteht. Dieser Prozess steht jedem GUUG-Mitglied offen. Der Ort dafür ist die Mailingliste <redaktion@uptimes.de>.

Themen und Zielgruppe

Die UpTimes richtet sich als Vereinszeitschrift der GUUG an Leser, die sich meistens beruflich mit Computernetzwerken, IT-Sicherheit, Unix-Systemadministration und artverwandten Themen auseinandersetzen. Technologische Diskussionen, Methodenbeschreibungen und Einführungen in neue Themen sind für dieses Zielpublikum interessant, Basiswissen im Stil von *Einführung in die Bourne Shell* hingegen eher nicht. Wer sich nicht sicher ist, ob sein Thema für die UpTimes von Interesse ist, kann uns gern eine E-Mail an <redaktion@uptimes.de> schicken.

Länge

Ein einseitiger Artikel hat mit zwei Zwischentiteln um die 2.700 Anschläge. Mit etwa 15.000 Anschlägen – inklusive 3 Abbildungen – landet man auf rund vier Seiten. Wir nehmen gern auch zum Beispiel achtseitige Artikel, achten dabei aber darauf, dass der Zusammenhang erhalten bleibt und dass es genug Bilder gibt, damit keine Textwüsten enstehen. Wer Interesse hat, für die UpTimes zu schreiben, macht sich am besten um die Zeichenzahl nicht so viele Gedanken – auch für kurze oder lange Formate finden wir einen Platz. Die Redaktion ist auch bei der konkreten Ideenentwicklung behilflich. Für eine Artikelidee an <redaktion@uptimes.de> reicht es, wenn Ihr ein bestimmtes Thema behandeln wollt.

Beitragsarten

Neben Fachbeiträgen sind Leserbriefe, Buchrezensionen, Konferenzberichte und Berichte aus dem Vereinsleben für uns immer interessant. Wer sich also nicht gleich traut, mehrseitige Artikel zu schreiben, darf sich gerne erstmal an kleineren Beiträgen versuchen.

Formate

LaTeX: Wir setzen die UpTimes damit. Wer es nicht kennt: Das ist ein Satzsystem, das aus dem technischen Buchdruck kommt. Daher können Autoren uns gern TeX- oder LaTeX-Dateien zusenden. Weil wir – wie es sich beim Publizieren gehört – mehrspaltig setzen und ein homogenes Erscheinungsbild anstreben, verwenden wir für die UpTimes aber bestimmte Formatierungen. Es ist deswegen relativ sinnlos, ausgefeilte Layoutanweisungen oder neben der Datei *Artikel.tex* noch Funktionsdateien einzusenden, die für das heimische Kompilieren nötig waren. Wir behalten uns vor, Texte für die Veröffentlichung in der UpTimes umzuformatieren. Eine Vorlage mit den von uns verwendeten Auszeichnungen etwa für Tabellen, Kästen und Abbildungen gibt es per Anfrage unter <redaktion@uptimes.de>.

ASCII: Alternativ können wir sehr gut blanke ASCII-Texte (UTF-8) verarbeiten.

OpenOffice bzw. LibreOffice, Word, PDF: Alle drei haben eine Formatierung, die wir grundsätzlich nicht übernehmen können.

Bilder: Wir können alle gängigen Bildformate verwenden, soweit ImageMagick sie verdaut und sie einigermaßen hochauflösend sind. Am besten eignen sich aber PNG- oder PDF-Dateien (für Bilder, nicht für Texte – siehe oben). Plant bei längeren Artikeln mit 1 Abbildung pro 300 Zeichen. Das

müssen aber nicht nur Bilder sein, sondern auch Tabellen, Listings oder ein Exkurskasten.

Listings: Der mehrspaltige Druck erlaubt maximal ca. 40 Zeichen Breite für Code-Beispiele. Breitere Listings setzen wir entweder als separate Abbildung, oder wir formatieren um.

Manuskripte einreichen

Am einfachsten ist es, wenn wir ein Manuskript per E-Mail an <redaktion@uptimes.de> erhalten. Das ist jederzeit möglich, spätestens jedoch vier Wochen vor dem Erscheinen der nächsten UpTimes. Zum eigentlichen Manuskript ist ein kleiner Infotext zum Autor wichtig, ein Bild ist sehr wünschenswert.

Wir sind sehr dankbar, wenn der Text vor Einsendung durch eine Rechtschreibkorrektur gelaufen ist. `aspell`, `ispell` oder `flyspell` für Textdateien sowie die von LibreOffice bieten sich an. Wir redigieren zwar die Einsendungen, doch üblicherweise bekommt man erst in mehreren Anläufen einen fehlerfreien Text hin. Wir halten uns an die neue deutsche Rechtschreibung.

Redaktion und Satz

Wir behalten uns vor, Texte für die Veröffentlichung in der UpTimes zu kürzen, und wir redigieren auch etwas. Das bedeutet, dass versehentliche Leeraussagen wegfallen, Syntax und Satzanschlüsse glatter werden, dass Passiva und Substantivierungen verringert oder auch Unklarheiten beseitigt werden (die zum Beispiel Fragen offen lassen oder aus Passivkonstruktionen resultieren, ohne dass der Schreibende das merkt). Manchmal ist dieser Prozess auch mit Nachfragen an den Autoren verbunden.

Die endgültige Textversion geht jedem Autoren am Ende zur Kontrolle zu. Dabei geht es um die inhaltliche Kontrolle, ob sich durch den Redaktionsprozess Fehler eingeschlichen haben – das ist weniger dazu gedacht, alles noch einmal umzuwerfen oder Verschreiber herauszusuchen. Ein bis zwei Wochen vor dem geplanten Erscheinungstermin setzt die Redaktion die Artikel. Nach Redaktion und Satz wird die UpTimes dann zum geplanten Termin veröffentlicht.

Rechtliches

Die Inhalte der UpTimes sollen ab Veröffentlichung unter der CC-BY-SA-Lizenz stehen, damit jeder Leser die Artikel und Bilder bei Nennung der Quelle weiterverbreiten und auch weiterverarbeiten darf. Bei allen eingereichten Manuskripten gehen wir davon aus, dass der Autor sie selbst geschrieben hat und der UpTimes ein nichtexklusives, aber zeitlich und räumlich unbegrenztes Nutzungs- und Bearbeitungsrecht unter der CC-BY-SA einräumt.

Bei Fotos oder Abbildungen, die nicht selbst erstellt wurden, flutscht das gern mal durch: Es ist wichtig, dass der Autor sich bei dem Urheber die Erlaubnis zu dieser Nutzung einholt, und fragt, wie die Quelle genannt zu werden wünscht. Die Frage nach der CC-BY-SA ist hierbei besonders wichtig.

An Exklusivrechten, wie sie bei kommerziellen Fachzeitschriften üblich sind, hat die UpTimes kein Interesse. Es ist den Autoren freigestellt, ihre Artikel noch anderweitig nach Belieben zu veröffentlichen.

Finanzielles

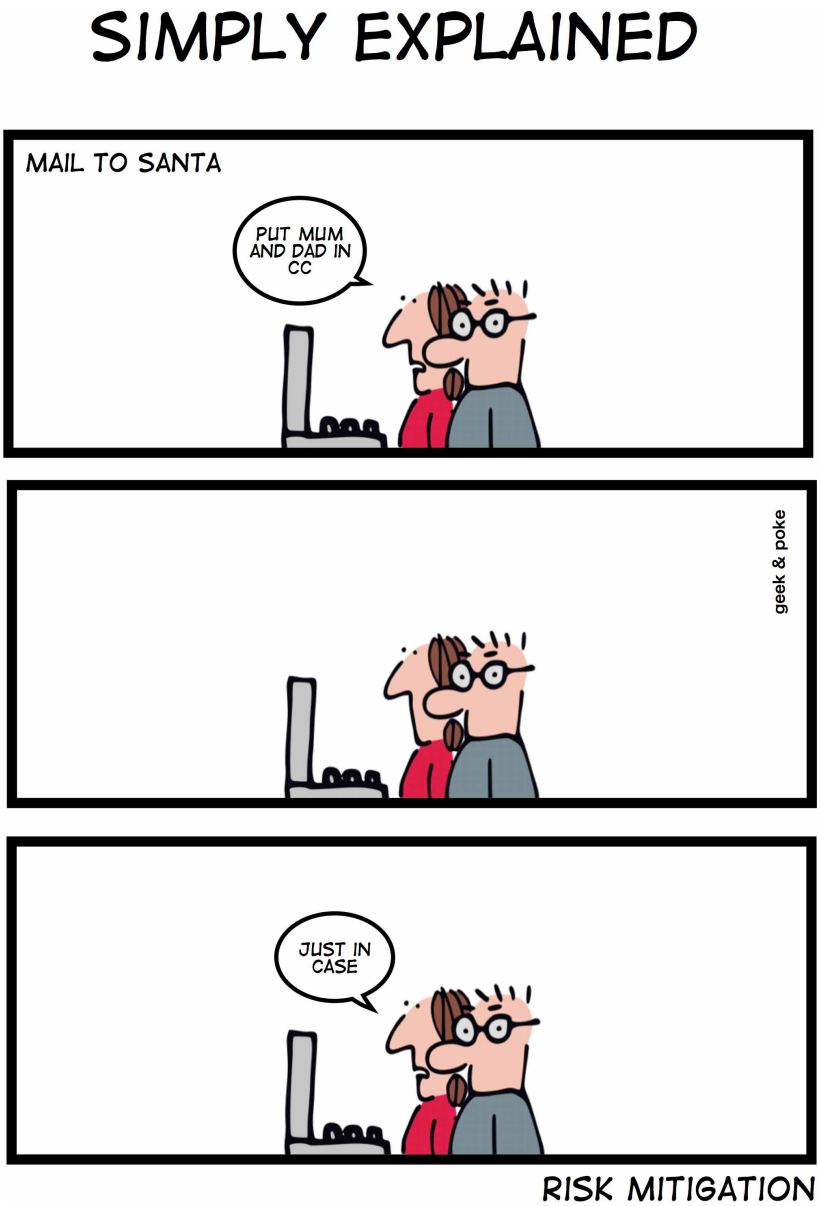
Für Fachbeiträge, die einen eigenen inhaltlichen Anspruch erheben, zahlt die GUUG dem Autor nach Rechnungstellung durch den Autor pro Seite 50 € zuzüglich eventuell anfallender USt. Leserbriefe, Buchrezensionen und Artikel bezahlter Redakteure sind davon ausgenommen. Gleiches gilt für Paper, wenn die UpTimes die Proceedings der Konferenz beinhaltet.

Nächste Ausgabe (print):
UpTimes 01-2013, Proceedings des Frühjahrsfachgesprächs 2013

- Redaktionsschluss: Montag, 4. Februar 2013.
- Erscheinung: 26. Februar 2013.
- Manuskripte an: <ffg2013@guug.de>

Nächste Ausgabe (digital):
UpTimes 02-2013, Sommer-Ausgabe

- Redaktionsschluss: Sonntag, 2. Juni 2013.
- Erscheinung: 31. Juli 2013.
- Gesuchte Inhalte: Fachbeiträge über Unix und verwandte Themen; Veranstaltungsberichte; eigene Buchvorstellungen, Rezensionen; Beiträge zum Vereinsleben.
- Artikelideen und Manuskripte an: <kehrer@guug.de> oder direkt an die Mailingliste <redaktion@uptimes.de>



Über die GUUG

Vereinigung deutscher UNIX-Benutzer

German Unix User Group e.V.

Die Vereinigung Deutscher Unix-Benutzer hat gegenwärtig 678 Mitglieder, davon 86 Firmen und Institutionen.

Im Mittelpunkt der Aktivitäten der GUUG stehen Konferenzen. Ein großes viertägiges Event der GUUG hat eine besondere Tradition und fachliche Bedeutung: In der ersten Jahreshälfte treffen sich diejenigen, die ihren beruflichen Schwerpunkt im Bereich der IT-Sicherheit, der System- oder Netzwerkadministration haben, beim *GUUG-Frühjahrsfachgespräch* (FFG).

Seit Oktober 2002 erscheint mit der *UpTimes* – die Sie gerade lesen – eine Vereinszeitung. Seit 2012 erscheint die UpTimes einerseits zu jedem FFG in Form einer gedruckten Proceedings-Ausgabe (ISBN), und andererseits im Rest des Jahres als digitale Redaktionsausgabe (ISSN). Daneben erhalten GUUG-Mitglieder zur Zeit die Zeitschrift *LANline* aus dem Konradin-Verlag kostenlos im Rahmen ihrer Mitgliedschaft.

Schließlich gibt es noch eine Reihe regionaler Treffen (http://www.guug.de/lokal): im Rhein-Ruhr- und im Rhein-Main-Gebiet sowie in Berlin, Hamburg, Karlsruhe und München.

Warum GUUG-Mitglied werden?

Die GUUG setzt sich für eine lebendige und professionelle Weiterentwicklung im Open Source-Bereich und für alle Belange der System-, Netzwerkadministration und IT-Sicherheit ein. Wir freuen uns besonders über diejenigen, die bereit sind, sich aktiv in der GUUG zu engagieren. Da die Mitgliedschaft mit jährlichen Kosten

Fördermitglied	350 €
persönliches Mitglied	90 €
in der Ausbildung	30 €

verbunden ist, stellt sich die Frage, welche Vorteile damit verbunden sind?

Neben der Unterstützung der erwähnten Ziele der GUUG profitieren Mitglieder auch finanziell davon, insbesondere durch die ermäßigten Gebühren bei den Konferenzen der GUUG und denen anderer europäischer UUGs. Mitglieder bekommen außerdem *c't* und *iX* zum reduzierten Abopreis.

Wie GUUG-Mitglied werden?

Füllen Sie einfach das umseitige Anmeldeformular aus und schicken Sie es per Fax oder Post an die unten auf dem Formular angegebene Adresse. Falls Sie die Seite nicht herausreißen wollen: Sie können den Mitgliedsantrag als PDF herunterladen, siehe URL auf dem Mitgliedsantrag.

Impressum

UpTimes – Mitgliederzeitschrift der
German Unix User Group (GUUG) e.V.
Herausgeber: GUUG e.V.
Bruno-Walter-Ring 30
D-81927 München
Tel.: +49-(89)-380 125 95 0
Fax: +49-(89)-380 125 95 9
E-Mail: <redaktion@uptimes.de>
Internet: http://www.guug.de/uptimes/

Autoren dieser Ausgabe: Jürgen Plate, Wolfgang Stief, Anika Kehrer Mathias Weidner, Jens Link, Stefan Schumacher
V.i.S.d.P: Wolfgang Stief, Vorstandvorsitzender, Anschrift siehe Herausgeber
Chefredaktion: Anika Kehrer
LaTeX-Layout (PDF): Robin Schröder
XHTML-Layout (ePub): Mathias Weidner
Titelbild und -gestaltung: Hella Breitkopf
Bildnachweis: Comicreihe *geek & poke* CC-by-sa, mit freundlicher Genehmigung von Oliver Widder. Andere Quellennachweise am jeweiligen Bild.
Verlag: Lehmanns Media GmbH, Hardenbergstraße 5, 10623 Berlin
ISSN: 2195-0016

Wenn Sie Interesse an Anzeigen in der UpTimes haben,
wenden Sie sich bitte an <werbung@guug.de>.

German Unix User Group e.V.
Mitgliedsantrag

Name, Vorname: .....

Firma/Organisation: .....

.....

Straße: .....

PLZ, Ort: .....

E-Mail: .....

Ich möchte/wir möchten

☐ persönliches Mitglied (Jahresbeitrag 90 €)
  ☐ Ich bin in der Ausbildung (bitte Nachweis beifügen) und zahle nur 30 €
☐ Fördermitglied (Jahresbeitrag 350 €)

werden.

☐ Bitte richten Sie mir den Alias ..... @guug.de ein.

☐ Ich bin mit der Weitergabe meiner Adressdaten an Verlage, die GUUG-Mitglieder im Rahmen der Mitgliedschaft kostenlos mit Zeitschriften beliefern, einverstanden.

Ich bin damit einverstanden, dass die o. g. Daten durch die GUUG elektronisch gespeichert werden. Mir ist bekannt, dass die Kündigung der Mitgliedschaft nur mit einer Frist von drei Monaten zum Ende des Kalenderjahres möglich ist.
Die Satzung ist unter http://www.guug.de/verein/satzung/ einsehbar.

....
Ort, Datum Unterschrift

Ich ermächtige die GUUG bis auf Widerruf zum jährlichen Einzug des Mitgliedsbeitrags von meinem Konto:

Name des Kontoinhabers: .....

Kontonummer: .....

BLZ: .....

Bank: .....

....
Ort, Datum Unterschrift

Bitte faxen Sie diesen Antrag an +49-(89)-380 125 95 9 oder schicken ihn an:

GUUG e.V. * Postfach 25 01 23 * D-44739 Bochum

Dieses Formular ist auch unter http://www.guug.de/verein/mitglied/guug-anmeldeformular.pdf zu finden.