Topthemen dieser Ausgabe

Webzugriff

Seite 3

Normalerweise muss man nicht im Detail wissen, wie ein Browser auf eine Webseite zugreift. Die genaue Kenntnis der Vorgänge kann jedoch bei der Fehlersuche eine große Hilfe sein. Das Zusammenspiel von DNS, Protokollen und Proxys soll in dem Artikel exemplarisch am Zugriff auf eine Webseite gezeigt werden. Dabei ist hoffentlich alles enthalten: Zerlegen der URL, Finden des Servers, Holen der Seite und anschließende Darstellung. (weiterlesen)

Perl-Tutorium: Teil 1 – Das erste Programm

Seite 22

Nachdem im vorigen Teil die Perl-Geschichte, Perl-Philosophie und Gemeinschaft der Nutzer vorgestellt wurde, beginnt jetzt die Reise zum ersten eigenen Perl-Programm. Nachdem geprüft ist, ob alle wichtigen Werkzeuge funktionstüchtig und griffbereit verpackt sind, geht es zum ersten Etappenziel: Skalare Variablen und einfache IO. (weiterlesen)

Kurzreview: Humble Indie Bundle 3

Seite 37

Das Humble Indie Bundle hat schon eine gewisse Tradition, wurde die erste Version bereits im Mai 2010 veröffentlicht. Teil des Bundles sind Spiele, die von verschiedenen Independent-Studios entwickelt wurden und auf allen großen Plattformen Linux, Mac OS X und Windows laufen. Ende Juli wurde die dritte Ausgabe veröffentlicht, auf deren Inhalt in dem Artikel ein kleiner Blick geworfen werden soll. (weiterlesen)

ISSN 1867-7991

Editorial

Sommerloch?

Nein, die eigentlich an Nachrichten arme Zeit des Jahres geht spurenlos an **freies**Magazin vorüber – und so haben wir auch diesen Monat wieder den einen oder anderen spannenden Artikel in petto. Darüber hinaus nutzen wir an dieser Stelle die Möglichkeit, um in eigener Sache um etwas Aufmerksamkeit zu bitten.

Layouter gesucht

Wie schon im Juli [1] angekündigt, sucht **freies**Magazin wieder nach neuem Teamnachwuchs. In der Zwischenzeit haben sich auch die ersten neuen Mitstreiter eingefunden. Darüber freuen wir uns sehr. Allerdings suchen wir nach wie vor noch Nachwuchs für das Layout in unserem Magazin.

Sollten Sie, liebe Leser, Interesse, Zeit und Lust mitbringen, um **freies**Magazin kreativ mitzugestalten und den Artikeln das richtige Layout zu verpassen – dann können Sie sich bei uns austoben.

Als Werkzeug der Wahl kommt für das Layout das Textsatzsystem LaTeX [2] zum Einsatz, sodass etwas Wissen auf diesem Gebiet sicherlich nicht schaden kann. Keine Bange, das Magazin ist aber so gehalten, dass sehr viel mit Makros gearbeitet wird. Aus diesem Grund sind auch nicht zwingend LaTeX-Profis gefordert, um das Magazin zu setzen. Wenn jemand Interesse hat, sich auf diesem Gebiet einzuarbeiten, helfen wir gerne weiter.

Daneben wäre Wissen im Umgang mit dem Versionskontrollsystem Subversion (SVN) gut, ist aber nicht zwingend erforderlich. Die wenigen SVN-Befehle, die dazu benötigt werden, sind schnell erlernt – auch Dank einer hervorragenden Dokumentation [3]. Zusätzlich stehen in den meisten Linux-Distributionen und Desktopumgebungen auch grafische Oberflächen für die Verwaltung bereit, sodass man nicht zwingend die Konsole bedienen muss – auch wenn es darüber manchmal schneller geht.

Falls Sie also Lust und Interesse haben, an der Gestaltung von **freies**Magazin mitzuwirken, dann schreiben Sie sich doch einfach über den üblichen Weg an redaktion@**freies**Magazin.de. Auf Ihr Engagement freuen wir uns!

Und nun wünschen wir Ihnen viel Spaß mit der neuen Ausgabe.

Ihre **freies**Magazin-Redaktion

Links

[1] http://www.freiesmagazin.de/20110725-freiesmagazin-sucht-unterstuetzung
[2] http://www.latex-project.org
[3] http://svnbook.red-bean.com

Inhalt

Das Editorial kommentieren

Webzugriff

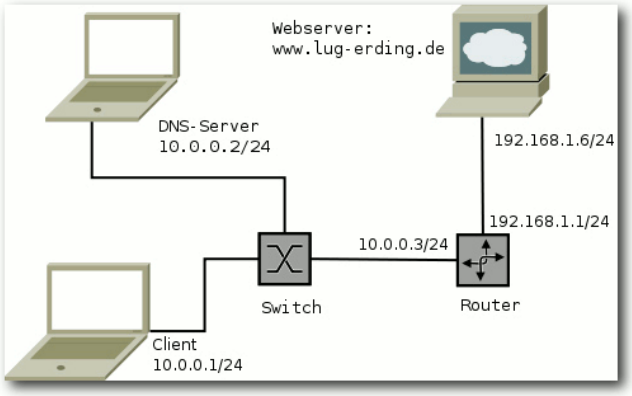
von Dirk Geschke

Normalerweise muss man nicht im Detail wissen, wie ein Browser auf eine Webseite zugreift. Die genaue Kenntnis der Vorgänge kann jedoch bei der Fehlersuche eine große Hilfe sein.

Redaktioneller Hinweis: *Der Artikel „Webzugriff“ erschien erstmals bei Pro-Linux [1].*

Einleitung

Das Zusammenspiel von DNS, Protokollen und Proxys soll hier exemplarisch am Zugriff auf eine Webseite gezeigt werden. Dabei ist hoffentlich alles enthalten: Zerlegen der URL, Finden des Servers, Holen der Seite und anschließende Darstellung.



Der verwendete Testaufbau.

Zerlegen der URL

Der erste Schritt besteht in der Analyse der URL. So wird diese in die drei wichtigen Bestandteile Protokoll, Servername und Pfad auf dem Zielsystem zerlegt. Beim Protokoll wird gewöhnlich HTTP verwendet. Wird kein Port angegeben, so ist es der www-Port oder auch der http-Port. Das ist ein Alias, wie in **/etc/services** definiert:

```
www     80/tcp    http    # WorldWideWeb HTTP
www     80/udp            # HyperText
                          # Transfer Protocol
```

Definiert sind die Services in der Regel immer für beide Protokolle, also TCP und UDP. Nur wird wohl keiner jemals UDP für HTTP verwenden.

Neben HTTP wird oft noch HTTP over SSL/TLS verwendet, das wird durch https angegeben. Dabei wird zwischen dem Transportprotokoll TCP und HTTP noch eine Verschlüsselungsschicht eingebaut. Da gibt es oft noch FTP, hier wird ganz normal Port 21 verwendet. Aber auch der Zugriff auf lokale Dateien ist mit file möglich.

Nach dem Protokoll folgt ein Doppelpunkt und zwei Slashes **://**. Bis zum nächsten Slash folgt dann der Servername, gefolgt vom Pfad auf dem Zielsystem. Also bei

```
http://www.lug-erding.de/index.html
```

ist HTTP das Protokoll, **www.lug-erding.de** der Servername und **index.html** der Pfad zu den Daten auf dem Zielsystem. Gewöhnlich kann **index.html** weggelassen werden, dann besteht der Pfad nur aus einem Schrägstrich (**/**). Der Webserver liefert dann in aller Regel die Datei **index.html** aus, oder eine Lokalisierung wie z. B. **index.html.de**, falls der Browser die deutsche Sprache bevorzugt.

Wer sich schon einmal gewundert hat, warum der Browser beim Zugriff auf lokale Dateien drei Slashes in die URL einbaut, hat nun die Antwort: Der Servername ist leer. Anderenfalls würde er den ersten Pfadteil als Servernamen interpretieren. Das ergibt bei file natürlich keinen Sinn.

Die URL für den lokalen Dateizugriff könnte also so aussehen:

```
file:///home/geschke/WEB/index.html
```

Auch die Protokollnamen, TCP oder UDP, also das nächst höhere Protokoll nach IP, kann in die entsprechenden Zahlenwerte per **/etc/protocols** aufgelöst werden:

```
tcp     6   TCP   # transmission
                  # control protocol
udp    17   UDP   # user datagram
                  # protocol
```

TCP ist also das IP-Protokoll 6, UDP hat die Nummer 17. Es ist auch später entstanden und deutlich einfacher als TCP.

Proxy finden

Ein Browser kann natürlich versuchen, die Seite direkt vom Server zu laden, es ist aber auch möglich, die Seite über einen (Caching-)Webproxy zu holen. Das kann den Vorteil haben, dass Daten dort im Cache liegen oder auch einfach nur ver-

schiedene Wege zu den Servern hier zentral gepflegt werden können. Auch besteht hier die Möglichkeit, zentral den Zugriff zu reglementieren.

Wenn der Browser für den Gebrauch eines Proxys konfiguriert ist, so muss er auch wissen, welchen Proxy er wann verwenden soll. Da gibt es mehrere Möglichkeiten:

1. keinen Proxy verwenden
2. manuelle Proxy-Konfiguration, also Angabe von der Adresse und Port
3. automatische Proxy-Konfiguration via URL, PAC-Datei (Proxy Auto-Config)
4. Proxy-Einstellungen des Systems, Environment-Variablen wie **http_proxy**, **ftp_proxy** oder **https_proxy**
5. Proxy-Einstellungen über das Netzwerk automatisch erkennen, **wpad.dat**

Bei der PAC-Datei und **wpad.dat** wird ein kleines JavaScript-Programm geladen, über dieses können verschiedene Proxys je nach Art der URL verwendet werden. Bei der manuellen Konfiguration kann man nur einen Proxy angeben, es besteht aber die Möglichkeit, dass man Ausnahmen definiert. Diese werden dann direkt vom Server bezogen und nicht via Proxy.

Die via Netzwerk automatisch erkannten Proxy-Einstellungen können über vielfache Wege ermittelt werden, das kann das Suchen nach einem Webserver in der eigenen Domain oder via DHCP oder DNS bedeuten. Genauer kann man es bei Wikipedia [2] nachlesen.

Hier wird aber auch mitunter der nächste Schritt schon relevant: Die Namensauflösung. Gerade wenn Namen und Ausnahmen für IP-Adressen verwendet werden oder die PAC-Datei den zuständigen Proxy auf Basis von IP-Adressen bestimmen will, so wird auch eine Namensauflösung benötigt. Hier wird DNS notwendig. Ein Vorteil eines Proxys ist, dass man den DNS leichter konfigurieren kann.

Sollte man also beim Surfen immer eine Verzögerung von 5-10 Sekunden bei Erstzugriffen merken, dann könnte es an der fehlerhaften Namensauflösung liegen. Das ist der typische Zeitraum, wann der Resolver aufgibt und der Browser einfach alles über den definierten Default-Proxy abwickelt.

Namensauflösung

Wird die Namensauflösung nicht über den Proxy geregelt oder der Proxy wird mit Namen statt IP-Adresse angegeben, so muss der Client, also der eigene PC, den Namen auflösen. Es gibt mehrere Verfahren, wie man vom Namen zur IP-Adresse kommen kann. Unter Linux wird das über den hosts-Eintrag in der Datei **/etc/nsswitch.conf** geregelt. Bei mir steht da z. B.:

```
hosts:          files dns
```

In diesem Fall wird der Name zuerst in der lokalen Datei **/etc/hosts** gesucht. Steht hier der Name mit einer IP-Adresse, so wird die Suche abgebrochen und die Adresse verwendet.

Damit kann man z. B. die Adressen vom DNS-Server „überladen“. Für Testzwecke kann das manchmal hilfreich sein, manche verwenden es auch, um „DoubleClick“ oder „Google-Analytics“ auf 127.0.0.1 umzubiegen. Damit gibt es weniger Werbung bzw. Verfolgung der Webaufrufe.

Kann der Name nicht aufgelöst werden, so erfolgt dann die Abfrage von DNS-Servern. Hierfür ist die Datei **/etc/resolv.conf** relevant. In ihr können bis zu drei Nameserver stehen, es kann auch eine Suchdomain angegeben werden: Wird der direkte Name nicht gefunden, so werden nach und nach die angegebenen Domainnamen angefügt und die Suche erneut gestartet, bis eine IP-Adresse gefunden wird. Anderenfalls wird ein Fehler zurückgeliefert, der Browser hängt eine Zeit lang und liefert dann eine entsprechende Fehlermeldung.

Ferner könnten in **nsswitch.conf** auch noch **nis/nisplus** oder **ldap** stehen. Das dürfte heute aber kaum noch einer für die Namensauflösung verwenden. Hat man den Avahi-Daemon installiert, das ist ein sogenanntes „Zeroconf“-Programm wie „Rendezvous“ oder „Bonjour“, gibt es noch ein Verfahren mehr. Mit Avahi kann ein Netzwerk ohne eigenes Zutun, lediglich durch das Verkabeln, aufgebaut werden. Dem einen oder anderen sind diese Adressen bestimmt schon begegnet, sie liegen im Bereich 169.254.x.y. Warum man für so etwas ein ganzes Class-B Netz verschwendet, ist eine andere, gute Frage. Oft wird dieser Daemon durch die Distribution automatisch mitinstalliert. Dann

gibt es noch die Option **mdns**, der Eintrag in **/etc/nsswitch.conf** kann dann so aussehen:

```
hosts:          files mdns4_minimal [NOTFOUND=return] dns mdns4
```

Diese **mdns**-Einträge suchen via Multicast nach der IP-Adresse, d. h. sie rufen in das Netzwerk hinein, ob jemand die IP-Adresse zu dem Namen hat. Meldet sich hier ein System mit „Nein“, so bricht die Suche ab. Man sollte diesen ganzen Avahi-Krempel einfach wegwerfen, er stört mehr, als er hilft. Wer von Netzwerken keine Ahnung hat, der soll einfach die Finger davon lassen und sich nicht auf die Magie des Betriebssystems verlassen. Aber dummerweise installieren viele Distributionen das Teil „automagisch“ mit.

Idealerweise sollte man also nur

```
hosts: files dns
```

in der Datei **/etc/nsswitch.conf** haben. Ferner sollte in **/etc/hosts** immer der Name **localhost** enthalten sein:

```
127.0.0.1       localhost
```

localhost wird von vielen lokalen Diensten verwendet, der Name sollte also zum einen immer auflösbar sein und zum anderen auf die Loopback-Adresse verweisen. Sonst kann man schon seltsame Effekte haben . . .

Finden des DNS-Servers

Da somit bekannt ist, wie die Namensauflösung erfolgt, kann man davon ausgehen, dass ein DNS-Server involviert ist und nicht über **/etc/hosts**-Auflösungen gesurft wird. Jetzt kommt der Netzwerkbereich: Wie wird der DNS-Server im Internet gefunden?

Zur kurzen Erinnerung an TCP/IP: Das Protokoll besteht aus mehreren unabhängigen Schichten, jede Schicht hat ihr eigenes Protokoll und ihre eigene Aufgabe. Oft wird dafür auch das OSI-Modell mit seinen sieben Schichten herangezogen. TCP/IP hält sich aber nur für die ersten Schichten daran. Relevant sind hier die ersten vier Schichten:

1. Physikalische Schicht, z. B. das Netzwerkkabel
2. Datenverbindungsschicht, oft Ethernet
3. Netzwerkschicht, IP
4. Transportschicht, oft TCP oder UDP

Die erste Schicht betrifft nur die Art der Verkabelung oder die Art des Funks. Relevant wird es mit der zweiten Schicht, das ist in aller Regel Ethernet. Früher war hier auch noch Token Ring verbreitet, das kommt aber wohl nur noch selten zum Einsatz. Aber das zeigt den klaren Vorteil des Schichtenmodells. Die höheren Schichten brauchen nicht zu wissen, ob Ethernet, WLAN oder gar der Token Ring zum Einsatz kommen. Ebenso brauchen TCP/UDP nicht zu wissen, ob darunter ein IPv4 oder ein IPv6 verwendet wird. Das macht es so leicht, hier andere Protokolle zu verwenden.

Für das Weitere wird einfach von Ethernet ausgegangen. IP ist das Protokoll, das die Internetsysteme miteinander verbindet, das bedeutet hier die Adressierung der Systeme an Hand der IP-Adressen. Für das lokale Netzwerk, also bis zum direkt angeschlossenen System oder dem Router/Gateway, ist aber Ethernet zuständig.

Wie wird denn nun der DNS-Server gefunden? Dazu muss man wissen, wie er erreicht werden kann. Ein Blick in die Routing-Tabelle offenbart dann, ob der DNS-Server lokal angeschlossen ist, also im gleichen Subnetz steht, oder ob er über einen Router erreicht werden muss.

Die logische Adressierung erfolgt über die IP-Adresse. Der Versand zum nächsten angeschlossenen System, also dem Server oder Router, erfolgt aber über Ethernet. Erst einmal muss also die Ethernetadresse des Systems gefunden werden, bekannt ist aber nur eine IP-Adresse.

Das erfolgt nun mit dem Address-Resolution-Protocol. Dabei sendet das suchende System, also der eigene PC, ein spezielles Ethernetpaket an alle angeschlossenen Systeme, und fragt nach der Hardware-Adresse zu der gewünschten IP-Adresse, also entweder der des DNS-Servers, wenn er direkt im Subnetz steht, oder die des Routers.

Die Hardware-Ethernet-Adresse besteht bei Ethernet aus sechs Bytes, und die sind für alle Ethernetkarten eindeutig. Sie werden auch MAC-Adresse (Media-Access-Control) genannt. Die ersten drei Bytes identifizieren dabei den

Hersteller der Ethernetkarte, die restlichen drei sind eine einmalige Adresse bei diesem Hersteller. Diese Adressen werden in der Regel als Hexadezimalzahlen, getrennt durch Doppelpunkte dargestellt, z. B.:

```
00:1a:70:63:16:f5
```

Die ersten drei Bytes liefern dann über die offizielle Liste [3] den Hersteller der Karte:

```
00-1A-70    (hex)     Cisco-Linksys, LLC
```

Es gibt auch spezielle Adressen, wie z. B. die Broadcast-Adresse. Bei dieser sind alle Bits gesetzt:

```
ff:ff:ff:ff:ff:ff
```

Da die Zieladresse zuerst im Ethernet-Header steht, können alle Ethernet-Karten leicht erkennen, ob das Paket für sie bestimmt ist, sie kennen ja die eigene MAC-Adresse. Bei Broadcast-Adressen wissen die Karten dann durch die spezielle Adresse auch gleich, dass sie das Paket entgegennehmen sollen. Derartige Pakete nimmt also jede direkt angeschlossene Ethernetkarte an und leitet diese an das Betriebssystem weiter.

Bei einem ARP-Request steht dann in den Nutzdaten, dass das eigene System die MAC-Adresse zu der angegebenen IP-Adresse sucht. Da alle Systeme diese sehen, sollte das System antworten, das die IP-Adresse besitzt. Das ist der ARP-Response. Mit anderen Worten:

- Man hat die IP-Adresse.
- Die Adressierung im LAN-Segement erfolgt über Ethernetadresse.
- Man ruft in die Runde: Wer hat die Ethernetadresse zu der IP-Adresse?
- Das System mit der IP-Adresse antwortet und sendet die MAC-Adresse, also die Ethernet-Hardware-Adresse mit.

Wer einen Blick auf das Bild oben wirft, der sieht nun, dass die MAC-Adresse vom DNS-Server mit der IP-Adresse 10.0.0.2 gesucht wird. Das ist in der Datei **lug.pcap** [4] der erste Eintrag:

```
00:13:77:5b:25:ad > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: ↷
arp who-has 10.0.0.2 tell 10.0.0.1
```

Das ist der Broadcast, zu erkennen an dem Ziel **ff:ff:ff:ff:ff:ff**. Die erste Adresse ist die eigene MAC-Adresse für Antworten. Diese Nachricht wird an alle Ethernetkarten gesendet, die im gleichen Segment stecken. Die Karten nehmen dieses Paket dann entgegen und reicht es an den Kernel weiter. Dieser antwortet dann, wenn er die IP-Adresse hat:

```
00:1e:68:ef:be:63 > 00:13:77:5b:25:ad, ethertype ARP (0x0806), length 60: ↷
arp reply 10.0.0.2 is-at 00:1e:68:ef:be:63
```

Hier sieht man noch mehr. Im ARP-Request steht neben der gesuchten Adresse auch die eigene. Damit kann das System, das gesucht wird, sich schon einmal die IP-Adresse und MAC-Adresse des Anfragenden merken (ARP-Cache), denn es gibt wohl eine hohe Wahrscheinlichkeit, dass da gleich noch mehr Traffic kommen wird.

Die Antwort wird auch direkt an die MAC-Adresse des Fragenden gesendet, der Absender hat die MAC-Adresse des gesuchten Systems.

Übrigens: Bei DHCP wird in der Regel auch erst ein ARP-Request für die frisch vergebene IP-Adresse vom System selber versendet. Damit wird getestet, ob sie nicht schon existiert, also jemand diese zum Beispiel von Hand konfiguriert hatte.

Das war ein kurzer Ausflug in die Sicherungsschicht [5], wer es genauer wissen will, kann in diesem Vortrag zu Netzwerkgrundlagen [6] noch mehr Infos finden.

DNS-Namensauflösung

Nachdem nun klar ist, wie der DNS-Server oder der Router dahin gefunden werden, kann das IP-Paket auf die Reise geschickt werden. Im Fall von DNS ist das in der Regel ein UDP-Paket an Port 53 des Servers. Auf der nächsten Seite sind allerdings zwei zu sehen, einmal eine Anfrage für die IPv4-Adresse und einmal die für die IPv6-Adresse.

Die MAC-Adressen (und Zeiten) wurden weggelassen. Man sieht dafür diverse Werte aus dem

```
IP (tos 0x0, ttl 64, id 16351, offset 0, flags [DF], proto UDP (17), length 63)
10.0.0.1.40105 > 10.0.0.2.53: 30718+ A? www.lug-erding.de. (35)
IP (tos 0x0, ttl 64, id 16352, offset 0, flags [DF], proto UDP (17), length 63)
10.0.0.1.40105 > 10.0.0.2.53: 9550+ AAAA? www.lug-erding.de. (35)
```

IP-Header. So wird kein TypeOfService verwendet, das Feld wird heute auch eher als QoS (Quality of Service) benutzt.

Interessant ist noch die TTL: Diese gibt an, wie lange das IP-Paket weitergeleitet werden darf. Jeder Router verringert diesen Wert um 1. Landet ein System bei 0, so wird das Paket verworfen und an den Absender wird ein ICMP Time Exceeded gesendet, um ihn darüber zu informieren.

Das Programm traceroute verwendet z. B. diese Option, um den Weg der Pakete zu bestimmen. Dabei wird sukzessive der Wert von 1 hochgezählt. Bei einem Wert von 1 antwortet der erste Router, bei 2 der zweite, etc. bis man am Zielsystem angekommen ist. Da die Router die ICMPs mit der eigenen IP-Adresse senden, kann man so den Weg bestimmen.

Die **id** ist wohl selbsterklärend, sie spielt in Verbindung mit **offset** eine Rolle. Müssen die IP-Pakete unterwegs zerlegt werden, da nur kleinere Bestandteile weitergeleitet werden können (DSL hat z. B. eine MTU (Maximum Transfer Unit) von 1492, Ethernet jedoch von 1500), so kann über die **id** und den **offset** bestimmt werden, wozu die Daten gehören.

Aber das DF-Flag verbietet eine Zerlegung in kleinere Pakte: „Don't Fragment." Das heißt, wenn eine Fragmentierung notwendig ist, darf man es auf Grund dieses Flags nicht tun. Stattdessen wird das Paket verworfen und eine ICMP-Meldung wird an den Absender geschickt. Diese besagt, dass das Paket verworfen wurde, da es zu groß war. Gleichzeitig wird noch mitgesendet, was die maximale Größe ist, die möglich wäre.

Wenn man also per Ethernet an seinen DSL-Router Pakete mit 1500 Bytes Größe sendet, so zerlegt der Router dieses Paket in zwei: Einmal 1492 Bytes und einmal 28 Byte, 20 Byte für einen neuen IP-Header plus die fehlenden 8 Bytes. Letzteres wird noch durch Füllbytes auf eine Mindestgröße aufgeblasen. Der Empfänger muss diese Pakete dann wieder zusammenbasteln.

Bei gesetztem DF-Bit wird das Paket verworfen, der ICMP wird gesendet und das sendende System erkennt nun: Mehr als 1492 geht nicht. Folglich werden dann alle Pakete mit einer maximalen Größe von 1492 Bytes versendet, es gibt keine Fragmentierung und auch nicht zwei statt einem Paket. Das erhöht den Durchsatz deutlich.

Die Werte danach besagen, dass die nächste Schicht UDP enthält, das ist laut **/etc/protocols** der Wert 17:

```
udp   17   UDP   # user datagram protocol
```

Danach folgen die IP-Adressen und UDP-Ports. Der Client-Port ist in aller Regel beliebig, der Zielport mit 53 (domain aus **/etc/services**) fest, also:

Ursprung : 10.0.0.1

Quellport : 40105

Ziel : 10.0.0.2

Zielport : 53

Anschließend kommt die Payload, also die eigentlichen Daten, die per UDP übermittelt werden. Das ist die Frage nach dem A- und AAAA-Record, also der IPv4- bzw. IPv6-Adresse von **www.lug-erding.de**. Die IPv6-Adresse existiert nicht, es gibt aber eine IPv4-Adresse. Die Ausgabe ist wieder etwas gekürzt:

```
IP 10.0.0.2.53 > 10.0.0.1.40105: ↷
30718* 1/1/1 A 192.168.1.6 (85)
```

Und schon ist das Ziel erreicht: Die IP-Adresse des Webservers ist gefunden. War doch ganz einfach . . .

Eine Frage dürfte noch aufgetreten sein: Wie bekommt der Nameserver diese Adresse, wenn er nicht für die Domain zuständig ist? Die Antwort ist einfach, der Nameserver muss rekursive Namensauflösung zulassen und er hangelt sich dann beginnend beim Root-Nameserver durch, bis er beim Nameserver für die Domain angekommen ist. Dort erhält er die korrekte Antwort und leitet diese weiter. Details dazu kann man im Artikel „DNS und BIND" [7] finden.

HTTP-Anfrage senden

Da die IP-Adresse des Servers jetzt bekannt ist, geht es fast analog zur Suche des Nameservers weiter. Zuerst wird die Routingtabelle analysiert, um herauszufinden, was der nächste Hop ist, also an welche MAC-Adresse das IP-Paket gesendet werden soll.

Die Routingtabelle wird bei Unix mit **netstat -r** ausgegeben, ein **-n** hilft dabei, die Namensauflösung zu unterdrücken. Bei Linux kann man auch einfach das route-Kommando verwenden, es liefert die gleiche Ausgabe.

```
$ netstat -rn
Kernel IP routing table
Destination   Gateway      Genmask          Flags   MSS Window  irtt Iface
10.0.0.0      0.0.0.0      255.255.255.0    U         0 0          0 eth0
0.0.0.0       10.0.0.3     0.0.0.0          UG        0 0          0 eth0
```

Es gibt hier nur zwei Einträge. Der erste betrifft das eigene Subnetz, d. h. alles, was im Bereich von 10.0.0.0 bis 10.0.0.255 liegt (der Bereich wird durch die Netzmaske festgelegt), wird direkt über **eth0** zugestellt. D. h. es wird ein ARP-Request direkt für das Zielsystem gesendet.

Der zweite Eintrag betrifft die Default-Route, d. h. für alle Adressen, für die es keinen Eintrag gibt, ist dieser zuständig. In unserem Fall ist der Zielserver 192.168.1.6, d. h. er muss über die Default-Route erreicht werden.

Bei Gateway steht nun der Router, der den Weg zum Ziel (hoffentlich) weiß. Folglich wird ein ARP-Request gesendet, um die MAC-Adresse zu der IP-Adresse 10.0.0.3 zu finden:

```
00:13:77:5b:25:ad > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: ↷
arp who-has 10.0.0.3 tell 10.0.0.1
00:0d:b9:1f:13:7e > 00:13:77:5b:25:ad, ethertype ARP (0x0806), length 60: ↷
arp reply 10.0.0.3 is-at 00:0d:b9:1f:13:7e
```

Der Rest läuft analog, nur werden die Pakete nun zum Router gesendet. Der nimmt diese entgegen und sucht das nächste Ziel. Dann werden die MAC-Adressen ausgetauscht, die TTL wird um eins reduziert und das Paket geht zum nächsten Router (aka. Hop).

Da HTTP in der Regel via TCP erfolgt, wird ein TCP-Paket gesendet:

```
00:13:77:5b:25:ad > 00:0d:b9:1f:13:7e, ethertype IPv4 (0x0800), length 74:↷
 10.0.0.1.47401 > 192.168.1.6.80: S 1287558560:1287558560(0) win 5840 <mss↷
 1460,sackOK,timestamp 1064977 0,nop,wscale 7>
```

Hier sieht man die MAC-Adressen, analog zur DNS-Anfrage. Die Ziel-MAC-Adresse ist die vom Router. Im IP-Header (oben nicht angezeigt) steht jetzt statt UDP, dass es sich um TCP handelt. Das kann man auch an dem **S** erkennen. Das ist ein SYN-Paket, der Client möchte eine Verbindung zum Server öffnen und bittet um SYNchronisation. Es folgen eine Sequence-Nummer und die Window-Größe, d. h. soviele Bytes darf einem der Server senden, bevor der Empfang bestätigt werden muss. Ferner gibt es noch ein paar Optionen. Hier besagen sie, dass SACK, selektives Bestätigen von Paketen, erlaubt ist, es werden Zeitstempel zur Bestimmung der Laufzeiten verwendet und mit **wscale** können größere Windows angegeben werden, als ursprünglich vom Protokoll erlaubt waren.

MSS gibt die Maximum Segement Size an, also die maximale Größe für die Payload, die Nutzdaten im TCP. Der IP-Header ist gewöhnlich 20 Bytes groß, der TCP-Header ebenfalls. Das macht bei Ethernet einen Wert von 1500-20-20=1460. Das ist die maximale Payload, die ein System senden darf, sie kann sich noch einmal etwas verringern, wenn die Gegenstelle Optionen von TCP verwendet, dann wird der TCP-Header größer. Das ist dann aber ein Problem der Gegenstelle, die das berücksichtigen muss.

Auf das SYN folgt dann diese Antwort:

```
00:0d:b9:1f:13:7e > 00:13:77:5b:25:↷
ad, ethertype IPv4 (0x0800), length↷
```

```
 74: 192.168.1.6.80 > ↷
10.0.0.1.47401: S ↷
1133845053:1133845053(0) ack ↷
1287558561 win 5792 <mss 1460,↷
sackOK,timestamp 388120 1064977,nop↷
,wscale 2>
```

Das sieht analog zu Obigem aus. Interessant ist hier zu sehen, dass als Absender wieder die MAC-Adresse des Routers auftaucht, obwohl die Antwort ja vom Webserver stammt. Der Server sendet seinerseit auch ein SYN, er will die Verbindung in Rückrichtung auch öffnen. Gleichzeitig sendet er auch ein ACK mit, damit bestätigt (ACKnowledge) er den Verbindungswunsch.

Die ACK-Nummer ist dabei die Sequenznummer plus eins vom Client. Danach kommen noch die TCP-Optionen vom Server. Dieser verwendet ebenfalls Zeitstempel. Hier sieht man nun zusätzlich den Zeitstempel des Clients, d. h. in der Antwort steckt die Uhrzeit, die der Client versendet hat. Darüber kann nun leicht die Laufzeit bestimmt werden, man muss sich also keine Gedanken darüber machen, ob die Zeiten synchron sind oder nicht. Es zählt immer nur die eigene Uhr! Um genau zu sein, braucht die Uhrzeit auch nicht genau sein, es zählt nur die Zeit, bis man diesen Wert wiedersieht.

Was hat es nun mit den Sequenz- und Acknummern auf sich? Ganz einfach, darüber wird geklärt, wo die Daten einzuordnen sind. Im Internet ist es zum Beispiel nicht gewährleistet, dass die Pakete in der richtigen Reihenfolge oder nur einmal aufschlagen.

Der Trick ist nun, dass die Sequenznummer anzeigt, wo im Datenpaket die folgenden Bytes liegen, es ist der Startwert. Die Acknowledge-Nummer gibt wiederum an, welche Daten man von der Gegenstelle erhalten hat. Dabei zeigt die ACK-Nummer auf das erste Byte, das noch fehlt, also die Seq-Nr. aus dem SYN-Paket plus die bereits empfangenen Daten plus eins.

tcpdump ist so freundlich und zieht diese Startzahlen bei der Darstellung gleich ab, dann braucht man selber nicht mehr rechnen und weiß, wieviele Bytes angekommen sind.

Das dritte Paket sieht, etwas gekürzt, so aus:

```
IP 10.0.0.1.47401 > 192.168.1.6.80:↷
 . ack 1 win 46 <nop,nop,timestamp ↷
1064978 388120>
```

Hier bedeutet **ack 1**, dass noch gar keine Bytes empfangen wurden. Das ist klar, in den SYN-Paketen sind auch keine Daten enthalten, das sind nur IP-Pakete mit TCP-Header.

Ab dieser Stelle ist der sogenannte TCP-Handshake abgeschlossen, diese Verbindung ist nun aktiv. Um es zu wiederholen:

1. Client sendet SYN
2. Server antwortet mit SYN und ACK (für das SYN des Clients)
3. Client antwortet mit ACK (für das SYN des Servers)

Jetzt steht die TCP-Verbindung, es sind aber noch keine Nutzdaten übertragen worden. Das folgt im vierten Paket, hier fließen dann tatsächlich Daten:

```
IP 10.0.0.1.47401 > 192.168.1.6.80:↷
 P 1:176(175) ack 1 win 46 <nop,nop↷
,timestamp 1064978 388120>
```

Hier sieht man, dass 175 Bytes übermittelt wurden. Man sieht aber auch noch mehr, ein Push-Flag. Das ist die Aufforderung an die Gegenstelle, dass die Daten nicht mehr gesammelt werden sollen. Der Kernel reicht diese daraufhin an die Anwendung weiter.

Man kann es sich jetzt denken: In dem Paket ist der vollständige HTTP-Request enthalten. Mehr Daten will der Client jetzt noch nicht senden, nun ist erst einmal der Server an der Reihe. Schaut man sich das Paket genauer an (Option **-X** bei **tcpdump**), so sieht man (auf der nächsten Seite) den HTTP-Request im TCP-Dump.

Links ist die Byte-Nummer des Zeilenanfangs, dann kommen die Rohdaten des IP-Paketes in Hexadezimaldarstellung und rechts findet man die ASCII-Darstellung der Zeichen. Punkte sind dabei nicht darstellbare Zeichen oder auch Leerzeichen. Zieht man den IP-Header und den TCP-Header ab, dann bekommt man:

```
GET / HTTP/1.1
Connection: close
Accept-Charset: utf-8,*;q=0.8
Accept-Encoding: gzip
Host: www.lug-erding.de
Referer: http://www.lug-erding.de/
User-Agent: Dillo/2.2
```

```
21:31:00.912680 IP 10.0.0.1.47401 > 192.168.1.6.80: P 1:176(175) ack 1 win 46
<nop,nop,timestamp 1064978 388120>
0x0000:  4500 00e3 d5b3 4000 4006 98b2 0a00 0001  E.....@.@.......
0x0010:  c0a8 0106 b929 0050 4cbe 95a1 4395 1a3e  .....).PL...C..>
0x0020:  8018 002e cc84 0000 0101 080a 0010 4012  ..............@.
0x0030:  0005 ec18 4745 5420 2f20 4854 5450 2f31  ....GET./.HTTP/1
0x0040:  2e31 0d0a 436f 6e6e 6563 7469 6f6e 3a20  .1..Connection:.
0x0050:  636c 6f73 650d 0a41 6363 6570 742d 4368  close..Accept-Ch
0x0060:  6172 7365 743a 2075 7466 2d38 2c2a 3b71  arset:.utf-8,*;q
0x0070:  3d30 2e38 0d0a 4163 6365 7074 2d45 6e63  =0.8..Accept-Enc
0x0080:  6f64 696e 673a 2067 7a69 700d 0a48 6f73  oding:.gzip..Hos
0x0090:  743a 2077 7777 2e6c 7567 2d65 7264 696e  t:.www.lug-erdin
0x00a0:  672e 6465 0d0a 5265 6665 7265 723a 2068  g.de..Referer:.h
0x00b0:  7474 703a 2f2f 7777 772e 6c75 672d 6572  ttp://www.lug-er
0x00c0:  6469 6e67 2e64 652f 0d0a 5573 6572 2d41  ding.de/..User-A
0x00d0:  6765 6e74 3a20 4469 6c6c 6f2f 322e 320d  gent:.Dillo/2.2.
0x00e0:  0a0d 0a                                  ...
```

Das ist also der vollständige HTTP-Request. Wer Details dazu braucht, der wird im Artikel „HTTP und Squid“ [8] mehr finden.

Was man im obigen Dump auch gut sieht, ist die Sequenz **0d0a0d0a** ganz am Ende. Schaut man in der ASCII-Tabelle nach (Tipp: **man ascii**), so findet man dort:

```
012   10   0A   LF   '\n' (new line)
015   13   0D   CR   '\r' (carriage ret)
```

Der erste Wert ist oktal, also Basis 8, dann dezimal gefolgt von hexadezimal. Danach folgt das Zeichen: Zeilenvorlauf und Zeilenrücklauf, so wie es bei der guten alten Schreibmaschine war. Das ganze zweimal, einmal für den Zeilenabschluss und einmal für eine Leerzeile. Es wird hier also nicht, wie in Unix üblich, nur ein Newline für den Zeilenabschluss verwendet.

HTTP-Antwort empfangen

Nachdem der Client den Request abgesendet hat (Push-Flag, um dies dem anderen System anzudeuten), ist der Server an der Reihe. Zuerst wird der HTTP-Request bestätigt:

```
IP 192.168.1.6.80 > 10.0.0.1.47401:↷
 . ack 176 win 1716 <nop,nop,↷
timestamp 388120 1064978>
```

Das heißt der Kernel reicht nun die Daten (den HTTP-Request) an den Webserver weiter. Dieser analysiert ihn und antwortet darauf (die TCP-Optionen wurden weggelassen, da sie hier keine wichtigen Informationen mehr enthalten):

```
IP 192.168.1.6.80 > 10.0.0.1.47401:↷
 . 1:1449(1448) ack 176 win 1716
IP 10.0.0.1.47401 > 192.168.1.6.80:↷
 . ack 1449 win 69
IP 192.168.1.6.80 > 10.0.0.1.47401:↷
 . 1449:2897(1448) ack 176 win 1716
IP 10.0.0.1.47401 > 192.168.1.6.80:↷
 . ack 2897 win 91
IP 192.168.1.6.80 > 10.0.0.1.47401:↷
 . 2897:4345(1448) ack 176 win 1716
IP 10.0.0.1.47401 > 192.168.1.6.80:↷
 . ack 4345 win 114
IP 192.168.1.6.80 > 10.0.0.1.47401:↷
 FP 4345:4604(259) ack 176 win 1716
IP 10.0.0.1.47402 > 192.168.1.6.80:↷
 S 1285270109:1285270109(0) win ↷
5840
IP 192.168.1.6.80 > 10.0.0.1.47402:↷
 S 1135847585:1135847585(0) ack ↷
1285270110 win 5792
IP 10.0.0.1.47402 > 192.168.1.6.80:↷
 . ack 1 win 46
IP 10.0.0.1.47403 > 192.168.1.6.80:↷
 S 1290686861:1290686861(0) win ↷
5840
IP 192.168.1.6.80 > 10.0.0.1.47403:↷
 S 1138145597:1138145597(0) ack ↷
1290686862 win 5792
IP 10.0.0.1.47403 > 192.168.1.6.80:↷
 . ack 1 win 46
IP 10.0.0.1.47401 > 192.168.1.6.80:↷
 F 176:176(0) ack 4605 win 137
IP 10.0.0.1.47402 > 192.168.1.6.80:↷
 P 1:188(187) ack 1 win 46
...
```

Hier kann man noch etwas beobachten: Erst werden die Daten gesendet und auch bestätigt. Dann schließt der Webserver die Verbindung mit einem FP, aber der Client macht gleich darauf noch zwei Verbindungen auf, einmal mit dem Source-Port 47402 und einmal mit 47403.

Warum macht er das? Ganz einfach: Nachdem die **index.html**-Datei übertragen worden war, hat der Client diese analysiert und gesehen, dass da noch mehr Elemente nachgeladen werden müssen. Dafür öffnet er dann mehrere Verbindungen, um die Daten parallel zu laden. Das könnten auch andere Server sein, dann wäre es vermutlich deutlich schneller.

Mit **tcpdump** kann man aber auch gezielt eine Verbindung heraussuchen, z. B. durch Angabe des Source-Ports:

```
$ tcpdump -n -r lug.pcap port 47401
```

```
IP 10.0.0.1.47401 > 192.168.1.6.80: S 1287558560:1287558560(0) win 5840
IP 192.168.1.6.80 > 10.0.0.1.47401: S 1133845053:1133845053(0) ack ↷
1287558561 win 5792
IP 10.0.0.1.47401 > 192.168.1.6.80: . ack 1 win 46
IP 10.0.0.1.47401 > 192.168.1.6.80: P 1:176(175) ack 1 win 46
IP 192.168.1.6.80 > 10.0.0.1.47401: . ack 176 win 1716
IP 192.168.1.6.80 > 10.0.0.1.47401: . 1:1449(1448) ack 176 win 1716
IP 10.0.0.1.47401 > 192.168.1.6.80: . ack 1449 win 69
IP 192.168.1.6.80 > 10.0.0.1.47401: . 1449:2897(1448) ack 176 win 1716
IP 10.0.0.1.47401 > 192.168.1.6.80: . ack 2897 win 91
IP 192.168.1.6.80 > 10.0.0.1.47401: . 2897:4345(1448) ack 176 win 1716
IP 10.0.0.1.47401 > 192.168.1.6.80: . ack 4345 win 114
IP 192.168.1.6.80 > 10.0.0.1.47401: FP 4345:4604(259) ack 176 win 1716
IP 10.0.0.1.47401 > 192.168.1.6.80: F 176:176(0) ack 4605 win 137
IP 192.168.1.6.80 > 10.0.0.1.47401: . ack 177 win 1716
```

Jetzt sieht man auch, wie die Verbindung abgebaut wird. Es können zwischen drei und vier Pakete sein:

1. FIN + ACK, hier noch mit Push-Flag
2. ACK des FINs -> half-closed Verbindung
3. FIN + ACK von der Gegenstelle
4. ACK, letztes FIN bestätigen, Verbindung ist zu

Im obigen Fall erfolgt 2. und 3. in einem Paket. Es ist aber auch möglich, eine Verbindung halboffen zu halten, dann kann nur noch eine Seite senden Die Payload enthält den HTTP-Header der Antwort des Servers (Ausgabe rechts oben).

Danach folgt eine Leerzeile und Binärdaten. Der Server hat den Inhalt der Webseite mit gzip (Content-Encoding: gzip) gesendet. Wer sich an den Request erinnert, da stand u. a.:

```
Accept-Encoding: gzip
```

```
HTTP/1.1 200 OK
Vary: Accept-Encoding
Content-Encoding: gzip
Last-Modified: Thu, 30 Dec 2010 15:09:38
GMT
ETag: "1886677020"
Content-Type: text/html
Accept-Ranges: bytes
Content-Length: 4321
Connection: close
Date: Thu, 01 Jan 1970 01:09:41 GMT
Server: lighttpd/1.4.28
```

Daher wurde hier alles gepackt verschickt. Dabei ginge es auch anders, das deutet der Server mit

```
Vary: Accept-Encoding
```

an. Er könnte die Daten auch anders ausliefern, vermutlich nicht-gepackt.

Aus diesem Grund wurde auch noch **lug2.pcap** [9] erstellt, da sieht man dann die Daten vom Server in der Antwort ungepackt.

Wer sich über das Datum wundert: Der 1.1.1970 ist der Start der Unix-Zeit, er entspricht 0 Sekunden. Von dieser Zeit an wird die Zeit in Sekunden gezählt. Der Webserver hier war aber mein DockStar. Scheinbar kann der sich die Uhrzeit nicht merken und fängt dadurch beim Einschalten bei 0, also beim 1.1.1970 an.

Weiteres Vorgehen im Browser

Der Browser analysiert die erhaltenen Daten. Dabei gibt der Content-Type Auskunft darüber, um was für Daten es sich handelt. Bei reinen HTML-Dateien steht da z. B.:

```
Content-Type: text/html
```

Der Browser kann diese Daten direkt darstellen. Dazu analysiert er aber erst die Datei, ob da noch weitere Elemente vorliegen, so wie verwendete CSS-Dateien, eingebettete Links, Bilder, etc. Wenn alle Elemente vorliegen, fängt der Browser an zu rendern, d. h. er versucht die Elemente in der angegebenen Weise zu arrangieren, so dass sie darstellbar werden. Da hat dann z. B. die aktuelle Browsergröße einen Einfluss. Hier wird auch oft die meiste Zeit beim Surfen gewartet: Oft hilft deswegen auch keine schnellere Leitung.

Was passiert, wenn es sich um nicht-HTML-Dateien handelt? Dann schaut der Browser anhand des Content-Types nach, ob er ein Plug-in zur Darstellung hat. (**about:plugins** in der URL-Zeile des Firefox gibt hier Auskunft oder auch „*Bearbeiten → Einstellungen → Anwendungen*“.)

Ist das nicht der Fall, so wird in **/etc/mailcap** oder **~/.mailcap** nachgesehen, ob hier steht, welches Programm die Daten anzeigen kann. Firefox fragt dann aber in aller Regel nach, ob er das Programm auch verwenden soll, eine Alternative dazu, oder ob die Daten nur gespeichert werden sollen.

In **mailcap** steht z. B.:

```
application/pdf; /usr/bin/gv '%s' ...
```

D. h. bei PDF-Dateien würde er hier das Programm gv starten, um die Daten anzeigen zu lassen. Oft gibt es aber mehrere Varianten, z. B. auch xpdf oder Evince. Normalerweise wird die erste genommen, manche Programme schlagen das auch vor, bieten aber die Option, ein anderes auszuwählen.

Fehlersuche

Das praktische Vorgehen ist meistens ein Abwarten der Fehlermeldung im Browser. Hat man eine, so ist der Rest schon fast selbsterklärend. Oft sind es DNS-Probleme, d. h. der Name konnte nicht aufgelöst werden. Gelegentlich gibt es auch Routing-Probleme: Der Zielserver ist über das Routing nicht zu erreichen.

DNS kann am einfachsten mit dig getestet werden, z. B.:

```
$ dig www.lug-erding.de
```

Führt dies nicht zur IP-Adresse, so sollte man einmal versuchen, den ganzen DNS-Baum durchzuhangeln. D. h. man befragt die Root-Nameserver für den zuständigen **de**-Nameserver, den befragt man für den für **lug-erding.de** zuständigen Nameserver und diesen wiederum nach der IP-Adresse für den Namen **www.lug-erding.de**. Einzelne Nameserver können bei **dig** mit dem @-Zeichen angegeben werden, z. B.:

```
$ dig www.lug-erding.de @ns9.↷
nameserverservice.de
```

oder gleich mit der IP-Adresse:

```
$ dig www.lug-eding.de @85↷
.25.128.54
```

Das Durchhangeln kann dig aber mit der Option **+trace** selbständig machen:

```
$ dig +trace www.lug-erding.de
```

Wenn die IP-Adresse bekannt ist, kann man z. B. mit telnet testen, ob der Server einen offenen Port hat und reagiert, z. B.:

```
$ telnet www.lug-erding.de 80
```

Gibt es die Meldung

```
telnet: Unable to connect to remote↷
 host: Connection refused
```

so ist der Webserver nicht am Laufen. Da kann man dann wenig machen, es sei denn, es ist der eigene Webserver ...

Um festzustellen, ob es sich um ein Routing-Problem handelt, bietet sich das Programm traceroute an. Per Default sendet das Programm UDP-Pakete an das Zielsystem. Normalerweise sind es drei Pakete an die fortlaufenden Ports ab 33434.

Wie funktioniert das Auffinden der Router zum Ziel? – Ganz einfach über das TTL-Feld im IP-Header: Jeder Router reduziert diesen Wert um eins, bis das Paket am Ziel ist oder der Wert Null erreicht wird. Das soll verhindern, dass Pakete ewig im Kreis laufen. Bei einem Wert von Null passieren in der Regel zwei Dinge:

1. Das Paket wird verworfen.

2. Ein ICMP Time Exceeded wird an den Absender des Paketes gesendet, die Absende-IP-Adresse ist die des Routers, der das ICMP generiert.

traceroute spielt mit dieser TTL. Beim ersten Paket ist die TTL auf 1 gesetzt. D. h. der erste Router muss das Paket schon verwerfen und ein ICMP senden. Damit hat man schon den ersten hop. Danach wird die TTL nach und nach erhöht, bis man am Ziel angekommen ist oder wo auch immer die Pakete verloren gehen.

Das funktioniert recht gut, allerdings spielen nicht alle Router mit, nicht jeder generiert ein ICMP Time Exceeded-Paket. Dann werden nur Sterne (*) statt der Router-IP-Adresse angezeigt.

Für den ersten Test des Routings sollte man die Option **-n** verwenden. Dadurch werden die IP-Adressen angezeigt und es wird nicht versucht, den Namen per DNS zu ermitteln. Hat man einen problematischen Router gefunden, dann hilft der DNS-Name oft herauszufinden, wo er stehen mag. Eine whois-Abfrage kann auch den Provider liefern. Ob dieser dann aber überhaupt auf Beschwerden reagiert, ist eine andere Frage ...

traceroute kann statt UDP-Pakete auch ICMP-Pakete, wie sie ping verwendet, benutzen. Dazu ist die Option **-I** da. Die lokale Routingtabelle kann unter Unix einheitlich mit **netstat -r** ausgelesen werden. Bei Linux geht auch einfach das **route**-Kommando. Hilfreich ist auch hier oft die Option **-n**, sie schaltet wieder die Namensauflösung ab.

Den ARP-Cache kann man mit **arp -a** auflisten. Die Einträge werden aber in der Regel, sofern kein Datenverkehr mit der Adresse besteht, nach 15-45 Sekunden gelöscht, man muss schon schnell schauen. Auch hier kann die Option **-n** mit dem gleichen Effekt verwendet werden.

Wenn alle Stricke reißen, dann kann **tcpdump** eine gute Wahl sein. Da sind Fehler aber nur mit geübtem Auge zu erkennen und oft sieht man vor lauter Wald die Bäume nicht. Während tcpdump sich in der Regel bei den unteren Protokollen gut auskennt, können die höheren Protokolle mit dem Programm wireshark gut analysiert werden. Dafür empfiehlt es sich aber, die Daten mit tcpdump in eine Datei zu schreiben und als normaler Benutzer diese in wireshark einzulesen.

Der Grund dafür ist einfach: wireshark ist sehr mächtig und analysiert sehr viele Protokolle. Da ist es eigentlich normal, dass dort noch viele Bugs enthalten sind, die vielleicht durch speziell konfigurierte Pakete ausgenutzt werden könnten. Da ist es dann nicht klug, diese live als Benutzer root analysieren zu lassen. Zeitversetztes Einlesen der Datei in wireshark als normaler Benutzer entschärft das dann deutlich.

Fazit

Oben wurde gezeigt, was hinter einem einzelnen Mausklick im Browser alles passiert, welche Prozesse involviert sind – bis hinunter zur Paketebene.

Gewöhnlich verschwendet man daran keinen Gedanken: Es funktioniert ja auch in der Regel recht gut. Warum sollte man sich also dafür interessieren, was unter der Haube passiert? Die meisten werden es in der Regel nicht tun, geschweige denn sich einmal im Detail ansehen, was da wirklich passiert.

Auf der anderen Seite gibt einem das Verständnis aber ein wenig Sicherheit im Umgang mit den Internet-Diensten. Dieser Punkt wurde allerdings noch nicht betrachtet: Sicherheit! Da nun im Detail bekannt ist, was abläuft, kann man sich auch relativ einfach überlegen, wo überall etwas schief gehen kann, vor allem, wenn es einer mit Absicht macht.

Wer aufgepasst hat, der weiß jetzt auch, wie sinnlos Websperren per DNS sein können: Sie treffen nur diejenigen, die sich nicht auskennen. Sonst nimmt man einfach einen anderen Nameserver, z. B. den von Google: 8.8.8.8. Wie soll man dessen Antworten nicht nur filtern, sondern auch noch ändern?

Oder soll mit den DNS-Sperren sämtlicher DNS-Verkehr geblockt werden, außer zum lokalen Provider? Das dürfte extrem schwer zu rechtfertigen sein. Wobei die T-Home das durchaus macht(e): Da war zumindest damals eine Abfrage des Google-Nameservers geblockt.

Warum alles fast immer so einwandfrei läuft, ist recht einfach erklärt: Das Internet ist in den mehr als 40 Jahren, die es existiert, darin gewachsen. Aber dennoch können immer wieder Probleme

auftauchen, sie sind nur so selten, dass es einem meist nicht auffällt. Wenn mal eine Webseite nicht funktioniert, dann braucht man in der Regel nicht lange zu warten, bis das repariert wird.

Und in vielen Fällen gibt es Redundanzen. So schreibt das DeNIC sogar vor, dass man wenigstens zwei Nameserver haben muss. Idealerweise sollten die auch nicht auf dem gleichen System sein, sie sollten nach Möglichkeit auch in unterschiedlichen Netzsegmenten liegen.

Man kann man mit DNS auch noch so einiges mehr anstellen, so wie es z. B. Akamai tut: Man kann die Zugriffe auf Webserver beschleunigen. Ein Verfahren dabei ist, dass Akamai weltweit Webserver für ihre Kunden betreibt und den Traffic auf den nächstgelegenen Webserver umleitet.

Wie machen die das? Ein Verfahren ist nun leicht zu verstehen, z. B. das von **www.rtl2.de** [10]. Die Nameserveranfrage offenbart es:

```
www.rtl2.de.   3600   IN   CNAME   www.↷
rtl2.de.edgesuite.net.
```

Das ist nicht ungewöhnlich, die Namensauflösung wird auf den rechten Namen weitergeleitet. Das ist ein Nameserver von Akamai. Dieser sieht nun, woher der Client kommt. Das dürfte nämlich von der IP-Adresse des ISPs kommen.

Anhand dieser Adresse kann Akamai nun feststellen, in welcher Region der Anfragende (vermutlich) sitzen wird. Um ihm dann den nächstgelegenen Server, möglichst der mit der geringsten Load, zuzuordnen, gibt es einen zweiten CNAME:

```
www.rtl2.de.edgesuite.net. 21600 IN↷
  CNAME   a1195.g.akamai.net.
```

Für diese Namensauflösung muss nun ein Nameserver von Akamai befragt werden, der in der Nähe liegen sollte. Dieser weiß, welche Webserver die geringste Last haben, und liefert dessen IP-Adresse aus. Damit diese schnell reagieren können, ist die TTL des Nameserver-Eintrags recht kurz:

```
a1195.g.akamai.net.   20   IN   A   ↷
95.100.249.122
a1195.g.akamai.net.   20   IN   A   ↷
95.100.249.113
```

Das sind nur 20 Sekunden, nach dieser Zeit muss die Namensauflösung erneut angestoßen werden. Dann könnte z. B. die Antwort so aussehen:

```
a1195.g.akamai.net.   20   IN   A   ↷
77.67.20.18
a1195.g.akamai.net.   20   IN   A   ↷
77.67.20.41
```

Und offenbar muss sich das bei der Performance rechnen: Die zusätzlichen Namensauflösungen scheinen durch den Standortvorteil gerechtfertigt zu sein, zumindest ist Akamai gut im Geschäft ...

LINKS

[1] http://www.pro-linux.de/artikel/2/1509/webzugriff.html

[2] https://secure.wikimedia.org/wikipedia/de/wiki/Web_Proxy_Autodiscovery_Protocol

[3] http://standards.ieee.org/regauth/oui/oui.txt

[4] http://www.pro-linux.de/files/webzugriff/lug.pcap

[5] http://de.wikipedia.org/wiki/OSI-Modell#Schicht_2_.E2.80.93_Sicherungsschicht

[6] http://www.lug-erding.de/vortrag/ng.html#ARP

[7] http://www.lug-erding.de/artikel/DNS+BIND.html

[8] http://www.lug-erding.de/artikel/HTTPundSquid.html

[9] http://www.pro-linux.de/files/webzugriff/lug2.pcap

[10] http://www.rtl2.de/

Autoreninformation

Dirk Geschke (Webseite) ist Gründer der Linux User Group Erding. Im Rahmen von Gesprächen bei der LUG offenbarten sich einige Verständnislücken über die Funktion von Netzwerken und auch was bei einem Webzugriff tatsächlich alles passiert. Aus einem Vortrag zu diesem Thema ist dieser Artikel entstanden.

Diesen Artikel kommentieren

Compositing nach X11 – KDE Plasma auf dem Weg nach Wayland

von Martin Gräßlin

Wayland [1] gilt als möglicher Nachfolger der X11 Architektur. Die ersten Projekte, wie z. B. MeeGo, planen Veröffentlichungen mit Wayland und auch die großen traditionellen Desktopumgebungen planen die Unterstützung dieser neuen Architektur. In diesem Artikel wird die Transition auf Wayland am Fallbeispiel der KDE Plasma Workspaces betrachtet. Der Inhalt dieses Artikels wurde auch am diesjährigen Desktop Summit in Berlin als Präsentation [2] vorgestellt.

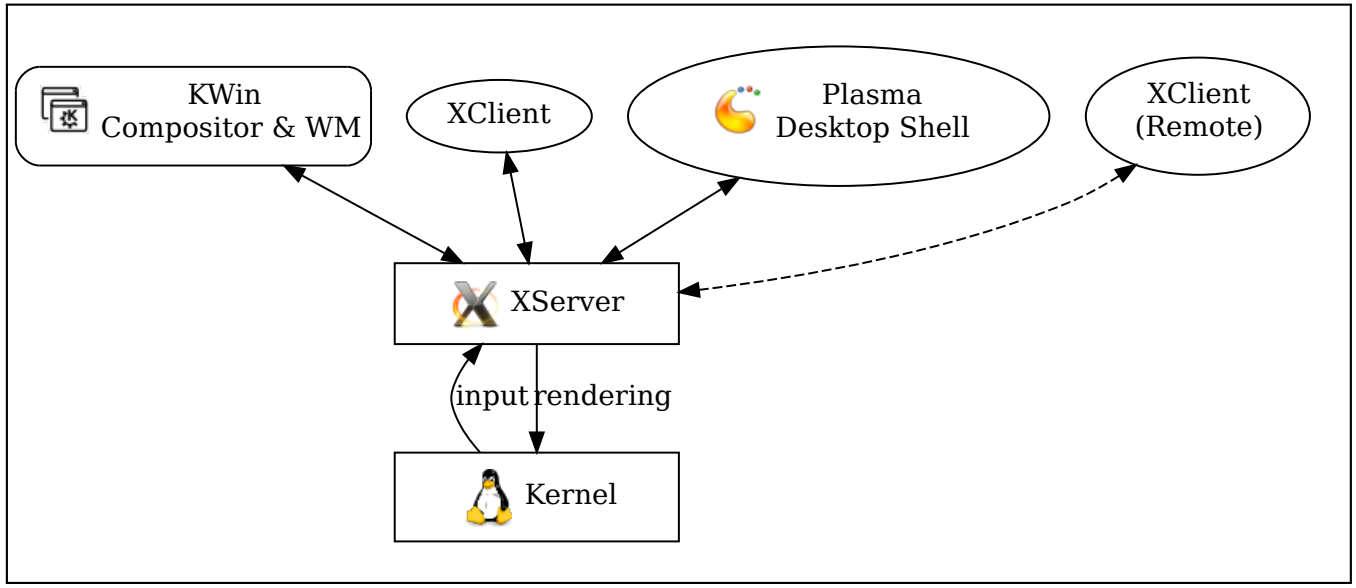
Die X11 Architektur

In freiesMagazin 03/2011 [3] wurde bereits die Architektur von X11 beleuchtet und warum es für die Zukunft nur die Lösung eines X freien Systems geben kann. X11 [4] ist eine Technologie aus den 80er Jahren des letzten Jahrhunderts, lange bevor irgend jemand an Anwendungsfälle wie Compositing [5] gedacht hat.

In einer modernen X11-Architektur, wie sie heute von allen Desktopumgebungen verwendet wird (siehe Skizze rechts), ist die Funktionalität des X-Servers auf die eines Proxys beschränkt. Der X-Server ist nicht mehr für das Zeichnen der Fenster zuständig. Dies wird komplett vom Compositor und Fenstermanager (z. B. KWin) übernommen. In der X11-Architektur kann der Compositor nicht direkt mit den X-Clients (den Fenstern) sprechen – alle Informationen werden durch den X-Server geleitet.

Diese Architektur schränkt die Möglichkeiten stark ein und erschwert die korrekte Implementierung. So werden zum Beispiel Maus- und Tastaturereignisse nicht durch den Compositor geleitet. Diese als „Input Redirection“ bekannte Funktionalität wäre aber sinnvoll, denn eigentlich entscheidet der Compositor, welches Fenster die Ereignisse erhalten soll. Insbesondere sind interaktive, transformierte Umgebungen dadurch nicht möglich. Man denke dabei an so triviale Anwendungsfälle wie ein Anwendungsstarter in einer Übersicht der virtuellen Desktops oder einem auf Lagesensor reagierenden Invertieren des Bildes bei einem Tablet (Beispiel-Video: [6]).

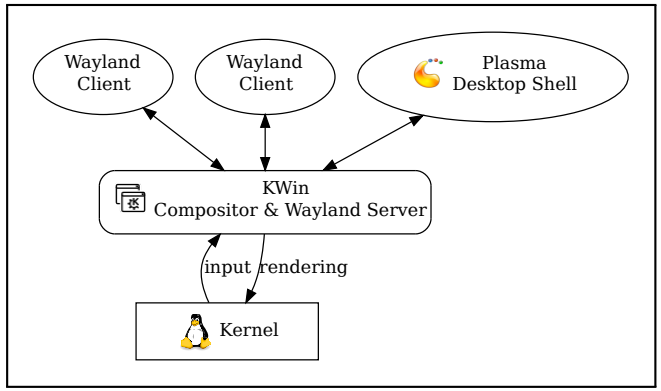
Generell entsteht ein Problem dadurch, dass der X-Server aus historischen Gründen noch für viele Funktionen verantwortlich ist, die eigentlich in den Compositor gehören. So pflegt der X-Server eine eigene „Stacking Order“ (Anordnung der Fenster [7]), obwohl alleine der Compositor über diese entscheidet. Der Compositor wäre eigentlich dafür verantwortlich, Richtlinien umzusetzen (z. B. Anordnung der Fenster, welches Fenster ist aktiv), kann dieses aber nicht, da die Funktion in X implementiert ist und jeder Client manuell den Zustand ändern kann. Dadurch entsteht ein dauernder Kampf zwischen Fenstermanager und Fenster, wie das Fenster aussehen soll.



Moderne X11 Architektur: X-Server als Proxy zwischen Compositor und Fenstern.

Die Wayland-Architektur

In der Wayland-Architektur ist der Proxy zwischen Anwendungen und Compositor entfernt. Der Compositor sitzt direkt auf der Hardware und nutzt diese zum Zeichnen und zum Empfangen von Eingabeereignissen. Es ist die Aufgabe des Compositors, die Eingabeereignisse an die Wayland-Clients (Fenster) weiterzuleiten. Die in der alten Architektur fehlende Input Redirection ist nun ein direkter Bestandteil der Architektur.



Mit Wayland wandert der Compositor in das Zentrum.

Die Verbindung zwischen Clients und Compositor ist auch denkbar einfach gehalten. Zur Kommunikation wird ein Unix-Socket verwendet und über diesen Socket werden Bufferinformationen ausgetauscht. Die Clients zeichnen in einen Buffer und informieren den Compositor über Änderungen zum vorhergehenden Buffer (Frame/Bild). Der Compositor wiederum informiert den Client, wenn ein Frame gezeichnet wurde, sodass Compositor und Clients synchron zeichnen können.

Ansonsten ist die Interaktion zwischen Clients und Compositor (noch) sehr gering. Es existiert noch kein Fenstermanagement-Protokoll und es ist fraglich, ob es jemals eines geben wird. Mit Wayland sind die Richtlinien zum Verwalten der Fenster komplett in den Compositor verlagert, wodurch vieles, was X11 ermöglichte, einfach überflüssig wird. Als Beispiel kann man den Zustand „minimiert“ verwenden: für den Client ist es eigentlich egal ob er minimiert ist, die einzige wichtige Information ist, wann er zuletzt gezeichnet wurde, was er sowieso erhält.

Wayland Entwicklungsstand

Aktuell ist die Entwicklung noch nicht so weit fortgeschritten, dass man an einen produktiven Einsatz auf einem Desktop denken kann. Vieles ist noch nicht spezifiziert oder noch nicht implementiert. Kaum ein aktuell verfügbares Toolkit unterstützt bereits Wayland. Die wichtigen Komponenten wie Unterstützung in den Grafikkartentreibern haben erst mit Mesa 7.11 (Release erfolgte im Juli) Einzug gehalten. Ohne diese ist der Einsatz unter Wayland noch undenkbar. Unterstützung in Qt wird ab Version 4.8 verfügbar sein, dies erfordert aber auch, dass Distributionen Wayland standardmäßig paketieren (was bisher noch nicht der Fall ist).

Im aktuellen Entwicklungsstand muss eine Anwendung über OpenGL ES 2.0 [8] zeichnen; „normales“ OpenGL [9] ist noch nicht vorgesehen, da dieses eine neue Schnittstelle ähnlich GLX [10] benötigen würde. Der Einsatz von GLX wird allgemein abgelehnt, da dies wieder eine X Abhängigkeit mit sich bringen würde. Dies bedeutet natürlich einen erheblichen Verlust an Funktionalität und reine OpenGL Anwendungen können nicht trivial einfach auf Wayland portiert werden.

Erschwerend kommt hinzu, dass bisher nur die freien Treiber das Projekt Wayland in Angriff genommen haben. Ob z. B. NVIDIA daran arbeitet, Wayland zu unterstützen, ist bisher nicht bekannt [11]. Nouveau [12] ist leider kein allgemein einsetzbarer Ersatz für den proprietären Treiber. Man denke hier an Anwendungsfälle wie garantierte Abwärtskompatibilität, Energieverwaltung, die Programmierschnittstelle CUDA [13] und patentierte Technologien, die nur im proprietären Treiber verfügbar sind.

Die für die Zukunft angedachte Möglichkeit einen X-Server unter Wayland zu betreiben, existiert auch nur in Gedanken. Ein kompletter Wechsel auf Wayland ist daher aktuell nur möglich, wenn man in Kauf nimmt, dass keine X-Anwendung mehr funktioniert. Dies ist ein akzeptabler Preis für mobile Einsatzgebiete wie MeeGo [14] oder KDE Plasma Active [15].

X-Server bleibt erhalten

Auf den Plattformen Desktop und Netbook kann von einem reinen Wayland-System daher auf absehbare Zeit keine Rede sein. Es gibt zu viele Anwendungen, die auf Legacy-Unterstützung angewiesen sind und bei denen auch keine Portierung auf Wayland zu erwarten ist.

Gerade auch die Tatsache, dass Wayland nicht auf allen Plattformen funktioniert (siehe

NVIDIA Treiber) macht deutlich, dass für die großen Desktopumgebungen wie die KDE Plasma Workspaces [16] vorerst X11 die erste Wahl bleiben muss. Eine zu frühe Umstellung auf Wayland würde Regressionen mit sich bringen und für die KDE Plasma Entwickler ist es das höchste Gebot, den Desktop nicht zu zerstören.

Auch nach einer erfolgten Umstellung auf Wayland muss die X11 Unterstützung erhalten bleiben. Auch wenn die meisten Anwender Wayland verwenden können, wird es auf lange Zeit noch Anwendungsfälle geben, die einen X-Server erfordern. Die Desktopumgebungen müssen daher in ihrer Entwicklung die Kompatibilität mit X berücksichtigen. Wird die Abwärtskompatibilität nicht berücksichtigt, so könnte im schlimmsten Fall Wayland von den Nutzern abgelehnt werden und sie bleiben bei X11.

Umstellung auf Wayland

Für die KDE Plasma Entwickler gibt es mehrere Möglichkeiten wie man Wayland angehen könnte. Diese sind:

- das Ignorieren von Wayland;
- das Ignorieren von X11;
- einen neuen, auf Wayland basierten Compositor und eine Desktop Shell parallel entwickeln;
- die schrittweise Migration auf Wayland.

Offensichtlich sind die ersten zwei Optionen nicht praktikabel. Wie in der Einleitung gezeigt, bietet Wayland eine verbesserte Architektur, von der die KDE Plasma Workspaces auch profitieren sollen. Die zweite Option ist nicht möglich, da wie im letzten Abschnitt gezeigt, der X-Server uns auf absehbare Zeit erhalten bleibt. Eine Einstellung der Entwicklung für X11 würde zu großen Akzeptanzproblemen unter den Nutzern führen.

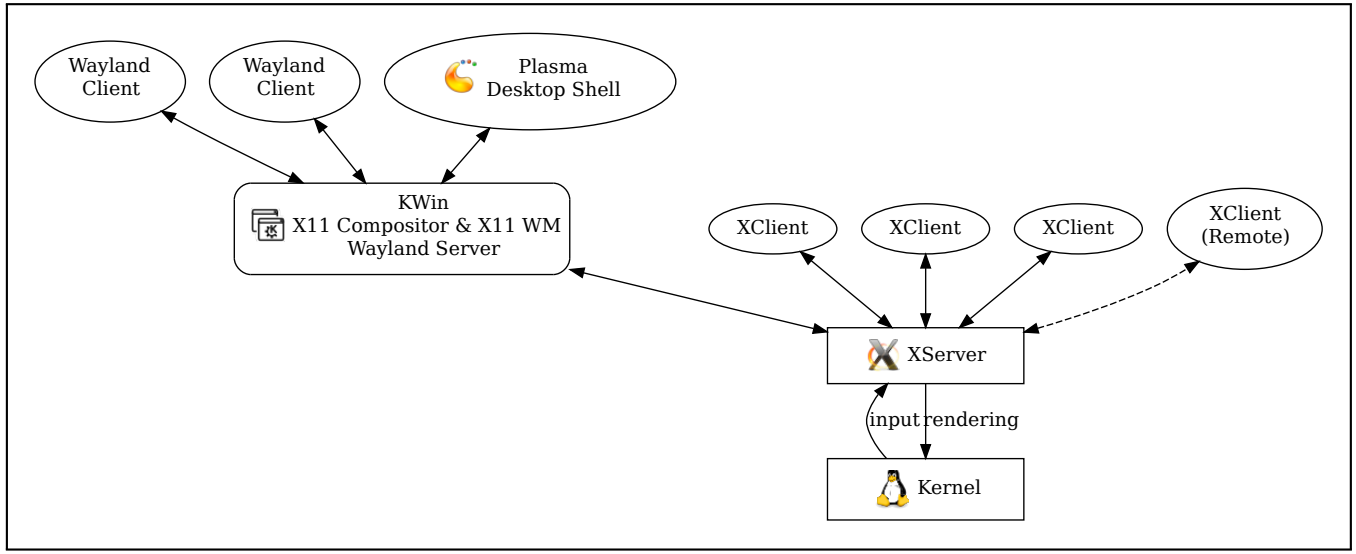
Auch die dritte Option ist nur schwer umzusetzen. Natürlich mag es verlockend klingen, eine alte Codebasis, zugeschnitten für ein nun obsoletes Fenstersystem, zu verwerfen. Jedoch stecken in KWin mehr als 12 Jahre Entwicklung und Fenstermanagement-Expertise. Durch Wayland ändert sich das grundlegende Verhalten jedoch nicht: ein Fenster ist immer noch ein Fenster. Zieht man noch die verfügbaren Entwicklerressourcen hinzu, wird offensichtlich, dass es nicht möglich ist, gleichzeitig an den X11 Workspaces zu entwickeln und nebenher noch einen neuen Wayland Workspace zu entwickeln.

Die zweite und dritte Option kommen auch mit dem großen Problem, dass es einen Tag X gibt an dem die allgemeine Umstellung von X11 auf Wayland erfolgen würde. Dies hätte vermutlich ähnliche Auswirkungen wie die Umstellung von KDE 3.5 auf KDE Plasma Workspaces. Vieles wäre unfertig und schlecht getestet, die Akzeptanz der Nutzer vermutlich eher schlecht. Nutzer würden lieber auf X11 bleiben, womit nichts gewonnen wäre.

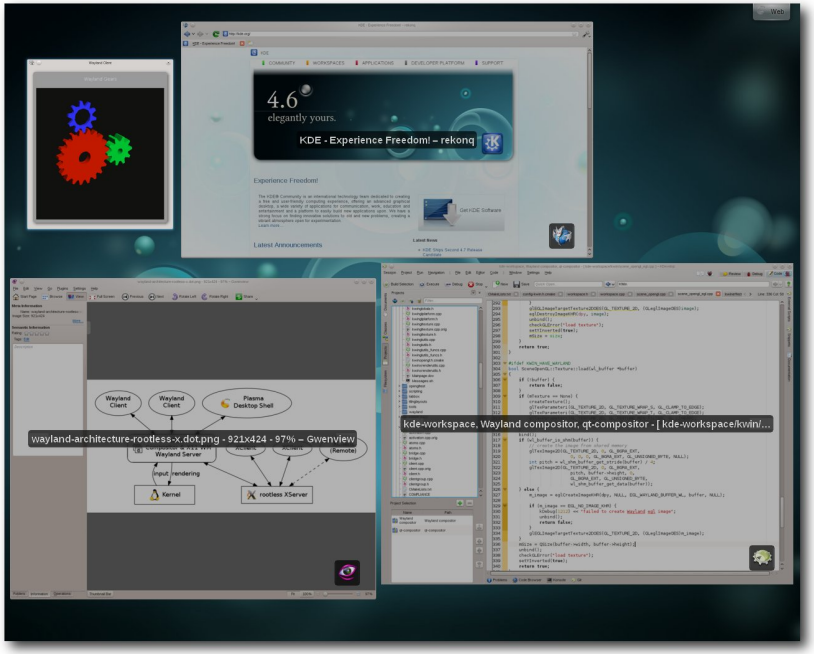
Die Umstellung der KDE Plasma Workspaces

Wie aus dem letzten Abschnitt ersichtlicht werden durfte, planen die KDE Plasma Entwickler keine direkte Umstellung auf Wayland. Die Portierung wird in drei Phasen erfolgen:

- Unterstützung von Wayland Clients unter X11;



Wayland Clients unter einem X11 Compositor.



Wayland Fenster (Zahnräder) wie ein normales X11 Fenster in den Compositor integriert.

- Wayland Clients unter Wayland;
- X11 Clients unter Wayland.

Aktuell arbeiten die Entwickler an der ersten Phase. Der Compositor und Fenstermanager KWin wurde um erste Wayland-Unterstützung erweitert [17]. Der Compositor kann im OpenGL ES 2.0/EGL-Backend Wayland Clients verwalten und in den Scenegraph integrieren. Jedoch hat jeder Wayland Client noch eine X11-Abhängigkeit. Jeder Client muss mit einer X11-Fensterdekoration versehen werden, um Eingabeereignisse im Compositor abfangen zu können (zur Erinnerung: X11 bietet keine Input Redirection) und an den Client weiterzuleiten.

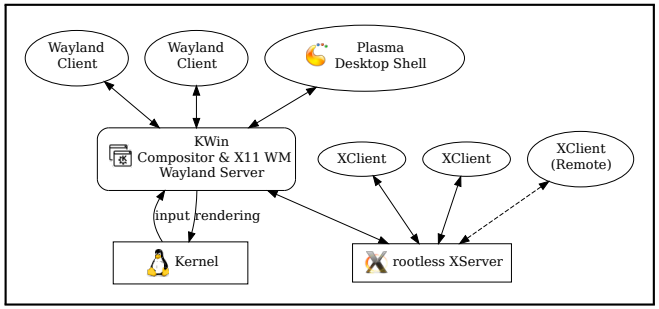
Aber nicht nur der Compositor muss Unterstützung erhalten. Auch der Plasma Desktop muss initialen Support erhalten, um z. B. Wayland Clients im Tasks Applet anzeigen zu können. Viele dieser Funktionen sind X11 spezifisch und müssen nun in generische Funktionen umgewandelt werden: die X11 Funktionalität muss in ein Backend ausgelagert werden.

Hierbei ist es wichtig zu wissen, dass im Gegensatz zu den KDE-Anwendungen (welche auch auf Microsoft Windows und Mac OS X portiert sind) der Workspace nie dazu vorgesehen war, unter einem anderen Fenstersystem als X11 zu funktionieren. Dank einer guten Abstraktionsschicht ist es jedoch gelungen, den Plasma-Desktop auch auf Microsoft Windows zu portieren und die ersten Ergebnisse der KWin-Portierung zeigen, dass auch dieses Projekt machbar ist. Hierbei wird das Verwalten der Fenster abstrahiert und die eigentliche Interaktion mit den Fenstern in Fenstersystem-spezifische Backends verlagert, ähnlich der bereits verwendeten Backend-Architektur im Compositing-Bereich [18] (KWin unterstützt XRender, OpenGL 1.x/GLX, OpenGL 2.x/GLX und OpenGL ES 2.0/EGL als Backends).

Bei der zweiten Phase geht es darum, einen Workspace ohne X11 Laufzeitabhängigkeit zu erstellen. Dieser würde dann ausschließlich Wayland Clients verwalten können. Dieser Entwicklungsschritt baut teilweise auf der ersten Phase auf. Voraussetzung hierfür ist, dass Wayland-Fenster bereits verwaltet und gezeichnet werden können. Nun ist es erforderlich, direkt auf der Hardware zum Rendern aufzusetzen und Inputereignisse weiterzuleiten, ohne den X-Server dazwischen zu haben.

Die Entwicklung der Phase 1 und 2 werden teilweise parallel erfolgen können. Auch hier hat bereits die Arbeit begonnen: mit Hilfe eines Google Summer of Code Projekts [19] wird der KWin Quellcode abstrahiert und die X11 Interaktion in ein Backend verlagert mit dem Ziel, Wayland Clients ohne X11 Abhängigkeit zu verwalten.

Die dritte Ausbaustufe ist erst von Interesse, wenn Phase eins und zwei umgesetzt sind. Hierbei geht es im Prinzip um ein „Umdrehen“ der ersten Phase. Anstatt Wayland-Clients unter X11, sollen nun X11-Clients unter Wayland verwaltet werden. Wie dieses umgesetzt werden kann,



X-Server unter einem Wayland Compositor.

ist noch nicht klar. Am wahrscheinlichsten sind die Optionen, das X11-Protokoll im Compositor umzusetzen oder einen „root-less“ X-Server zu starten.

Ziel der gesamten Entwicklung ist es, dass der Anwender niemals weiß, ob er unter X11 oder Wayland arbeitet und ob ein Fenster nun ein X11 oder Wayland Fenster ist. Von der Benutzung her sollen sich die Systeme nicht unterscheiden. Erst nach Abschluss der zweiten Phase kann man beginnen, neue, nur Wayland-spezifische, Erweiterungen einzubauen.

Ausblick

An dieser Stelle ist es nun angebracht, einen Ausblick zu liefern, wann die Anwender mit Wayland arbeiten werden können. Dies ist natürlich sehr schwierig. Wayland ist immer noch eine sehr junge Technologie und mögliche Probleme, welche die Entwicklung behindern könnten, sind noch nicht abzusehen.

Grundsätzlich plant die KDE Plasma Community, wie am Desktop Summit in Berlin vorgestellt, im Winterrelease 2012 die Phase eins bereits als Entwicklervorschau zu integrieren. Das Ziel ist es, Anwendungs- und Workspaceentwicklern etwas in die Hand zu geben, um ihre Anwendung unter Wayland zu testen. Natürlich ist das auch für interessierte Anwender eine Option, um sich früh mit den neuen Möglichkeiten vertraut zu machen, jedoch wird vom produktiven Einsatz von Wayland Clients zu diesem frühen Zeitpunkt abgeraten und die Entwickler werden noch keine Bugreports dafür annehmen.

Die zweite Ausbaustufe wird parallel gestartet und zielt auf die KDE Plasma Active Initative. Hierzu hatten die Entwickler bei ihrem Tokamak V Sprint [20] bereits Ende April sich als Ziel gesetzt, das zweite Release auf Wayland aufzubauen. Als mobile Plattform ist genauso wie für MeeGo der Verlust von, unter X11 bekannter, Funktionalität kein Problem und dieser Formfaktor profitiert am meisten durch ein Ausschalten des X-Servers.

Somit wird nach aktueller Planung das Sommerrelease 2012 es ermöglichen, einen X-freien Workspace zu betreiben. Jedoch wird dieser auf dem Desktop kaum einsetzbar sein, da noch zu viel fehlen wird. Wie lange es tatsächlich dauern wird, bis ein komplett einsetzbarer Wayland Desktop zur Verfügung steht, ist aktuell noch nicht absehbar.

LINKS

[1] http://wayland.freedesktop.org

[2] https://desktopsummit.org/program/sessions/compositing-after-x-kwin-road-wayland

[3] http://www.freiesmagazin.de/freiesMagazin-2011-03

[4] http://de.wikipedia.org/wiki/X_Window_System

[5] http://en.wikipedia.org/wiki/Compositing_manager

[6] http://www.youtube.com/watch?v=BrK4c7iFJLs

[7] http://en.wikipedia.org/wiki/Stacking_window_manager

[8] http://www.khronos.org/opengles/

[9] http://www.opengl.org/

[10] http://de.wikipedia.org/wiki/GLX

[11] http://www.nvnews.net/vbulletin/showthread.php?p=2343452#post2343452

[12] http://nouveau.freedesktop.org/wiki/

[13] http://de.wikipedia.org/wiki/Compute_Unified_Device_Architecture

[14] https://meego.com/

[15] http://community.kde.org/Plasma/Active

[16] http://kde.org/workspaces/

[17] http://blog.martin-graesslin.com/blog/2011/06/discovering-a-new-world/

[18] http://blog.martin-graesslin.com/blog/2011/05/the-compositing-modes-of-kde-plasma-workspaces-explained/

[19] http://www.google-melange.com/gsoc/project/google/gsoc2011/aarlt/12001

[20] http://vizzzion.org/blog/2011/04/tokamak-in-nijmegen/

Autoreninformation

Martin Gräßlin (Webseite) arbeitet als KWin Maintainer aktiv an der Portierung von KWin nach Wayland und hielt auf dem Desktop Summit 2011 einen Vortrag zu diesem Thema.

Diesen Artikel kommentieren

Der Juli im Kernelrückblick

von Mathias Menzer

Basis aller Distributionen ist der Linux-Kernel, der fortwährend weiterentwickelt wird. Welche Geräte in einem halben Jahr unterstützt werden und welche Funktionen neu hinzukommen, erfährt man, wenn man den aktuellen Entwickler-Kernel im Auge behält.

Als weiter Wegpunkt auf dem Weg zu Kernel 3.0 wurde Anfang Juli -rc6 [1] veröffentlicht. Ein großer Anteil der enthaltenen Änderungen betrafen den neuen Treiber isci für Intels C600 Chipsatz, der im Zusammenspiel mit Xeon-Prozessoren zum Einsatz kommt. Torvalds rechnete nicht damit, dass dieser Treiber Probleme machen sollte und dachte auch schon darüber nach, das Release folgen zu lassen. Dazu kam es dann jedoch erst einmal nicht, denn mit -rc7 [2] folgte eine weitere Vorabversion, da die Änderungen an RCU (Read-Copy-Update), das gleichzeitige Zugriffe auf Speicherbereiche regeln soll, immer noch Probleme im Zusammenspiel mit dem Scheduler aufwiesen. Diese konnten (hoffentlich) beseitigt werden, sodass Torvalds schließlich den Kernel 3.0 [3] freigab.

Linux 3.0

Während der Entwicklungsphase hat Torvalds es oft genug erwähnt, nun ist es offensichtlich: Linux 3.0 sticht nicht durch irgendwelche besonderen Funktionen heraus, auch wurde diesmal nicht mit alten Konzepten gebrochen, wenn man von der dreistelligen Versionsnummer einmal absieht. Den Vergleich mit 2.6.38 samt dessen Wunder-Patch oder mit 2.6.37, der erstmals ohne den Big Kernel Lock auskam, oder mit 2.6.29, bei dem erstmals Kernel Modesetting eingeführt wurde, kann der neue Kernel nicht standhalten, dafür verlief die Entwicklung jedoch recht ruhig, was auf einen guten Kernel mit wenig Ausbesserungen im Nachgang hoffen lässt. Daneben hat 3.0 auch einen neuen Namen erhalten: „Sneaky Weasel“, also gewieftes – oder auch hinterlistiges – Wiesel.

Ein paar Neuerungen hat jedoch auch auch Linux 3.0 parat: So defragmentiert Btrfs nun seine Partitionen automatisch. Dies ist notwendig, da Btrfs als Copy-on-Write-Dateisystem Daten nicht an die alte Stelle der Partition zurück- und damit überschreibt, sondern auf einem freien oder zumindest nicht mehr genutzten Teil ablegt. Die dadurch entstehende Verteilung eigentlich zusammengehöriger Daten, Fragmentierung genannt, wirkt sich auf die Geschwindigkeit beim Lesen des Datenträgers negativ aus, wogegen die Defragmentierung hilft. Dies kann manuell angestartet werden oder Mittels der Option `autodefrag` beim Einbinden der Partition während der Nutzung automatisch erfolgen. Weitere Verbesserungen bei Btrfs sind die Integritätsprüfung des Dateisystems mittels Scrubbing und schnelleres Löschen und Anlegen von Dateien. Netzwerkkommunikation kann mittels des neuen Syscalls `sendmmsg()` beschleunigt werden. Dabei reicht der eine Aufruf aus, um eine Menge an Daten zu versenden, für die ansonsten mehrere Aufrufe von `sendmsg()` notwendig wären. Das Konzept folgt `recvmmsg()`, das bereit in Kernel 2.6.33 eingeführt wurde.

Eine lange Geschichte hat der Paravirtualisierer Xen [4] aufzuweisen. Nachdem bereits seit Jahren Distributoren ihre Kernel entsprechend modifizieren oder angepasste Kernel in ihren Quellen anbieten, kann Linux nun auch von Haus aus als Host-System arbeiten, die Unterstützung für die privilegierte Domäne, im Xen-Jargon dom0 genannt, ist nun endlich in den Linux-Kernel eingezogen.

Fehlersuche im Netzwerk erfordert oftmals Zugriff auf Netzwerkpakete auf unterster Ebene im System, direkt bevor Bits und Bytes in elektrische Signale auf dem Netzwerkkabel gewandelt werden. Hierzu dient BPF (Berkeley Packet Filter [5]), der das heraussuchen der gewünschten Netzwerkpakete übernimmt, als Schnittstelle für entsprechende anwenderseitige Werkzeuge wie tcpdump. Dabei erhält BPF Anweisungen zum Filtern in einer eigenen Syntax. BPF hat nun einen Compiler erhalten, der die an BPF übergebenen Anweisungen beim Aufruf in Maschinen-Code (bislang nur für die x86-64-Architektur) umsetzt, welcher vom System schneller abgearbeitet werden kann als der bisherige BPF-Code und so speziell auf netzwerkseitig gut ausgelasteten Systemen eine Verbesserung der Leistung bringt.

Eine weitere kleine Neuerung ist Wake-on-WLAN, womit der PC aus dem Ruhezustand aus der Ferne geweckt werden kann. Hierbei bleibt der WLAN-Adapter aktiv und lauscht weiterhin auf eingehende Pakete. Das Ping-Kommando, mittels dem die Netzwerkverbindung zu einem anderen Rechner geprüft werden kann, kann nun auch ohne root-Rechte genutzt werden. Hierfür wurde ein neues Protokoll für Sockets eingeführt, das nur ICMP-Nachrichten verarbeiten kann. Natürlich muss das Ping-Kommando entsprechend angepasst sein, um Nutzen aus dieser Änderung ziehen zu können.

Wie immer sind die Änderungen auf Kernel Newbies in englischer Sprache übersichtlich aufgeführt [6].

LINKS

[1] https://lkml.org/lkml/2011/7/4/320

[2] https://lkml.org/lkml/2011/7/11/427

[3] https://lkml.org/lkml/2011/7/21/455

[4] https://secure.wikimedia.org/wikipedia/de/wiki/Xen

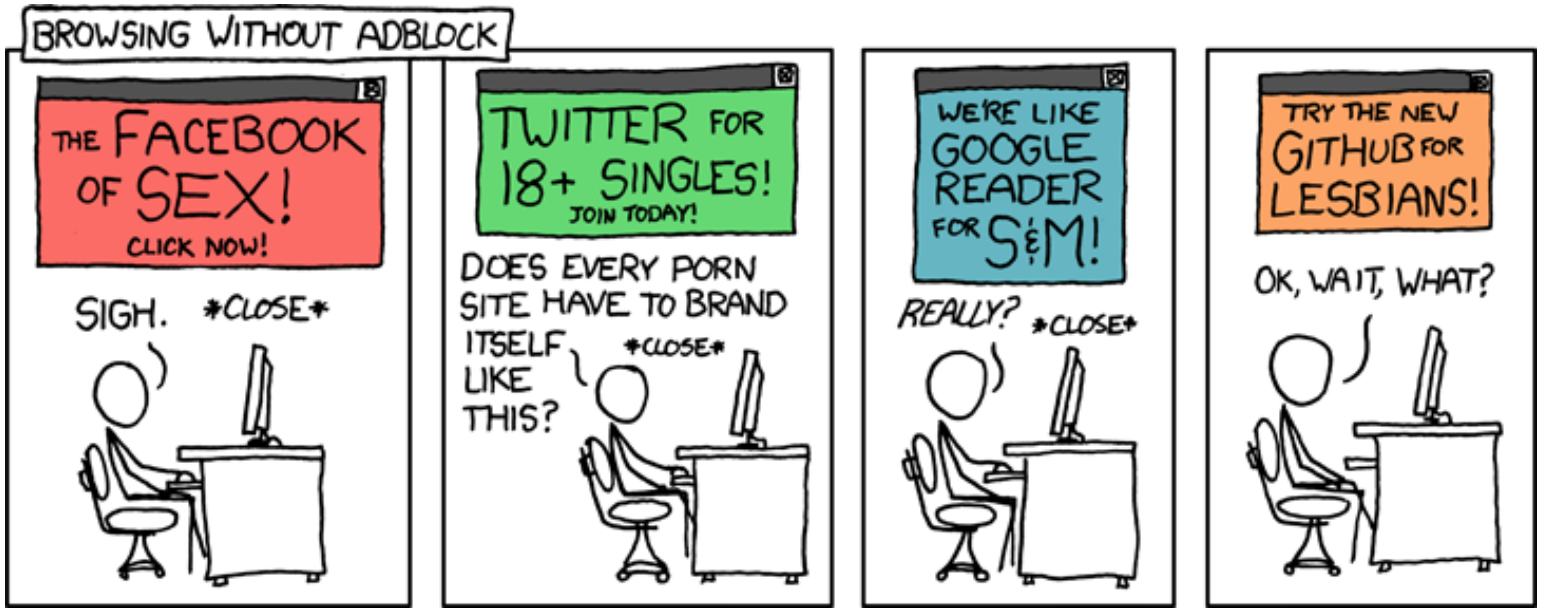
[5] https://secure.wikimedia.org/wikipedia/de/wiki/Berkeley_Filter

[6] http://kernelnewbies.org/LinuxChanges

Autoreninformation

Mathias Menzer (Webseite) hält einen Blick auf die Entwicklung des Linux-Kernels. Dafür erfährt er frühzeitig Details über neue Treiber und interessante Funktionen.

Diesen Artikel kommentieren



„Branding“ © by Randall Munroe (CC-BY-NC-2.5), http://xkcd.com/624

Perl-Tutorium: Teil 1 – Das erste Programm

von Herbert Breunung

Nachdem im vorigen Teil (freiesMagazin 07/2011 [1]) die Perl-Geschichte, Perl-Philosophie und Gemeinschaft der Nutzer vorgestellt wurde, beginnt jetzt die Reise zum ersten eigenen Perl-Programm. Nachdem geprüft ist, ob alle wichtigen Werkzeuge funktionstüchtig und griffbereit verpackt sind, geht es zum ersten Etappenziel: Skalare Variablen und einfache IO. Die weitere Route darf durch die Kommentarfunktion am Ende des Artikels mitbestimmt werden.

Wir brauchen Perl und ein Ziel

Wer dieses Magazin liest, tut das wohl vor seinem Bildschirm, der vom Linux seiner Wahl mit Pixeln versorgt wird. Selbst Mac-Nutzer haben wenigstens ein Unix unter der Haube und somit Perl bereits installiert. Wen das launige Schicksal jedoch ins Fensterland verschlug, bekommt mit Strawberry Perl [2] nach wenigen Klicks eine Arbeitsumgebung mit Perl, Make und C-Compiler, die es ihm erlaubt, diesem Tutorium vollständig zu folgen. Denn hier wird für die echte Praxis geübt, nicht für geschönte Realitätsausschnitte. Dieses Tutorium wird Folge für Folge ein brauchbares Programm aufbauen und dabei jeden Schritt dokumentieren, damit die Leser gut vorbereitet sind und in jeder digitalen Wildnis Ideen umsetzen können.

Außerdem macht es wesentlich mehr Spaß, ein Programm nach eigenen Wünschen anzupassen, als beinah sinnlose Codeschnipsel abzutippen. Das geplante Programm ist ein Notizbuch, weil es eine kleine, praktische Sache ist, die jeder ab und an gut gebrauchen kann und das Linux eigene Programm Note auf externe Editoren zurückgreift und nicht immer einfach zu bedienen ist. Je nach zugesandten Anregungen wird es am Ende zwitschern können oder es erlauben, dass Notizen von mehreren Nutzern in Echtzeit bearbeitet werden.

Der Anfang

Der Anfang ist jedoch ganz einfach, lediglich ein Rechner und etwas logisches Denkvermögen werden vorausgesetzt. Und natürlich ein aktuelles Perl, mindestens Version 5.12. Ein kurzes

```
$ perl -v
```

in der Kommandozeile (auch Shell oder Terminal) gibt Auskunft, was installiert ist. Wer jetzt 5.10 liest, wird an einigen Stellen kleinere Abstriche machen müssen. Aber spätestens mit einem 5.8.x sollte man über eine Aktualisierung nachdenken und die Paketverwaltung bemühen. Wer das nicht darf oder es nicht riskieren möchte, weil wichtige Programme oder eigene Projekte von Perl abhängig sind und er nicht wieder alle Module neu aufspielen will, sollte sich das Modul Perlbrew [3] installieren (lassen). Auch Menschen, welche die vielen Module, die während dieses Tutoriums vorgestellt werden, nur testweise installieren möchten oder die es nicht mögen, hinter dem Rücken der hauseigenen Paketverwaltung zu installieren, sollten den folgenden Abschnitt sorgfältig lesen, alle anderen dürfen ihn ignorieren.

Perl und Module installieren

Um Module zu installieren wird hier CPANMINUS [4] empfohlen, weil es nicht konfiguriert werden muss, sehr einfach zu bedienen ist und keine Ausgaben macht, die Anfänger verwirren könnten. Wer es nicht hat, wird noch einmal auf den Standardclient CPAN zurückgreifen müssen und kann sehen was mit „verwirrenden Ausgaben“ gemeint war:

```
# cpan App::cpanminus
```

Wenn die letzte Zeile der Ausgabe

```
/usr/bin/make install -- OK
```

lautet, weiß man, dass alles gut ging. Nun folgt im Terminal mit Root-Rechten:

```
# cpanm App::perlbrew
```

Jetzt kann man sich überzeugen, dass die darauf folgende Ausgabe wesentlich kompakter und verständlicher ist. Sie endet hoffentlich mit:

```
Successfully installed App-perlbrew↷
-0.27
1 distribution installed
```

Wer unter Unix rechtlich davon ausgeschlossen ist, außerhalb seines Homeverzeichnisses etwas

zu tun oder das Tutorium als Experiment sieht, welches er jederzeit nach **/dev/null** schicken kann, installiert Perlbrew lokal mit:

```
$ curl -L http://xrl.us/↷
perlbrewinstall | bash
```

Dies lädt das Skript **perlbrewinstall** herunter und leitet es an die Bash weiter, die es dann ausführt und damit Perlbrew installiert.

Damit die Brauerei ihren Betrieb aufnehmen kann, muss noch die **~/.bashrc** (oder die jeweilige Konfigurationsdatei der aktiven Shell) um Folgendes erweitert werden:

```
PATH="$HOME/perl5/perlbrew/bin:$PATH"
source perl5/perlbrew/etc/bashrc
```

Nutzer der C-Shell setzen **$PATH** mit **setenv** und setzen in die zweite Zeile ein **cshrc** statt dem **bashrc**.

Die letzten Schritte der Einrichtung sind sehr einfach und werden auch erklärt, wenn man nur **perlbrew** eingibt. Ähnlich zu **hg** oder **git** (zwei Versionsverwaltungsprogrammen) braucht es zuerst ein

```
$ perlbrew init
```

Das Verzeichnis, in dem man dies tut, ist nicht wichtig. Mit

```
$ perlbrew available
```

kann man sich auflisten lassen, was derzeit aktuell ist. Im Sommer 2011 wird dies auf ein

```
$ perlbrew install perl-5.14.1
```

hinauslaufen. Nun ist etwas Geduld gefragt, weil es einige Minuten dauert, Perl und seine Kernmodule zu laden, zu kompilieren und zu testen und Perlbrew dabei einfach nur schweigt.

Nach einem Neuaufruf der Shell mit **exec** oder einem neuen Terminalfenster und einem einmaligen

```
$ perlbrew switch 5.14.1
```

ist das aktuelle Perl endlich aktiv und wird immer gerufen, wenn man **perl** eingibt. (Der Interpreter wird klein geschrieben, die Sprache groß und PERL schreibt nur, wer es auf einen unfreundlichen Besuch aus der Perlgemeinde ankommen lassen will.)

Um zum System-Perl zu wechseln, genügt ein

```
$ perlbrew off
```

was sich jederzeit mit dem obigen **switch** wieder rückgängig machen lässt.

Dabei sollte man beachten, dass alles, was der Nutzer, der sich Perlbrew installierte, in der Shell tut, sich immer auf das aktive Perl bezieht, egal ob man Module installiert, Dokumentation liest oder Werkzeuge benutzt, welche mit Perl geliefert werden. Jedes nach dem **switch** geöffnete Terminal bezieht sich auf die lokale Installation (in **~/perl5/perlbrew/perls/perl-5.14.1**), auch nach einer Neuanmeldung oder Neustart des Systems.

Mein erstes Programm

Andere Einsteigertutorien nehmen sich mehr Zeit, das allererste Programm zu bejubeln. Klar, es ist es ein großartiges Gefühl mit einem simplen **print "Hallo ihr da draussen"** ein richtiges Programm geschrieben zu haben, wo man doch bisher Programmieren für eine schwarze Kunst gehalten hat. Diesen Moment der Begeisterung sollte jeder einmal genossen haben. Und wenn man dabei nicht nachdenken muss, was **#include <stdio.h>** oder **class HelloWorldApp** bedeutet, umso besser. Aber irgendwann meldet sich das Gehirn: „Und wozu ist das jetzt gut? Alle Perlmodule im Verzeichnis auszugeben hätte wenigstens noch einen praktischen Nutzen.“

```
$ perl -e 'print "tschuess\n"'
$ perl -E 'say <*.pm>'
```

print kennt man vielleicht aus Ruby, Python, Java oder PHP. Es gibt den ihm folgenden Wert, hier ein mit (doppelten) Anführungszeichen markierter Text, auf der Standardausgabe aus, was meist die aktive Shell ist, von der aus **perl** gestartet wurde. Die zweite Zeile gibt alle auf **.pm** endenden Dateien im aktuellen Verzeichnis aus, was meist Perl-Module sind. Jeder, der nicht zum ersten mal ein Terminal öffnet, kann das verstehen, aber kaum eine Sprache ermöglicht das so kompakt ohne zwei, drei andere Befehle aus dem Werkzeugkasten zu holen, die mit dem Problem nichts zu tun haben und vom Neuling auch erst einmal begriffen werden müssen.

Mit der Option **-e** oder **-E** führt **perl** sofort das in (hier in einfache) Anführungszeichen gestellte als Programm aus. Die großen Perl-Magier geben so direkt ihre Zaubersprüche ab, ohne Stab und Editor. Im Folgenden wird jedoch ein Editor benutzt, denn das Programm wird stark wachsen.

Die meisten Leser haben sicherlich bereits eine feste Meinung, welcher Editor der beste ist und die besseren bieten alles, was während dieses Tutoriums gebraucht wird. Unentschlossene könnten Padre [5] oder Kephra [6] probieren, welche beide in Perl geschrieben sind, also per **cpanm** beziehbar sind. Padre bietet mehr IDE-artige Funktionen, wohingegen Kephra eher auf Konsistenz und das schnelle Bearbeiten von Text ausgerichtet ist.

Das Projekt vorbereiten

Das neue Projekt bekommt am besten ein eigenes Verzeichnis und einen **alias** in der **.bashrc**:

```
alias bn='perl $HOME/code/perl/↷
betternote/bn.pl'
```

So kann es jederzeit einfach verwendet werden. Denn nur, wer öfters benutzt, was er entwickelt, merkt auch, was noch verbessert werden kann. Ebenso wird sich der Stolz auf das eigene Programm nur so voll zeigen. Datei- und Verzeichnisnamen sind dabei nur Vorschläge. Wichtig ist nur zu wissen, dass Perlskripte meist mit **perl Dateiname** aufgerufen werden:

```
$ perl ~/perl/script.pl
$ ~/perl/script.pl
```

Die zweite Variante kann aber nur gewählt werden, wenn mit **chmod** oder über den Dateibrowser (Rechtsklick auf die Datei und dann „*Eigenschaften → Zugriffsrechte*“) die Rechte der Datei auf ausführbar gesetzt wurden und die erste Zeile der Datei die sogenannte Shebang

```
#!/usr/bin/perl
```

enthält (siehe auch „Shebang – All der Kram“, freiesMagazin 11/2009 [7]).

Dies ist natürlich nur unter Unix sinnvoll und auch nur, wenn nicht Perlbrew verwendet wird. Deshalb verwendet der vorgeschlagene **alias** die erste Variante. Windows-Nutzer legen sich statt des Alias eine Verknüpfung der **.pl**-Datei auf den Desktop. Einfach die Datei auswählen (einfacher Linksklick), mit der rechten Maustaste das Kontextmenü aufklappen und eine Verknüpfung erstellen, die dann auf den Desktop ziehbar ist.

Perls Format

Bevor es endlich wirklich losgeht, noch ein paar allgemeine Regeln, die zu kennen alles sehr viel einfacher macht.

Regel 1: Leerzeichen spielen (fast) keine Rolle. Solange man sich nicht innerhalb der vielen Arten von Anführungszeichen befindet oder **print** als **pri nt** schreibt, ist es vollkommen gleich, wo und wie viele Leerzeichen oder Zeilenanfänge stehen.

```
      print

'huhu'     ;
```

Regel 2: Semikolons trennen die Befehle. Solange andere Befehle nicht eingreifen, arbeitet **perl** das Programm Befehl für Befehl von oben nach unten und links nach rechts ab. Da Zeilenenden nicht anzeigen können, wo ein Befehl aufhört, macht es das Semikolon. Auch wenn man nach dem letzten Befehl eines Programms oder Teilprogramms kein Semikolon setzen muss, empfiehlt Perl-Guru Damian Conway es trotzdem zu tun. Das beugt späteren Problemen vor, wenn man einen Befehl schnell mal in ein anderes Programm kopieren möchte. Sein Buch „Perl Best Practices“ enthält viele solcher nützlichen Hinweise und hat deshalb einen hohen Stellenwert bei vielen Perl-Programmierern.

Regel 3: Die Raute (**#**) leitet Kommentare ein. Wenn man etwas in das Programm schreiben möchte, das der Interpreter ignorieren soll, fügt man eine Raute ein und alles von diesem Zeichen bis zum nächsten Zeilenende gehört nicht zum ausgeführten Programm. Für längere Kommentare nimmt man **POD**, das später vorgestellt wird und ein **__END__** oder **__DATA__** markiert das Ende eines Programms.

```
say 'yes'; # und ich dachte Perl
           # sei schwer
say 'pound sign: #'; # ab hier ist
                     # Kommentar
```

Sag es einfach

Was macht eigentlich dieses **say**, was man hier schon zweimal sehen konnte? Es ist eigentlich nicht viel mehr als ein **print**, das noch ein Steuerzeichen anfügt, welches die Zeile beendet.

Die nächste Ausgabe beginnt dann in einer neuen. In Ruby und C nennt sich das **puts**, in Python funktioniert das **print** auf diese Art.

Das **say** ist sehr praktisch, denn es lässt sich sehr leicht tippen. Die Buchstaben s, a und y liegen auf der normalen QWERTZ-Tastatur nebeneinander. Außerdem ist es kürzer, spart bis zu sechs Anschläge und tut letztlich genau das, was man in den meisten Fällen mit **print** machen will. Es wurde allerdings erst mit Perl-Version 5.10 eingeführt und muss einzeln oder mit allen anderen Neuerungen angemeldet werden, mit denen ältere Programme vermeidbare Probleme haben könnten.

```
use feature 'say';
# oder
use v5.10;
```

Benutz es einfach

Neue Projekte sollten aber darauf nicht verzichten. Deshalb wird die erste Zeile des Notizprogramms nach der Shebang

```
use v5.12;
```

sein. Somit hat das Programm nicht nur Zugang zu allen neuen und interessanten Funktionen, es spart auch eine Zeile, die zuvor allen Anfängern in Perl-Foren eingebläut wurde und die man endlich nicht mehr so oft erwähnen muss, weil sie mit **use v5.12;** aufgerufen wird:

```
use strict;
```

Was dieser Befehl genau macht, erklärt der nächste Abschnitt. Einfach gesagt findet er für den Programmierer viele seiner Tippfehler.

Dass **use** „benutze“ heißt, weiß man vielleicht noch aus dem Englischunterricht. Aber in Perl kann es dreierlei bedeuten: Wenn eine Zahl folgt, prüft es die Version des laufenden **perl**. Ist es älter, wird sofort abgebrochen. Ab Version 5.10 aktiviert es, wie bereits beschrieben, alle neuen Funktionen bis zur angegebenen Version. Der vorhin verwendete Kommandozeilenparameter **-E** lädt alle verfügbaren neue Funktionen.

Folgt dem **use** ein Name, sucht **perl** nach einer gleichnamigen Datei die auf **.pm** endet und lädt sie als Modul. Manche sagen auch Bibliothek dazu. Da daher 95% der Macht aller Programme kommt, sollte der Befehl eigentlich noch vor **print** gelehrt werden.

Durch Gewohnheitsrecht hat sich durchgesetzt, dass Modulnamen in „CamelCase“ [8] also wie **SuperModul** geschrieben werden, aber mindestens mit einem Großbuchstaben beginnen. Steht nach **use** etwas Kleingeschriebenes, ist es kein Modul sondern ein „Pragma“, d. h. ein Befehl an Perl, sich anders zu verhalten. Nach **strict** ist das zweitnützlichste Pragma **warnings**, was **perl** veranlasst, ausführlichere Fehlermeldungen zu geben. Wer sich sich gerne belehren lässt, kann sogar **diagnostics** aktivieren. Dann werden zu den Warnungen die Erklärungen aus perldiag (eine Seite der Perl-Dokumentation) angezeigt.

Skalarvariablen

Selbst wer noch nie programmiert hat, kennt Variablen wohl vom Mathematikunterricht. Da sind es Buchstaben, die für eine Zahl stehen, die zu berechnen ist. Für Programmierer sieht das aber etwas anders aus. Hier sind Variablen eine Art Kiste (Speicherplatz), in die man jederzeit einen anderen Wert hineintun kann. (Außer in streng funktionalen Sprachen wie Haskell natürlich.) Die Namen der Kiste darf man in Perl beliebig wählen, solange sie mit einem Buchstaben anfangen. Bei Verwendung des Pragmas **utf8** kann man sogar Umlaute oder japanische Katagana für Variablen und andere eigene Konstrukte verwenden.

In vielen anderen Programmiersprachen gibt man den Kartons neben dem Namensschild noch eine Inhaltsbeschreibung und zeigt so an, ob man Zahlen oder doch lieber Kartoffeln in der Kiste abzulegen gedenkt. Bei Perl steckt diese Inhaltsangabe in einem Sonderzeichen, das vor dem Variablennamen steht: **$**, **@** oder **%**, wobei es in diesem Teil des Tutoriums nur um die einfachen gehen soll, die mit **$** beginnen. Das **$** sieht aus wie ein S und erinnert den Benutzer, dass es ein Skalar, also ein einfacher Wert, ist. Andere Sprachen unterscheiden da auch noch zwischen ganzen Zahlen, gebrochenen Zahlen, Buchstaben, Text und vielem mehr. Aber Perl ist das alles erst einmal egal.

```
$n = '5';
say $n + 3;
```

Damit wird ein kleiner Text (alles was in Anführungszeichen steht!) in die Kiste namens **$n** getan. = weist das Ergebnis der rechten Seite der linken zu. Eine Zeile später wird damit gerechnet. Da Perl weiß, was gemeint ist, lautet das Ergebnis natürlich 8. Manche Programmierer würden sich bei sowas die Haare raufen, weil das jede Computerlogik verletzt. Larry Wall versucht es aber „normalen“ Menschen einfach zu machen, in deren Alltag + sich auf Zahlen bezieht, nicht auf Schriftzeichen. Dann muss halt **perl** den Mehraufwand betreiben und prüfen, ob ein Wert auch als Zahl verstanden werden kann, nicht der Programmierer. Ein Text wird aber nur als Zahl deutbar, wenn er auch mit einer Ziffer beginnt. Ist die Zeichenkette leer (**”**), entspricht das einer **0**.

Wurde der Variablen beim ersten Erwähnen kein Wert gegeben, ist sie nicht leer, sondern hat keinen definierten Inhalt, was **$var = undef;** oder **undef $var;** entspricht. Bei ja/nein-Entscheidungen ist **”**, **0** oder **undef** negativ, alles andere positiv.

Strikte Regeln

So wie das letzte Beispiel dort steht, würde es einen Fehler hervorrufen. Wenn man eine Variable zum ersten Mal erwähnt, muss man dazuschreiben, in welchem Bereich sie bekannt ist. In der Informatik sagt man Gültigkeitsbereich. Das wären für den Anfang die Befehle **my** oder **our**:

```
my $notiz = 'Ich wollt nur mal Tach↷
 sagen.';
our $gruss = 'Tach!';
```

$notiz „lebt“ nur bis zur nächsten schließenden, geschweiften Klammer (nur im aktuellen Block), **$gruss** dagegen im ganzen „Modul“. (Das war etwas gemogelt, reicht aber als erste Vereinfachung.) Wäre kein **my** oder **our** angegeben, gäbe es diese Variable im ganzen Programm. Vor 20 Jahren, als Perlskripte klein waren, spielte das keine Rolle, aber heute wäre es wie das Fahren ohne Gurt. Man bräuchte nur ein Modul benutzen, dass ein Modul benutzt, in dem es auch eine Variable **$notiz** gibt, die während eines selbst ausgelösten Befehls verändert wird. Es könnten Wochen vergehen, bis so eine Problemursache überhaupt gefunden wird. Deshalb schränkt man den Geltungsbereich von Variablen möglichst stark ein und beide **$notiz** können koexistieren ohne voneinander Notiz zu nehmen. Genau dafür wurde **use strict;** erfunden. Es zwingt, immer einen Geltungsbereich zu bestimmen, was man sonst nicht müsste und verbietet damit globale Variablen (überall bekannte). Der angenehme Nebeneffekt davon ist, sollte man sich verschreiben (**$notitz**), gibt es einen harten Fehler (compile error) und das Programm bricht ab, bevor es ausgeführt wurde, weil vor dem **$notitz** kein **my** stand. Dabei gibt Perl die Zeilennummer samt Inhalt der schlimmen Stelle an und ein Fehler weniger konnte sich einschleichen.

IO „von Hand“

Bis jetzt ist das Programm sehr klein:

```
#!/usr/bin/perl
use v5.12;
use warnings;
# use diagnostics; # nur wer mag

print "Notiz: ";
my $notiz = readline STDIN;
```

Die letzte Programmzeile liest vom Benutzer ein, was immer er in die Shell tippt und mit Enter absegnet. Weil die Entertaste aber ein Zeichen abgibt, das die Zeile abschließt (das gleiche unsichtbare Zeichen, dass den Unterschied zwischen **print** und **say** ausmacht), kommt auch die ganze Zeile Text in **$notiz** an. **readline** (englisch für „lese Zeile“), tut dies auch und **STDIN** sagt von wo (der Standardeingabe, das in den meisten Betriebsystemen das Terminal ist, aber auch auf eine Braillezeile oder anderes umgeleitet sein kann). Perl wäre nicht Perl wenn das nicht auch kürzer ginge. Statt **readline** lässt sich auch der am Anfang gezeigte Diamantoperator <> nehmen. Nur wenn er ein Muster bekommt, sucht er damit die Dateinamen im aktuellen Verzeichnis ab. Und solange das Skript keine Parameter bekommt, ließe sich sogar **STDIN** weglassen:

```
my $notiz = <>;
chomp $notiz;
```

Da das Zeilenendzeichen in **$notiz** nicht wirklich gebraucht wird, schneidet man es mit **chomp** ab. Manche schreiben lieber beides in einer einzigen Zeile, was Geschmackssache ist:

```
chomp( my $notiz = <> );
```

Bereits die äußerlich sehr unterschiedlichen Arten eine Zeile einzulesen, zeigt die Wahlmöglichkeiten, für die Perl geliebt und abgelehnt wird und die im vorigen Teil angesprochen wurden.

Doch wohin mit der Notiz? Sie sollte bis zum nächsten Programmstart erhalten bleiben, was Variablen niemals leisten können. Dateien schon, da diese sicher auf der Festplatte lagern, zumindest solange kein großes Magnetfeld zu Besuch kommt.

Um eine Datei zu öffnen, verwendet man den Befehl **open**. Dieser bekommt drei mit Komma getrennte Information dahinter, die auch Parameter genannt werden: erstens das Handle, zweitens den Modus und drittens den Dateinamen. Dateinamen sind erst einmal normaler Text. Die möglichen Modi sind lesen (<), schreiben (>), anfügen (>>). Für Lese- und Schreibzugriffe stellt man ein + davor (+<). Aus den Pfeilzeichen lässt sich mehr Sinn ableiten, wenn man sich merkt, dass wenn die Spitze in den Dateinamen zeigt, damit schreiben gemeint ist und umgekehrt. Es lassen sich Modus und Dateiname auch als ein Text angeben, dies ist aber nicht empfohlen.

Wie Handle funktionieren, ist dem Leser schon bekannt, denn **STDIN** ist so eines. Diese Schreibweise (ohne **$**) zu verwenden, wird aber auch nicht empfohlen, weil es globale Variablen sind, die **use strict;** nicht verbieten kann. Deshalb sollte man Skalare nehmen, in denen die Handle gespeichert werden und die genau wie ein Handle verwendet werden. Sobald die Lebensdauer des Skalars erreicht ist, wird die Datei freigegeben, was auch jederzeit mit **close** geschehen kann. Um die Datei für einen Schreibzugriff vorzubereiten, muss Folgendes eingetippt werden:

```
open my $FH, '>', 'notizblock.txt';
print $FH $notiz;
close $FH;
```

print oder **say** sind eigentlich Universalwerkzeuge, um Text irgendwohin zu verschicken. Was bisher geschah war insgeheim ein:

```
print STDOUT $notiz;
```

STDOUT ist das Handle für die Standardausgabe, was auch meist nur das Terminal ist. Wichtig ist: Zwischen Handle und Text sollte man kein Komma setzen. Da **print** seine Nachricht auch in mehreren, durch Komma getrennten Happen erhalten kann, braucht es einen Weg, das Handle unterscheiden zu können. Die Nachricht wird wie folgt aus der Datei gelesen:

```
open my $FH, '<', 'notizblock.txt';
$notiz = <$FH>;
close $FH;
```

Statt **readline** (a.k.a. <>) geht auch **read**, um eine genau bestimmte Anzahl von Bytes zu lesen:

```
read( $FH, $notiz, 1);
```

Bei einem Zeichen wäre allerdings **getc** („get character“, zu deutsch „bekomme ein Zeichen“) kürzer, was vor allem für Abfragen wie **[J/N]** effektiv ist und das **chomp** spart:

```
$notiz = getc $FH;
```

Das Ergebnis vom **read** ist die Anzahl der Zeichen, die tatsächlich gelesen werden konnten und **eof $FH** kann melden, ob man schon am Ende der Datei angekommen ist (eof ist kurz für „end of file“).

Aufgabe

Aus allem Vorgestellten ein vollständiges Programm zu machen, soll die Hausaufgabe für das nächste Mal sein. Dafür fehlt allerdings noch eine Information. Mit dem obigen Aufruf übernimmt das Skript das aktuelle Arbeitsverzeichnis (**$CWD**) der Shell. Startet also jemand **perl projekte/note/bn.pl** von **E:\Perl** aus und im Code wird ein **open $FH, '<', 'notizblock.txt';** ausgeführt, sucht Perl **E:\Perl\notizblock.txt**. Die Textdatei sollte aber im Projektverzeichnis liegen und das Skript muss auch dort nachsehen, egal von wo aus gestartet. Dafür bring Perl das Kernmodul (ist immer dabei) **FindBin** mit, welches das ermöglicht:

```
use FindBin;
chdir $FindBin::Bin;
```

chdir wechselt das Arbeitsverzeichnis (wie **cd** in der Shell unter Linux und Windows) und **$FindBin::Bin** ist einfach eine mit **our** angemeldete Variable des Moduls, in der das gewünschte Verzeichnis gespeichert ist. Ansonsten bitte keines der jetzt gezeigten Module benutzen.

IO-Module

Auf die gezeigte Art können Dateien nur in Portionen (meist Zeilen) gelesen und geschrieben werden. Es sei denn, man kennt den Trick mit

```
my $notiz = do { local $/; <$FH> };
```

wodurch der ganze Dateinhalt in **$notiz** landet. Das ist aber auch nicht die perfekte Lösung. Zum Glück hat Uri Guttman bereits vor Jahren **File::Slurp** geschrieben. Dadurch geht alles sauber, knapp, in einem Befehl und ohne Handle:

```
use File::Slurp;

my $notiz = read_file( 'notizblock.txt' ); # lesen
write_file( 'notizblock.txt', $notiz );    # schreiben
append_file( 'notizblock.txt', $notiz );   # anhaengen
```

Da Golf (das kürzeste Programm gewinnt) der Sport der echten Perlprogrammierer ist, gibt es noch eine kürzere Lösung, dank des ebenfalls legendären Brian Ingerson. Sein **IO::All** kann alles, das irgendwie mit Ein- oder Ausgabe zu tun hat. Damit lässt sich sogar ein Uhrzeitserver mit einer Zeile Perl schreiben. Doch in diesem Teil interessiert nur lesen und schreiben von Dateien:

```
use IO::All;

$notiz < io('notizblock.txt');  # lesen
io('notizblock.txt') > $notiz;  # auch lesen
$notiz > io('notizblock.txt');  # schreiben
$notiz >> io('notizblock.txt'); # anhaengen

$notiz < io('-');               # STDIN lesen
```

Die offizielle Dokumentation

Mit Perl kommt eine Dokumentation, die recht gut ist und an der ab Version 5.14 wieder stärker gearbeitet wird. Eine erste Übersicht über die dort enthaltenen Themen listet

```
$ perldoc perl
```

auf. Die Hilfe zu **perldoc** selber liefert

```
$ perldoc perldoc
$ man perldoc
```

Die gleiche Dokumentation kann auch auf perldoc.org [9] nachgeschlagen werden.

Einen kleinen Teil der perldoc gibt es auch übersetzt im Wiki der perlcommunity.de [10]. Eine Liste mit fast sämtlichen anderen Perl-Ressourcen ist dort auch verlinkt [11].

Ausblick

Sobald man mehr als einen Wert hat, sollte man die Variablen kennen, die mit **@** beginnen. Die nächste Folge wird genau das behandeln, denn der Notizblock soll eine Liste an Notizen verwalten können. Auch die verschiedenen Formate, in denen sich Werte (Literale) deklarieren lassen, werden voraussichtlich erklärt.

Zusätzlich sei noch einmal daran erinnert, dass über die Kommentarfunktion am Ende des Artikels auch Wünsche zum weiteren Verlauf des Tutoriums geäußert werden können.

LINKS

[1] http://www.freiesmagazin.de/freiesMagazin-2011-07

[2] http://strawberryperl.com/

[3] http://www.perlbrew.pl/

[4] http://metacpan.org/module/App::cpanminus

[5] http://padre.perlide.org/

[6] http://kephra.sourceforge.net/site/de

[7] http://www.freiesmagazin.de/freiesMagazin-2009-11

[8] https://secure.wikimedia.org/wikipedia/de/wiki/Camelcase

[9] http://perldoc.org/

[10] http://wiki.perl-community.de/foswiki/bin/view/Perldoc/WebHome

[11] http://wiki.perl-community.de/foswiki/bin/view/Wissensbasis/PerlWebSites

Autoreninformation

Herbert Breunung (Webseite) ist seit sieben Jahren mit Antworten, Vorträgen, Wiki- und Zeitungsartikeln in der Perlgemeinschaft aktiv. Dies begann mit dem von ihm entworfenen Editor Kephra.

Diesen Artikel kommentieren

Variable Argumente in LaTeX nutzen

von Dominik Wagenführ

Dadurch, dass man in LaTeX eigene Befehle definieren kann, lassen sich wiederkehrende Aufgaben bzw. Formatierungen leicht umsetzen, ohne unnötig LaTeX-Code mehrfach schreiben zu müssen. Ein zweiter Vorteil bei einem eigenen Befehl besteht darin, dass man bei einer gewünschten Änderung nur eine Textstelle bearbeiten muss und nicht mehrere separate Stellen im Dokument. Der Artikel soll zeigen, wie man mithilfe des Paketes xkeyval [1] optionale Argumente bei selbstdefinierten LaTeX-Befehlen nutzen kann, ohne den Überblick zu verlieren.

Das Paket **xkeyval** ist dabei eine Erweiterung des Paketes **keyval** [2], mit welchem man einen Großteil der Beispiele des Artikels auch nachvollziehen kann. Daneben gibt es noch **pgfkeys** [3] und **kvoptions**/**kvsetkeys** [4] [5], die mit einer Schlüsselbehandlung bei Optionen umgehen können.

Befehlsdefinition

Eigene Befehle definiert man durch das Kommando **\newcommand**. Das folgende Minimalbeispiel wird im Laufe des Artikels um neue Befehlsdefinitionen erweitert:

```
\documentclass[parskip=half-]{%
scrartcl}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

% Blindtext zum Testen
\newcommand*\lorem{%
Lorem ipsum dolor sit amet,
consetetur sadipscing elitr, sed
diam nonumy eirmod tempor invidunt
ut labore et dolore magna aliquyam
erat, sed diam voluptua. At vero
eos et accusam et justo duo dolores
et ea rebum.}

\newcommand\framedtextA[1]{%
\fbox{#1}}

\begin{document}
\framedtextA{\lorem}
\end{document}
```

Listing 1: framedtextA.tex

Hier wird der Befehl **\framedtextA** definiert, der genau ein Argument erwartet (**[1]**) und dieses dann in einer simplen Box umrahmt darstellt. Mittels **#1** kann man dann auf das Argument zugreifen.

Hinweis: Damit man irgendetwas zum Darstellen hat, wird ein Blindtext über das Kommando **\lorem** definiert.

In der aktuellen Definition hat dies natürlich den Nachteil, dass **\fbox** die Zeilen nicht automatisch umbricht, sodass man den Inhalt besser in eine Minipage setzt und **\framedtextA** wie folgt korrigiert:

```
\usepackage{calc}

\newcommand\framedtextA[1]{%
\fbox{%
\begin{minipage}{%
\linewidth-2\fboxsep-2\fboxrule}%
#1%
\end{minipage}}}
```

Listing 2: framedtextA.tex (2. Version)

Damit die Box um die Minipage genau so breit ist, wie die aktuelle Zeile, wird von der aktuellen Zeilenlänge **\linewidth** die Rahmenstärke **\fboxrule** (auf beiden Seiten) sowie der Rahmenabstand **\fboxsep** (ebenfalls auf beiden Seiten) abgezogen. Damit die Rechnung funktioniert, benötigt man das LaTeX-Paket **calc** [6].

Als nächstes soll der Befehl erweitert werden, bis er so kompliziert bei der Benutzung wird, dass man sich eine Alternative wünscht.

Rahmenstärke und -abstand angeben

Als erste Erweiterung soll man die Rahmenstärke und den Abstand zum innenliegenden Text angeben können:

```
\newcommand\framedtextB[3]{%
\begingroup%
\setlength{\fboxrule}{#1}%
\setlength{\fboxsep}{#2}%
\fbox{%
\begin{minipage}{%
\linewidth-2\fboxsep-2\fboxrule}%
#3%
```

```
\end{minipage}}%
\endgroup}
```

Listing 3: framedtextB.tex

Die Benutzung im Dokument wäre dann:

```
\framedtextB{3pt}{10pt}{\lorem}
```

was den gleichen Text wie zuvor in einer Box darstellt, wobei der Rahmen aber eine Breite von 3 Punkt (3pt) hat und der Abstand zum Text 10 Punkte (10pt) beträgt.

Hinweis: Für die Befehle wird **\begingroup ... \endgroup** genutzt, damit die Veränderung von **\fboxrule** oder **\fboxsep** am Ende eines Befehls automatisch aufgehoben wird.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

*Die Box mit **\framedtextB**.*

Textfarbe ändern

Etwas bunter darf es ruhig zugehen, daher soll der Text farbig dargestellt werden. Dafür benötigt man das LaTeX-Paket **xcolor** [7]:

```
\usepackage{xcolor}

\newcommand\framedtextC[4]{%
\begingroup%
\setlength{\fboxrule}{#1}%
\setlength{\fboxsep}{#2}%
\fbox{%
\begin{minipage}{%
\linewidth-2\fboxsep-2\fboxrule}%
\textcolor{#3}{#4}%
\end{minipage}}%
\endgroup}
```

Listing 4: framedtextC.tex

Die Benutzung ist:

```
\framedtextC{3pt}{10pt}{red}{\lorem↷
}
```

Das erstellt den Text in der Box in roter Farbe.

Hinweis: Das Paket **xcolor** wird dem Paket **color** vorgezogen, da es eine weitaus bessere Farbunterstützung bietet und einen kleinen Fehler in **\fcolorbox** korrigiert.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

*Die Box mit **\framedtextC**.*

Rahmen- und Hintergrundfarbe ändern

Da der Befehl **\framedtextC** immer noch recht überschaubar ist, soll auch die Rahmenfarbe und die Hintergrundfarbe verändert werden können:

```
\newcommand\framedtextD[6]{%
\begingroup%
\setlength{\fboxrule}{#1}%
\setlength{\fboxsep}{#2}%
\fcolorbox{#4}{#5}{%
\begin{minipage}{%
\linewidth-2\fboxsep-2\fboxrule}%
\textcolor{#3}{#6}%
\end{minipage}}%
\endgroup}
```

Listing 5: framedtextD.tex

Die Benutzung ist nun:

```
\framedtextD{3pt}{10pt}{white}{↷
green}{black}{\lorem}
```

was den Text weiß auf schwarzem Hintergrund darstellt und dies alles grün umrahmt.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

*Die Box mit **\framedtextD**.*

Textausrichtung festlegen

Da die Darstellung im Blocksatz innerhalb der Box ziemlich langweilig ist, soll man von außen auch auf rechtsbündigen, linksbündigen oder zentrierten Text umschalten können.

Der Einfachheit halber wird dabei mit einem optionalen siebten Argument gearbeitet, welches man in eckigen Klammern nach der Anzahl der Argumente angibt. Für die Entscheidung, welche Ausrichtung gewählt wird, wird das Paket **xifthen** [8] benutzt:

```
\usepackage{xifthen}

\newcommand\framedtextE[7][b]{%
\begingroup%
\setlength{\fboxrule}{#2}%
\setlength{\fboxsep}{#3}%
\fcolorbox{#5}{#6}{%
\begin{minipage}{%
\linewidth-2\fboxsep-2\fboxrule}%
  \ifthenelse{\equal{#1}{r}}%
    {\flushright}{%
  \ifthenelse{\equal{#1}{l}}%
    {\flushleft}{%
  \ifthenelse{\equal{#1}{c}}%
    {\centering}{%
  \ifthenelse{\equal{#1}{b}}%
    {}%
    {\error{1}}}}}%
\textcolor{#4}{#7}%
\end{minipage}}%
\endgroup}
```

Listing 6: framedtextE.tex

Jetzt schwillt der Aufruf zu folgendem Konstrukt an, wenn der Text rechtsbündig ausgegeben werden soll:

```
\framedtextE[r]{3pt}{10pt}{black}{↷
black}{white}{\lorem}
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Die Box mit ***\framedtextE****.*

Dies kann man jetzt noch eine Weile fortführen, spätestens beim zehnten Argument stößt man aber auf eine natürliche Begrenzung von LaTeX, denn man wird dezent darauf hingewiesen, dass man bereits neun Parameter hat:

```
! You already have nine parameters
```

Und wer sich – ohne die Definition von **\framedtextE** anzuschauen – auch noch nach einem Monat gemerkt hat, welche Angabe die Text-, Rahmen- oder Hintergrundfarbe bzw. Rahmenstärke und -abstand ändert, hat ein wirklich gutes Gedächtnis.

Bei derartig vielen Parametern wäre es besser, wenn die Benutzung etwas intuitiver vonstatten ginge. So wäre es nicht schlecht, wenn man klare Begriffe mit einer Farbe assozieren könnte, oder wenn die Reihenfolge der Angaben beliebig ist. Und es ist auch nicht gerade sinnvoll, dass man grundsätzlich diese ganzen Angaben machen muss, selbst wenn man überall die Standardwerte nutzen möchte.

Hinweis: Für neue Umgebungen mittels **\newenvironment** gelten im Übrigen die gleichen Aussagen.

Argumente mithilfe von xkeyval

Glücklicherweise hat sich schon jemand Gedanken zu den obigen Wünschen gemacht. Mit dem LaTeX-Paket **xkeyval** kann man alle Probleme umgehen, die sich gestellt haben. Der folgende Abschnitt soll anhand von Beispielen die Benutzung des Paketes aufzeigen, sodass auch ein LaTeX-Anfänger relativ schnell zu einem Ergebnis kommt. Für eine ausführliche Beschreibung aller Optionen und Möglichkeiten sollte man sich die Dokumentation auf der Webseite durchlesen.

Bei der Benutzung von **xkeyval** werden die optionalen Argumente nach sogenannten „Schlüssel-Wert-Paaren“ durchsucht. Zu jedem Schlüssel wird dann der zugehörige Wert ausgelesen und der LaTeX-Schreiber kann den Wert entsprechend weiterverarbeiten. Manchmal wird dieser direkt an ein weiteres LaTeX-Kommando weitergegeben, manchmal aber auch erst zwischengespeichert, um später darauf zugreifen zu können.

Die Angabe eines Schlüssels und des zugehörigen Wertes geschieht dabei in der Regel über die Angabe **key=value**.

Farbangaben der Box ersetzen

Als Erstes sollen die drei Farbangaben für die Text-, Rahmen- und Hintergrundfarbe so geändert werden, dass man diese als optionales Argument angeben kann.

Als Basis für die Änderung wird die Version **\framedtextD** von oben benutzt. Die Möglichkeit zur Ausrichtung des Textes wird dann später wieder eingefügt.

Zuerst sollte man sich für die drei Farben drei neue Befehle definieren, die den Farbnamen enthalten und später überschrieben werden:

```
\newcommand*\TextColor{black}
\newcommand*\BackgroundColor{white}
\newcommand*\BorderColor{black}
```

Danach kann man auch schon mit der Definition der neuen Schlüssel beginnen:

```
\makeatletter
\define@key{TextBox}{textcolor}{%
\renewcommand*\TextColor{#1}}
\define@key{TextBox}{background}{%
\renewcommand*\BackgroundColor{#1}}
\define@key{TextBox}{bordercolor}{%
\renewcommand*\BorderColor{#1}}
\makeatother
```

Dies war es auch schon. Mittels **\define@key** definiert man also einen neuen Schlüssel. Das erste Argument (**TextBox**) gibt dabei die Familie an, die später für die Identifikation benutzt wird. Das zweite Argument steht für die Schlüsselbezeichnung. Als drittes folgt der LaTeX-Code, in dem man den Schlüssel verarbeitet, auf den man mittels des Argumentes **#1** zugreifen kann. Im obigen Fall werden die drei Werte also einfach nur in den vordefinierten Kommandos zwischengespeichert.

Der neue Befehl sieht dann wie folgt aus:

```
\newcommand\framedtextF[4][]{%
\setkeys{TextBox}{#1}%
\setlength{\fboxrule}{#2}%
\setlength{\fboxsep}{#3}%
\fcolorbox{\BorderColor}{%
\BackgroundColor}{%
\begin{minipage}{%
\linewidth-2\fboxsep-2\fboxrule}%
\textcolor{\TextColor}{#4}%
\end{minipage}}%
}
```

Listing 7: framedtextF.tex

Wichtig ist die Zeile

```
\setkeys{TextBox}{#1}%
```

denn über diese gibt man alle optionalen Argumente, die im ersten Argument (**#1**) stehen, an **xkeyval** zur Interpretation. Dieses sucht dann darin nach Schlüsseln und Werten und führt den zugehörigen Code aus. Hierbei ist es wichtig, die richtige Familie anzugeben, da sonst die Schlüssel nicht erkannt werden.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

*Die Box mit **\framedtextF**.*

Nach dem Setzen der Schlüssel kann man über die Kommandos **\BorderColor**, **\BackgroundColor** und **\TextColor** auf die Farbdefinitionen zugreifen.

Eine Benutzung sieht dann wie folgt aus:

```
\framedtextF[textcolor=white,↷
bordercolor=green,background=black↷
]{3pt}{10pt}{\lorem}
```

Dies ist zwar länger als zuvor, hat aber den Vorteil, dass aufgrund der Assoziation mit einem Schlüsselwort wie **textcolor** sofort klar ist, was das Argument beeinflusst. Daneben ist auch die Reihenfolge der Angaben nicht mehr relevant:

```
\framedtextF[background=black,↷
textcolor=white,bordercolor=green↷
]{3pt}{10pt}{\lorem}
```

erzeugt die gleiche Box.

Da es sich um optionale Werte handelt, kann man natürlich auch nur einen Teil angeben, beispielsweise

```
\framedtextF[textcolor=black]{3pt↷
}{10pt}{\lorem}
```

Das Ergebnis ist nicht wie erwartet, da der Text „verschwunden“ ist. Der Grund liegt darin, dass noch keine Standardwerte definiert wurden. Das heißt, fehlt ein optionales Argument, wird es derzeit nicht verändert und bei erneuter Anwendung von **\framedTextF** einfach der zuvor gesetzte

Wert beibehalten. Dadurch sieht man nun (neuen) schwarzen Text auf (altem) schwarzen Hintergrund. (Wobei etwas getrickst und die Gruppierung entfernt wurde, um diesen Effekt zu zeigen.)

Fehlende Schlüssel vorbelegen

Will man vorbelegte Standardwerte nutzen, wenn die Angabe einer Option komplett fehlt, fügt man folgende Zeile (am besten unter der Definition der Schlüssel) ein:

```
\presetkeys{TextBox}{%
bordercolor=black,textcolor=black,%
background=white}{}%
```

Erzeugt man nun das LaTeX-Dokument mit gleichem Code wie oben, wird der Hintergrund weiß angezeigt, sodass man den schwarzen Text wieder sehen kann. Wer nicht glaubt, dass das funktioniert, kann auch die Textfarbe auf Rot ändern:

```
\framedtextF[textcolor=red]{3pt}{10↷
pt}{\lorem}
```

Der Befehl **\presetkeys** definiert wie gesagt die Standardwerte, wenn eine Option nicht angegeben ist. Das erste Argument gibt wieder die Familie an. Das zweite und dritte Argument sind ähnlich, denn das zweite setzt die Standardwerte *bevor* die benutzerdefinierten Werte mit **\setkeys** gesetzt werden, das dritte Argumente setzt diese erst *danach*. Die Standardwerte werden dabei aber nur gesetzt, falls der Wert nicht als Option vom Benutzer angegeben wurde.

In obigem Beispiel hat die Unterscheidung, ob man die Standardwerte zuerst oder zuletzt setzt, keine Bedeutung. Dies ist aber beispielsweise dann wichtig, wenn man in einer Schlüsseldefinition auf einen anderen Schlüsselwert zugreifen will. Näheres dazu erfährt man in der Dokumentation.

Die Vorbelegung bleibt aktiv, bis sie mit **\presetkeys** erneut überschrieben wird. Das heißt, man kann in jedem Befehl vor **\setkeys** die Vorbelegung anders setzen, wenn das gewünscht ist.

Hinweis: Natürlich muss man **\presetkeys** nicht einsetzen, sondern kann auch selbst für eine Vorbelegung sorgen, indem man vor dem **\setkeys**-Befehl alle Werte selbst mit **\renewcommand** initialisiert. Alternativ kann man auch ein zweites **\setkeys** aufrufen, um Werte einmalig zu setzen.

Boxgrößen ändern

Wie oben bei den Farben soll natürlich entsprechend mit den Größenangaben zu Rahmenstärke und Rahmenabstand verfahren werden. Hier kann man zuerst mit der Zwischenspeicherung der Längen wie oben beginnen:

```
\newlength{\BorderWidth}
\newlength{\BorderSeparation}

\makeatletter
\define@key{TextBox}{border}{%
\setlength\BorderWidth{#1}}
\define@key{TextBox}{bordersep}{%
\setlength\BorderSeparation{#1}}
\makeatother
```

Die Vorbelegung sollte man auch nicht vergessen:

```
\presetkeys{TextBox}{%
bordercolor=black,textcolor=black,%
background=white,border=0.8pt,%
bordersep=3pt}{}
```

Und natürlich benötigt man noch ein neues Kommando:

```
\newcommand\framedtextG[2][]{%
\begingroup%
\setkeys{TextBox}{#1}%
\setlength{\fboxrule}{%
  \BorderWidth}%
\setlength{\fboxsep}{%
  \BorderSeparation}%
\fcolorbox{\BorderColor}{%
\BackgroundColor}{%
\begin{minipage}{%
\linewidth-2\fboxsep-2\fboxrule}%
\textcolor{\TextColor}{#2}%
\end{minipage}}%
\endgroup}
```

Listing 8: framedtextG.tex

Die Benutzung im Standardfall sieht nun ganz einfach aus:

```
\framedtextG{\lorem}
```

Oder mit einem etwas dickeren und grünen Rahmen wie folgt:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Die Box mit **\framedtextG**.

```
\framedtextG[bordercolor=green,↷
border=3pt]{\lorem}
```

Eigentlich ist diese Lösung aber umständlich, da man die Längen für **\fboxrule** und **\fboxsep** auch gleich ohne Umwege setzen kann. Dafür entfernt man die beiden **\newlength**-Zeilen wieder und ersetzt die Definition der Schlüssel sowie den Befehl **\framedtextG** durch:

```
\makeatletter
\define@key{TextBox}{border}{%
  \setlength\fboxrule{#1}}
\define@key{TextBox}{bordersep}{%
  \setlength\fboxsep{#1}}
\makeatother
\newcommand\framedtextG[2][]{%
\begingroup%
\setkeys{TextBox}{#1}%
\fcolorbox{\BorderColor}{%
\BackgroundColor}{%
\begin{minipage}{%
\linewidth-2\fboxsep-2\fboxrule}%
\textcolor{\TextColor}{#2}%
\end{minipage}}%
\endgroup}
```

Listing 9: framedtextG.tex (2. Version)

So wirkt das Ganze schon etwas übersichtlicher.

Fehlende Schlüsselwerte vorbelegen

Neben der Vorbelegung mit Standardwerten, wenn der Schlüssel fehlt, gibt es auch die Besonderheit, einen Schlüssel mit einem Wert vorzubelegen, wenn der Schlüssel angeben wird, aber kein Wert.

Als sinnvolles Beispiel soll der Rahmen der Box standardmäßig gar nicht angezeigt werden. Nur wenn man mindestens **border** oder **border=WERT** als Option schreibt, soll ein Rahmen angezeigt werden.

Dafür ändert man die Schlüsseldefinition wie folgt:

```
\define@key{TextBox}{border}[0.8pt↷
]{\setlength{\BorderWidth}{#1}}
```

und die Vorbelegung in

```
\presetkeys{TextBox}{%
bordercolor=black,textcolor=black,%
background=white,border=0pt,%
bordersep=3pt}{}
```

Das bedeutet also, dass wenn die Option **border=WERT** bei der Angabe von **\framedtextG** fehlt, kein Rahmen angezeigt wird. Schreibt man dageben nur **border**, wird der in eckigen Klammern angegebene vordefinierte Wert bei **\define@key** benutzt.

Als kleine Übung kann man den Abstand zum Rahmen ebenfalls per Standard auf **0pt** setzen, bei der Angabe von **bordersep** aber den Wert **3pt** nehmen. Fortgeschrittene LaTeX-Nutzer und **xkeyval**-Kenner können versuchen, den Wert bei **bordersep** in Abhängigkeit von **border** zu setzen, denn nur wenn der Rahmen überhaupt sichtbar ist, ist auch ein Rahmenabstand sinnvoll.

Textausrichtung einstellen

Was jetzt noch fehlt, ist die Einstellung der Textausrichtung. Es wäre zwar möglich, diese nach wie vor über simple Buchstaben wie **r**, **l**, **c** oder **b** zu definieren, nur würde man dann für jede dieser Optionen einen eigenen Schlüssel benötigen. Und die Frage ist, was passiert, wenn jemand **r,l** als Option angibt.

Aus diesem Grund ist es besser, einen Schlüssel **align** zu definieren, der nur bestimmte Werte akzeptiert. Hierfür gibt es den sogenannten **\choicekey**:

```
\makeatletter
\define@choicekey{TextAlignment}{%
align}[\val\al]{right,left,center,%
block}
{%
  \ifcase\al\relax%
    \flushright%
  \or%
    \flushleft%
  \or%
    \centering%
  \or%
    % nichts tun
  \fi}
\makeatother
```

```
\presetkeys{TextAlignment}{%
align=block}{}
```

Listing 10: TextAlignment.tex

Alternativ könnte man auch die Verwendung von **align** so ändern, dass der Benutzer immer den echten Befehl angeben muss, also beispielsweise **align=\flushright** für rechtsbündigen Text. Dies erfordert aber, dass der Benutzer die genauen LaTeX-Befehle kennen muss, und würde außerdem dem Beispiel hier im Artikel die Grundlage entziehen.

Die Argumente von **\define@choicekey** sind als Erstes wieder die Familie, danach das Schlüsselwort und als drittes, optionales Argument die Zuweisung des Wertes (in unserem Beispiel kann man über **#1** darauf zugreifen). Danach folgt die Liste der erlaubten zuweisbaren Werte und zum Schluss der LaTeX-Code, der mittels **\ifcase** zu den erlaubten Werten den richtigen Code ausführt. Dabei muss die Reihenfolge der erlaubten Werte mit den auszuführenden Befehlen übereinstimmen.

Man sollte bei dem Beispiel aber aufpassen, weil eine neue Familie **TextAlignment** benutzt wird. Den Grund dafür zeigt die Definition des neuen Befehls:

```
\newcommand\framedtextH[2][]{%
\begingroup%
\setkeys*{TextBox}{#1}%
\fcolorbox{\BorderColor}{%
  \BackgroundColor}{%
\begin{minipage}{%
\linewidth-2\fboxsep-2\fboxrule}%
\setrmkeys{TextAlignment}%
\textcolor{\TextColor}{#2}%
\end{minipage}}%
\endgroup}
```

Listing 11: framedtextH.tex

Die Schlüssel für die Ausrichtung (englisch „alignment“) dürfen erst innerhalb der Minipage-Umgebung gesetzt werden, da die daraus entstehenden Befehle wie **\flushleft** oder **\centering** außerhalb der Umgebung nicht die richtige Wirkung zeigen würden.

Die Familie **TextBox** kennt aber keinen Schlüssel **align**, daher muss dieser ignoriert werden. Man könnte dies entweder durch die Zeile

```
\setkeys{TextBox}[align]{#1}%
```

erreichen, bei dem alle unbekannten Schlüsselworte in den eckigen Klammern ignoriert werden.

Besser ist aber die Lösung über **\setkeys*** oben. Alle in der Familie unbekannten Werte werden in einer extra Liste gespeichert, die man dann später mittels **\setrmkeys** der richtigen Familie zuweisen kann (**rm** steht für „remaining“).

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Die Box mit ***\framedtextH****.*

Von der unsauberen Lösung

```
\setkeys{TextBox,TextAlignment↷
}{#1}%
```

anstelle des **\setrmkeys** sollte man absehen, da man im Allgemeinen nicht sicherstellen kann, dass die Schlüsselzuweisung von **TextBox** keine negativen Nebeneffekte auf den LaTeX-Code hat.

Text mit Umbrüchen

Obige Definition von **\framedtextH** hat den Nachteil, dass es keine Texte mit mehreren Absätzen zulässt. Als Verbesserung kann man folgende Definition nutzen:

```
\newsavebox\MyFrameBox
\newcommand\framedtextI[2][]{%
\begingroup%
\setkeys*{TextBox}{#1}%
\sbox{\MyFrameBox}{%
\begin{minipage}{%
\linewidth-2\fboxsep-2\fboxrule}%
\setrmkeys{TextAlignment}%
\color{\TextColor}%
#2%
\end{minipage}%
}%
\fcolorbox{\BorderColor}{%
\BackgroundColor}{\usebox{%
\MyFrameBox}}%
\endgroup}
```

Listing 12: framedtextI.tex

Dadurch wird der Inhalt im Boxregister **\MyFrameBox** definiert und dieser erst danach in **\fcolorbox** verwendet. Zusätzlich wurde

von **\textcolor** auf **\color** umgestellt, da dies sonst mehrere Absätze im Text verhindern würde.

Die Benutzung ist dann

```
\framedtextI[border]{\lorem\par\↷
lorem}
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
Und noch etwas mehr nichtssagender Füll-text, damit der Artikel bündig am Ende abschließt.

Die Box mit ***\framedtextI****.*

Weitere xkeyval-Schlüssel

Es gibt noch zwei weitere Befehle, die für den einen oder anderen Benutzer wichtig sein könnten. Zum einen:

```
\define@cmdkey{FAMILIE}{SCHLUESSEL↷
}[STANDARDWERT]{\latex{}-CODE}
```

Dies definiert, analog zu oben, einen normalen Schlüssel. Zusätzlich hat man innerhalb des LaTeX-Codes (letztes Argument) über das Kommando **\cmdKV@FAMILIE@SCHLUESSEL** Zugriff auf den Inhalt des definierten Schlüssels.

Der zweite Befehl definiert boolesche Schlüssel:

```
\define@boolkey{FAMILIE}{SCHLUESSEL↷
}[STANDARDWERT]{\latex{}-CODE}
```

Diese ähneln **\choicekey**, denn es sind nur die Schlüsselwerte **true** und **false** erlaubt. Man kann sich zum Beispiel eine neue boolesche Variable definieren

```
\newboolean{Variable}
```

und im LaTeX-Code Folgendes schreiben:

```
\setboolean{Variable}{#1}
```

Danach kann man in seinem Code entsprechend des Wertes unterschiedliche Befehle ausführen:

```
\ifthenelse{\boolean{Variable}}{%
  % Code, falls Variable true ist
}{%
  % Code, falls Variable false ist
}
```

Abschließende Bemerkung

Wie oben angekündigt, geben die im Artikel vorgestellten Beispiele nur einen kleinen Überblick über das, was man mit **xkeyval** machen kann. Dennoch kann man auch als LaTeX-Anfänger sehr schnell gute Resultate erzielen, wenn man das recht einfache Prinzip verstanden hat. Und selbst, wenn man nur Copy & Paste nutzt, kommt man womöglich zum Ziel, auch ohne die Details zu verstehen.

Das LaTeX-Dokument, welches alle Beispiele enthält, kann auch komplett heruntergeladen werden: framedtext.tex.

Links

[1] http://tug.ctan.org/pkg/xkeyval
[2] http://tug.ctan.org/pkg/keyval
[3] http://tug.ctan.org/pkg/pgfkeys
[4] http://tug.ctan.org/pkg/kvoptions
[5] http://tug.ctan.org/pkg/kvsetkeys
[6] http://tug.ctan.org/pkg/calc
[7] http://tug.ctan.org/pkg/xcolor
[8] http://tug.ctan.org/pkg/xifthen

Autoreninformation

Dominik Wagenführ (Webseite) ist Chefredakteur bei **freies**Magazin und kümmert sich dort auch um die Gestaltung der TeX-Infrastruktur. xkeyval hat sich dabei als sehr großer Helfer herausgestellt.

Diesen Artikel kommentieren

Kurzreview: Humble Indie Bundle 3

von Dominik Wagenführ

Das Humble Indie Bundle [1] hat schon eine gewisse Tradition, so wurde die erste Version bereits im Mai 2010 veröffentlicht. Teil des Bundles sind Spiele, die von verschiedenen Independent-Studios entwickelt wurden und auf allen großen Plattformen Linux, Mac OS X und Windows laufen. Ende Juli wurde die dritte Ausgabe veröffentlicht, auf deren Inhalt in dem Artikel ein kleiner Blick geworfen werden soll.

Geschichte

Das Humble Indie Bundle wurde erstmals im Mai 2010 veröffentlicht [2]. Es wurde vor allem deswegen bekannt, weil jeder Käufer den Preis selbst bestimmten konnte. Umso erstaunter waren die Anbieter, dass nach den zwei Wochen, die das Paket erhältlich war, über eine Million US-Dollar eingenommen werden konnte [3]. Den Kaufpreis kann man dabei auf die Spielentwickler, die Anbieter des Humble Indie Bundles, die Electronic Frontier Foundation [4] und die Charity-Organisation Child's Play [5] verteilen. Ein weiterer wichtiger Aspekt des Spielepaketes ist aber auch, dass die Spiele DRM-frei sind und unter den drei großen Plattformen Linux, Mac OS X und Windows spielbar sind.

Aufgrund des Erfolges des ersten Paketes, welches u.a. großartige Spiele wie World of Goo (siehe freiesMagazin 03/2009 [6]), Aquaria [7] oder Gish (siehe freiesMagazin 07/2010) [8] enthielt, gab es im Dezember 2010 ein zweites Spielepaket zu kaufen. Diesmal waren hervorragende Spiele wie Machinarium (siehe freiesMagazin 02/2010 [9]) und Osmos (siehe freiesMagazin 07/2010 [10]) dabei.

Im April 2011 gab es dann eine kleine Sonderedition in Form eines Humble Frozenbyte Bundle [11], welches nur Spiele der Spieleschmiede Frozenbyte enthielt, darunter unter anderem Trine (siehe freiesMagazin 07/2011 [12]).

Das dritte Humble Indie Bundle steht seit dem 27. Juli 2011 zum Kauf bereit. Die Aktion läuft noch bis zum Dienstag, den 9. August 2011, wird aber gegebenenfalls wie die beiden Bundles zuvor etwas verlängert. Da das Spielepaket nach der Aktion nicht mehr erhältlich ist, sollte man zuschlagen, solange man noch kann.

In den nächsten Zeilen wird etwas auf die enthaltenen Spiele eingegangen, welche zu einem großen Teil aus Physik- und/oder Denkaufgaben bestehen.

Hinweis: Alle Spiele wurde unter Ubuntu 10.04 64-bit auf einem Intel Core2 Duo mit 3 GHz, 4 GB RAM und einer NVIDIA-GTX460-Grafikkarte getestet.

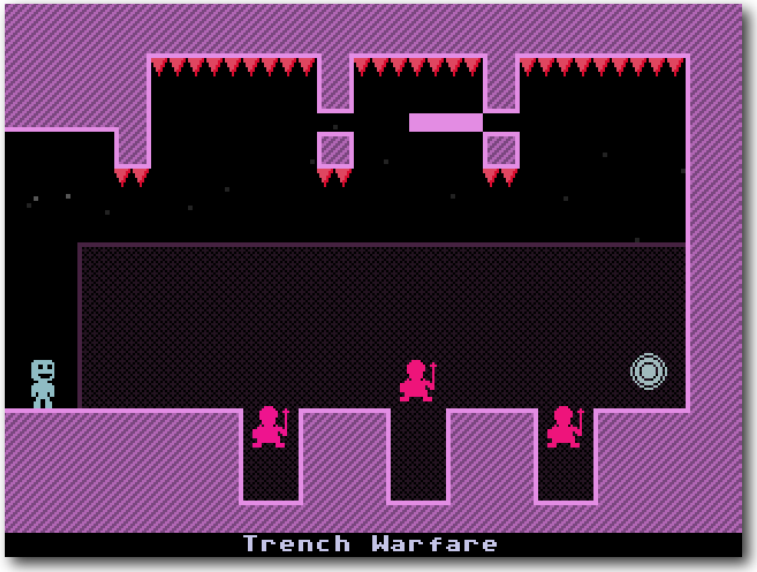
Cogs

Cogs [13] ist ein sehr einfach zu erlernendes Puzzlespiel. Die Aufgabe ist es, durch das Verschieben von Platten, auf denen Zahnräder oder Röhren befestigt sind, eine Maschine anzutreiben, Glocken läuten zu lassen oder ähnliches. Die einzelnen Elemente müssen so angeordnet werden, dass sie ineinander greifen. Damit dies nicht zu langweilig ist, befinden sich die Puzzles manchmal auf mehreren Seiten eines Würfels oder auf der Vorder- und Rückseite einer Tafel.

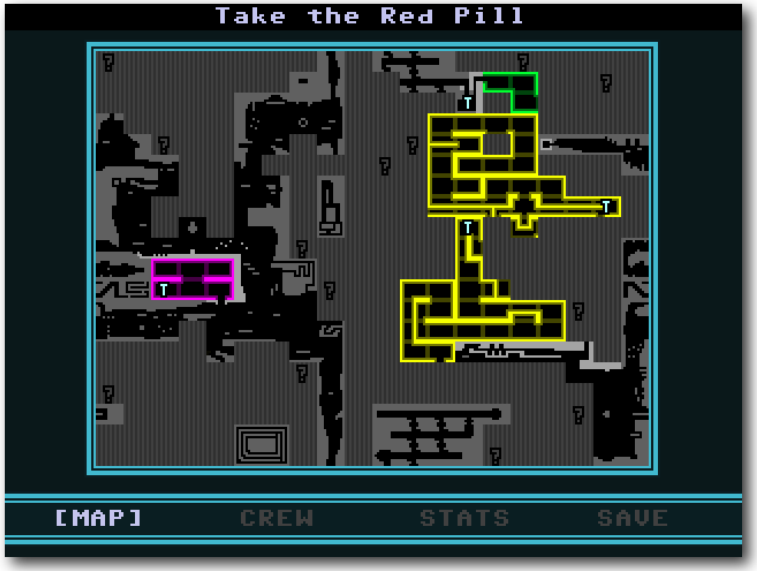
Als kleine Herausforderung erhält man Trophäen, wenn man ein Puzzle innerhalb einer gewissenen Zeit oder mit einer geringen Anzahl von Zügen erledigt.

Die Spielidee ist nicht sonderlich neu, die grafische Präsentation ist aber recht hübsch. Entsprechend wurde Cogs auch auf dem Independent Games Festival 2010 in der Kategorie „Excellence In Design“ nominiert [14].

Wenn die Zahnräder richtig stehen, sieht man dieses lustige Kerlchen aus der Box springen.



Ein einfaches Durchqueren des Raums ist nicht immer möglich.



Die Räume des Raumschiffs in der Übersicht.

VVVVVV

VVVVVV [15] ist ein Jump'n'Run-Spiel, welches von der grafischen Präsentation stark an die Spiele der C64-Ära erinnert. Dies tut dem Spielspaß aber keinen Abbruch, sondern verschafft manchem sicher auch eine Art Nostalgie-Gefühl.

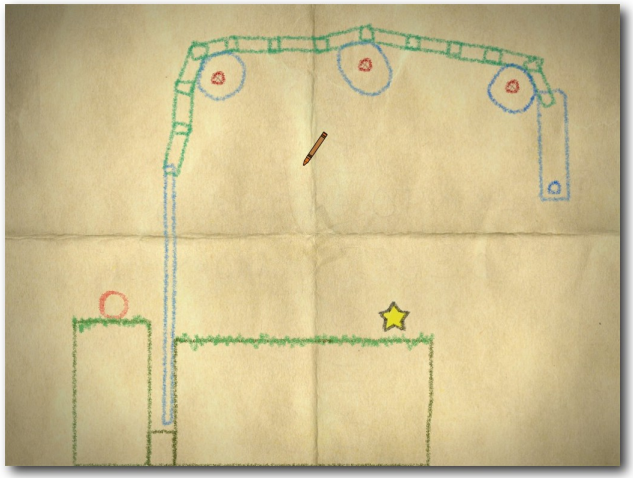
Man selbst übernimmt die Rolle des Raumschiff-Captains Viridian, der auf eine Raumanomalie stößt, welche die ganzen Crew auf dem Raumschiff verteilt. Die Aufgabe ist es, sich durch das Raumschiff zu kämpfen und die Crew zu suchen, wobei weitere Störungen wie Endlosschleifen oder Portale die Suche nicht einfacher machen.

Die Steuerung ist sehr simpel, da man sich nur nach rechts und links bewegen kann. Mit der Aktionstaste kehrt man aber die Gravitation auf dem Raumschiff um und kann dementsprechend an der Decke entlang gehen. Dies ist auch notwendig, um alle Räume des Raumschiffes zu erreichen.

Obwohl das Spiel so simpel ist, kann es für einige Stunden fesseln. Dank der Übersichtskarte sieht man sogar, welche Gebiete des Raumschiffes man noch nicht erkundet hat. Und gerade das animiert einen dazu, doch noch den nächsten Raum aufzusuchen.

Crayon Physics Deluxe

Crayon Physics Deluxe [16] ist ein Physik-Puzzle-Spiel, bei der man sich als Maler austoben kann. Mit einem Wachsmalstift bewaffnet ist es die Aufgabe, in jedem Level einen roten Ball zu einem Stern zu bringen. Dabei zeichnet man mit dem Stift Objekte, die dann mit den vorhandenen Spielelementen physikalisch korrekt interagieren. So kann man beispielsweise eine Schleuder bauen, um den Ball zum Stern zu befördern. Das Prinzip erinnert dabei an das Physik-Spiel Phun (siehe freiesMagazin 04/2008 [17]).



Wie bekommt man den Ball zum Stern?.

Das Spielprinzip ist leicht erklärt, macht aber Spaß. Etwas Experimentierfreude vorausgesetzt kann man sich sehr lange mit Crayon Physics Deluxe beschäftigen. Schade ist, dass auf dem Testrechner der Ton nicht so ging, sodass man sich eine eigene musikalische Untermalung suchen musste.

Im Menü wählt man das Level aus.

Ebenso schade ist es, dass das Spiel nur in Englisch vorliegt. Vor allem für jüngere Spieler wäre es gut gewesen, auch eine deutsche Übersetzung anzubieten, denn diese können viel Spaß beim Malen haben. Für das gute Spielprinzip hat Crayon Physics Deluxe auch zurecht den Seumas McNally Grand Prize auf dem Independent Games Festival 2008 [18] erhalten.

And Yet It Moves

Das Jump'n'Run And Yet It Moves [19] besticht vor allem durch seine Optik, denn alle Levels sind im Scherenschnitt gehalten. Bäume sind also beispielsweise nicht bloß gezeichnet, sondern wirken eher wie aus mehreren Fotos herausgerissen und zusammengeklebt.

Das Spielprinzip setzt VVVVVV von oben fort, auch wenn man hier nicht die Gravitation umdreht, sondern die Welt mit den Pfeiltasten drehen kann, um zum Ausgang eines Levels zu kommen. In jedem Level gibt es sehr fair verteilt Speicherpunkte, die einem zugleich die Richtung weisen, in denen das Level weitergeht. Das ist auch notwendig, denn wenn man sich bzw. die Welt mehrmals gedreht hat, weiß man nicht mehr, wo eigentlich oben und unten ist.

Spannend wird das Spiel nicht nur, weil man den Ausgang suchen muss, sondern auch, weil man aufpassen muss, nicht aus zu großer Höhe zu fallen oder von herabfallenden Steinen erschlagen zu werden. Daneben versperren auch fleischfressende Pflanzen oder schlafende Gorillas den Weg.

Erfreulich ist, dass das Spiel ins Deutsche übersetzt wurde, auch wenn man eigentlich keine große Erklärung benötigt. Der Schwierigkeitsgrad ist angenehm steigend und somit auch für jüngere Spieler geeignet. Gestört haben nur die Tonstörungen, die an einigen Stellen auftauchten.

Auch And Yet It Moves konnte beim Independent Games Festival 2007 als Student Showcase Winner einen Preis erzielen [20].

Hammerfight

Das aus Russland stammende Hammerfight [21] ist ein 2-D-Actionspiel, bei

Die fleischfressende Pflanze lässt einen nicht so einfach vorbei.

Klopperei mit Hammern.

dem man mit einem an einer Flugmaschine versehenen Hammer durch rhytmische Mausbewegungen diesen zum Schwingen bringen muss, um dann damit den Gegner zu treffen.

Das Spielprinzip klingt nicht so schlecht, auch wenn es sicher nicht jedermanns Geschmack ist, der Test schlug nur leider komplett fehl, weil die Maussteuerung so langsam reagierte, dass man das Fluggerät nicht kontrolliert bewegen, sondern nur hektisch mit der Maus fuchteln konnte. Aus dem Grund wurde im Test selbst nach mehreren Versuchen nicht einmal das erste Level überstanden, sodass sehr wenig über die Spielentwicklung gesagt werden kann.

Grafisch präsentiert sich das Spiel ganz nett, auch wenn bei der geringen Auflösung viele Details untergehen.

Fazit

Auch wenn (nach meiner Meinung) mit And Yet It Moves, VVVVVV und Crayon Physics Deluxe drei sehr gute Spiele im Humble Indie Bundle enthalten sind, sind die anderen beiden doch nur nettes Beiwerk. Im Vergleich zu den vorherigen Bundles könnte man fast meinen, dass die Qualität sinkt, was aber sicher nicht stimmt, denn die Geschmäcker sind nur verschieden.

Hinweis: Nachdem der Artikel war, wurde zusätzlich noch das Spiel Steel Storm: Burning Retribution [22] zum Bundle hinzugefügt.

Dennoch ist natürlich der Selbstbestimmungspreis unschlagbar, sodass jeder genau das zahlen kann, was er denkt, dass die Spiele wert sind. Zur Zeit (01.08.2011, 22 Uhr) steht der Verkaufzähler bei 200.000 verkauften Einheiten und Einnahmen in Höhe von über 940.000 US-Dollar. Wie auch die Bundles zuvor geben die Linux-Nutzer durchschnittlich das Dreifache eines Windows-Nutzers für den Kauf aus.

Schade ist nur, dass die Spiele nach der Humble-Aktion wohl nicht mehr verfügbar sein werden-Paket als auf das Spiel selbst, denn obwohl ein Linux-Port zur Verfügung steht, bieten die Entwickler nach der Aktion nicht immer einen Linux-Kauf an, wie die Vergangenheit gezeigt hat. So gibt es beispielsweise Aquaria, Gish oder Trine nicht als Download für Linux.

Es sei noch darauf hingewiesen, dass die Spiele nicht Open Source sind, was sich aber gegebenenfalls in der Zukunft ändern kann, wie das auch beim ersten Humble Indie Bundle der Fall war.

LINKS

[1] http://www.humblebundle.com/
[2] http://www.deesaster.org/blog/index.php?/archives/1420
[3] http://www.deesaster.org/blog/index.php?/archives/1425
[4] https://www.eff.org/
[5] http://www.childsplaycharity.org/
[6] http://www.freiesmagazin.de/freiesMagazin-2009-03
[7] http://www.bit-blot.com/aquaria/
[8] http://www.freiesmagazin.de/freiesMagazin-2010-07
[9] http://www.freiesmagazin.de/freiesMagazin-2010-02
[10] http://www.freiesmagazin.de/freiesMagazin-2010-07
[11] http://www.deesaster.org/blog/index.php?/archives/1647
[12] http://www.freiesmagazin.de/freiesMagazin-2011-07
[13] http://www.cogsgame.com/
[14] http://www.igf.com/2010finalistswinners.html#finalists
[15] http://thelettervsixtim.es/
[16] http://www.crayonphysics.com/
[17] http://www.freiesmagazin.de/freiesMagazin-2008-04
[18] http://www.igf.com/2008finalistswinners.html
[19] http://www.andyetitmoves.net/
[20] http://www.igf.com/2007finalistswinners.html#and
[21] http://www.koshutin.com/
[22] http://www.steel-storm.com/

Autoreninformation

Dominik Wagenführ (Webseite) spielt sehr gerne unter Linux. Vor allem Geschicklichkeits- und Denkspiele machen ihm dabei viel Spaß.

Diesen Artikel kommentieren

Freie Webanalytik mit Piwik

von Sujeevan Vijayakumaran

Eine freie Webanalytik-Software, die in PHP geschrieben ist und MySQL als Datenbank nutzt, ist *Piwik* [1]. Mitte Juni wurde *Piwik* in Version 1.5 veröffentlicht. Die Software ist unter der GPL v3 lizenziert und soll eine freie Alternative zu Google Analytics [2] darstellen.

Allgemeines

Eines der wichtigen Instrumente für einen Webseitenbetreiber ist die Webanalytik. Durch Webanalytik-Software ist es möglich, das Verhalten der Webseitenbesucher zu analysieren. Die Software zeichnet viele Statistiken eines Webseitenbesuchers auf. Neben der Besuchshäufigkeit und Besuchsdauer der eigenen Webseite wird auch die IP-Adresse und Herkunft des Benutzers gespeichert.

Die bekannteste Webanalytik-Software kommt aus dem Hause Google. Google Analytics bietet die umfangreichste Webanalytik auf dem Markt und ist daher auch sehr weit verbreitet. Der große Nachteil von Google Analytics ist, dass alle Daten direkt an die Google-Server übermittelt und auch dort verarbeitet werden. Eine freie Alternativsoftware wie *Piwik* ermöglicht ebenfalls eine gute Webanalytik.

Einen Einblick in die Analysemöglichkeiten von *Piwik* soll dieser Artikel geben. *Piwik* erzeugt aus den gespeicherten Daten sehr viele Berichte, die in verschiedene Kategorien eingeordnet werden. Die Daten werden für jeden einzelnen Besucher erhoben, die je nach Statistik zusammen oder getrennt aufgelistet werden.

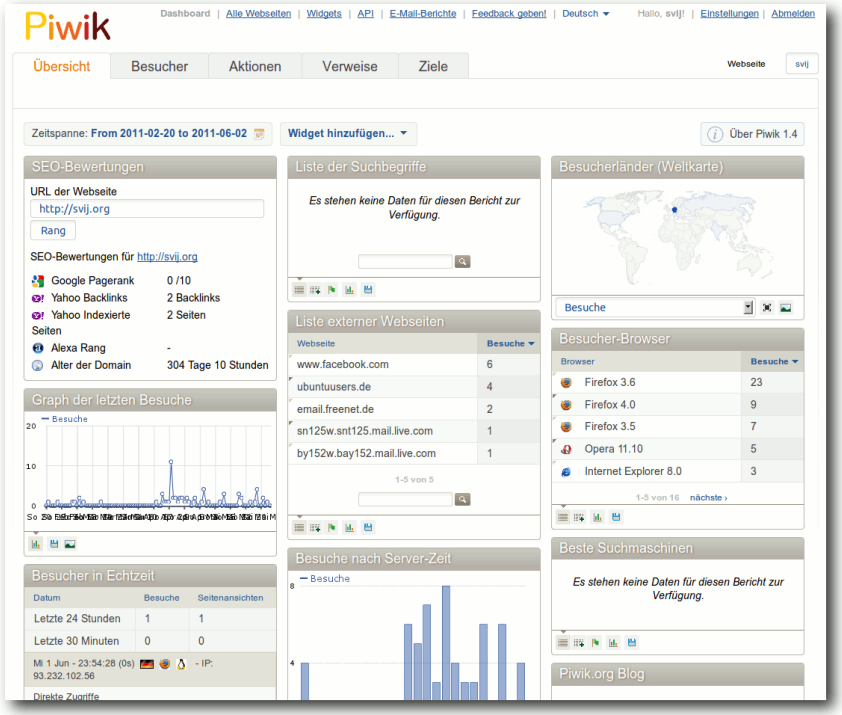
Piwik nutzt, wie andere Webanalytik-Software auch, einen JavaScript-Code zum Aufzeichnen des Verhaltens der Webseitenbesucher. Der JavaScript-Code muss auf jeder einzelnen Seite eines Webauftritts eingebunden sein, damit alle Daten ordnungsgemäß und vollständig erhoben werden können. In den meisten Content-Management-Systemen lässt sich dieser Code sehr einfach in das Template des Footer einbinden. Es wird empfohlen, den Standard JavaScript-Code zu verwenden, es ist jedoch auch möglich den Code anzupassen. So lassen sich einige Funktionen ein- bzw. ausschalten. Eine Alternative für die Benutzer, die JavaScript im Browser deaktiviert haben, gibt es ebenfalls. *Piwik* realisiert dieses mit einen Image Tracker Code, der logischerweise kein JavaScript verwendet. Nachteilig ist hierbei jedoch, dass einige wichtige Funktionen nicht aufgezeichnet werden können. Darunter fallen unter anderem Suchschlüsselwörter, Browser-Plug-ins und Bildschirmauflösungen.

Installation und Updates

Die Installation von *Piwik* lässt sich zügig in etwa fünf Minuten erledigen und erfolgt ziemlich ähnlich der Installation von diversen Content-Management-Systemen. Verfügbare Aktualisierungen erscheinen im Administrationsmenü von *Piwik*. Diese lassen sich mit wenigen Klicks direkt im Browser installieren.

Übersicht

Piwik unterteilt die Bedienoberfläche in die Seiten „*Besucher*“, „*Aktionen*“, „*Verweise*“ und



Die Start- und Übersichtsseite von Piwik.

„*Ziele*“. Auf jeder Unterseite lassen sich die Daten der überwachten Webseite für einen speziell ausgewählten Zeitraum erfassen. So kann man die Statistiken zu jeden einzelnen Tag, Monat, und Jahr getrennt aufrufen. Neben diesen vordefinierten Zeiträumen, kann man ebenfalls seinen eigenen Zeitraum definieren.

Die Übersichtsseite, welche in drei Spalten gegliedert ist, in denen die wichtigsten Widges liegen, bildet gleichzeitig auch die Startseite von *Piwik*. Standardmäßig werden dort unter anderem die Statistiken zu den letzten Besuchern, Referrer-Webseiten und die Besucher-Browser angezeigt. Der Administrator kann die Widgets komplett nach seinen eigenen Belieben anpassen. So ist es möglich, vorhandene Widgets zu löschen oder neue hinzufügen, außerdem lassen sie sich per Drag & Drop frei verschieben und neu anordnen.

Besucher

Piwik zeichnet sehr viele Informationen zu jedem einzelnen Besucher auf. Auf der Besucher-Übersichtsseite lässt sich die Benutzerentwicklung nachvollziehen, des Weiteren werden Zahlen zur durchschnittlichen Aufenthaltsdauer, Absprungrate der Benutzer oder auch die Besucheranzahl angezeigt.

Das Besucher-Log zeigt genaue Informationen für jeden Besucher an. Als Beispiel (siehe Bild) dient hier ein Besucher einer Webseite, der die Fotogalerie aufgerufen hat. Neben der IP-Adresse, die hier anonymisiert gespeichert ist,

Die gespeicherten Informationen zu einem eindeutigen Besucher.

wird auch die sekundengenaue Uhrzeit angezeigt, zu der der Besucher die Seite besucht hat. Sofern sich aus der IP-Adresse der Provider herauslesen lässt, wird auch der Provider angezeigt, welches in diesem Beispiel Citykom sein soll. Weiterhin geht aus der Tabelle hervor, dass der Besucher aus Deutschland kommt, Windows als Betriebssystem verwendet und den Internet Explorer 9.0 als Browser nutzt. *Piwik* erkennt an Hand von einem abgespeicherten Cookie, ob der Besucher die Webseite bereits besucht hat, oder ob es ein neuer Besucher ist, und markiert dies in den Statistiken in Form eines Symbols. Ebenso werden die verwendeten Browser-Plug-ins, wie Flash, PDF, Java oder Quicktime, erfasst und ausgewertet. Einzig die Plug-ins von Besuchern, die den Internet Explorer verwenden, werden, wie hier im Beispiel, nicht gespeichert. Nur die Verwendung von Java wird ausgegeben. Weitere Informationen etwa über die Version des Betriebssystems, der verwendeten Version des Browsers und der genutzten Auflösung wird in einem Mouse-Over über den dazugehörenden Symbolen angezeigt.

Darüber hinaus merkt sich *Piwik* auch die Herkunft des Besuchers, sofern die Referrer-Seite ermittelt werden konnte, ansonsten zeigt *Piwik* an, dass der Besucher direkt auf die Seite zugegriffen hat. Ergänzend ist es mit *Piwik* auch möglich zu sehen, was der Besucher genau auf der Webseite getan, welche Unterseiten er besucht, und

wie lange er sich insgesamt auf der gesamten Webseite aufgehalten hat. Zudem werden Downloads, beispielsweise von Bildern, aufgezeichnet.

Die weiteren Unterkategorien von „*Besucher*“ fassen einzelne Informationen thematisch zusammen. Unter „*Standorte und Provider*“ werden Graphen und Zahlen von Besuchern sortiert in Kontinente, Länder und Provider dargestellt. Im weiteren Punkt „*Einstellungen*“ werden die verwendeten Browser und Plug-ins aufgelistet, als auch die Betriebssysteme und die Bildschirmauflösungen. Das Datenmaterial wird teilweise in Kreisdiagrammen angezeigt, aber auch in Listen mit absoluten Werten und stellenweise Prozentwerten. Der Anwender kann nach eigenen Belieben zwischen den verschiedenen Ausgabeoptionen wie Tabellen und Diagrammen wechseln.

Neben diesen Informationen werden, wie bereits erwähnt, auch die Verweilzeiten gespeichert, die ebenfalls nochmal getrennt in einem Diagramm visualisiert sind. In der Unterkategorie „*Engagement*“ wird unter anderem die durchschnittliche Aufenthaltsdauer auf der Webseite dargestellt sowie die Menge der wiederkehrenden Besucher.

Nützlich sind besonders die zahlreichen Exportoptionen von *Piwik*. Sie ermöglicht das Exportieren eines Datensatzes als CSV, TSV, XML als auch in Json, PHP und RSS.

Aktionen

In die Kategorie „*Aktionen*“ fallen alle Daten über die einzelnen Seiten. Unterteilt wird die Kategorie in „*Seiten*“, „*Eingangsseiten*“, „*Ausstiegsseiten*“, „*Seitentitel*“, „*Ausgehende Verweise*“ und „*Downloads*“. In der Unterkategorie „*Seiten*“ wird unter anderem dargestellt, welcher Teil der Webseite die meisten Seitenansichten hat, wie hoch die Absprung- und Ausstiegsrate ist, aber auch die durchschnittliche Verweildauer auf der Seite. In „*Eingangs- und Ausstiegsseiten*“ wird nun differenziert dargestellt, welche Einzelseite die Besucher zuerst öffnen, und wiederum bei welcher Einzelseite sie die Webseite verlassen haben. Während bei „*Seiten*“ die Daten nach der Struktur und den Dateinamen einzelnen Seiten geordnet sind, werden bei „*Seitentitel*“ die Titel der Einzelseiten aufgezählt und mit den gleichen Fakten veranschaulicht.

Mit den aufgezeichneten Daten unter „*Ausgehende Verweise*“ kann der Webseitenbetreiber leicht erkennen, über welche Links die Besucher am häufigsten die Seite verlassen. Der letzte Unterpunkt unter „*Aktionen*“ ist der Bereich „*Downloads*“. Dort wird ziemlich simpel aufgelistet, welches Datenmaterial und wie oft heruntergeladen wurde.

Verweise

Die Kategorie „*Verweise*“ wertet hauptsächlich die Herkunft der Besucher aus. Unterteilt in „*Suchmaschinen und Suchbegriffe*“ sammelt *Piwik* die Anzahl der Besuche unterteilt nach den verschiedenen Suchmaschinen. Zusätzlich wird erfasst, welche Suchbegriffe die Besucher verwendet haben, um auf die Webseite zu gelangen.

Neben der Herkunft der Besucher von Suchmaschinen kommen diese ebenfalls von anderen Webseiten. Unter „*Webseiten*“ wird hier aufgezählt, welche Seiten in besonderem Maße die Besucher auf die eigene Seite weiterleiten.

Ziele

Eigene Statistiken lassen sich unter „*Ziele*“ erstellen. Der Webseitenbetreiber kann so beispielsweise ein Ziel für Registrierung von neuen Benutzern erstellen oder die Aufrufanzahl von „Bitte Lesen“-Posts verfolgen.

Weiteres

Piwik bietet neben dem Webauftritt auch die Möglichkeit der E-Mail-Benachrichtigung an. Die Berichte lassen sich individuell gestalten und an mehrere Leute senden. Die einzelnen Datensätze lassen sich aktivieren und deaktivieren. So ist es möglich, nur Teile der Daten an weitere Personen zu senden, während der Administrator selbst eine E-Mail mit vollständigen Datensätzen bekommt.

Eigene Funktionen lassen sich mit Hilfe der API realisieren. Die bereits enthaltenen Funktionen sind als Plug-ins in *Piwik* eingebunden. In den Plug-in-Einstellungen ist es möglich, einige der Funktionen ab- und anzuschalten, sodass nicht alles gespeichert wird.

Datenschutz

Ein Punkt, den man bei der Webanalytik nicht außer Acht lassen darf, ist der Datenschutz. Problematisch ist hierbei die Erfassung personenspezifischer Daten wie beispielsweise die IP-Adresse, mit der eine eindeutige Identifizierung (mehr oder

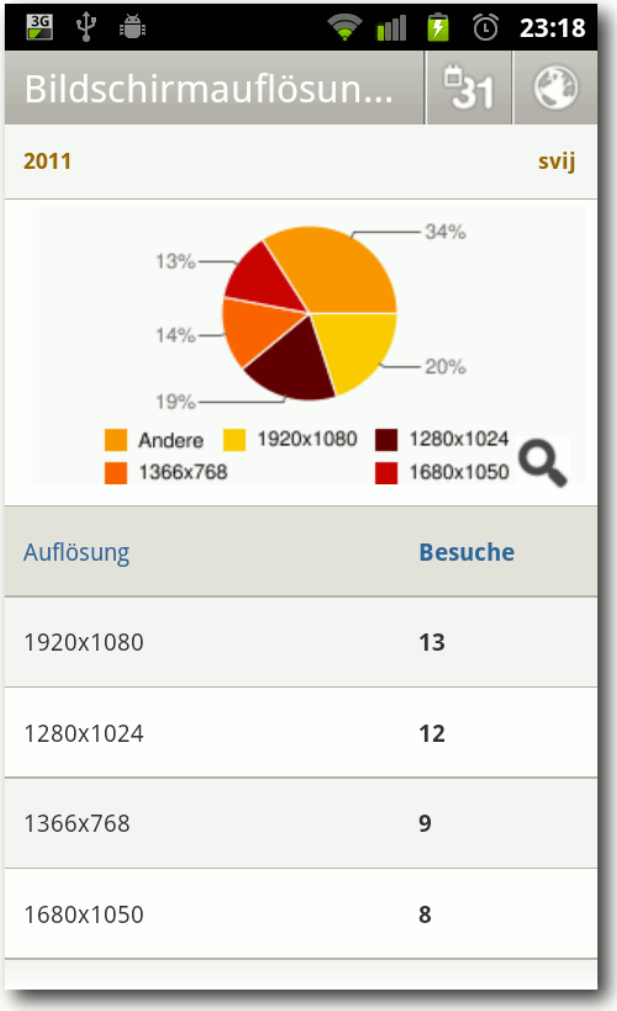
weniger) möglich ist. (Es ist von deutschen Gerichten aber immer noch nicht einstimmig geklärt, ob die IP-Adresse wirklich ein personenbezogenes Datum ist oder nicht [3].)

Piwik arbeitet aus diesem Grund mit dem Unabhängigen Landeszentrum für Datenschutz Schleswig-Holstein (ULD) zusammen [4]. Die angeregten Verbesserungen wurden bereits in der Version 1.2 eingepflegt. Das ULD hat ein Dokument veröffentlicht [5], in welchem angegeben wird, wie man *Piwik* datenschutzrechtlich einwandfrei nutzen kann. Darunter fallen unter anderem die Anonymisation von IP-Adressen als auch die Opt-Out-Funktion, die es ermöglicht, die Besucher entscheiden zu lassen, ob *Piwik* seine Daten speichern darf oder nicht. Standardmäßig werden die Daten jedoch automatisch gespeichert. Der Administrator hat die Möglichkeit, mittels eines iframes dem Besucher die Möglichkeit zu geben, der Speicherung seiner Daten zu widersprechen. Durch die Deaktivierung seitens des Besuchers wird dann der eindeutige Webanalyse-Cookie deaktiviert.

Piwik Mobile

Eine Mobil-Seite für den Abruf der Statistiken von einem Smartphone oder Tablet gibt es bisher nicht. Stattdessen gibt es eine offizielle und kostenlose App unter dem Namen „Piwik Mobile“ für Android [6] und iOS [7], welches ebenfalls unter der GPL lizenziert ist.

Die App bietet ziemlich den gleichen Funktionsumfang wie auch die eigentliche Webseite an sich. Die einzelnen Statistiken sind in Gruppen zusammengefasst und in einer Liste geordnet. Jeder Unterpunkt der gesammelten Informationen wird in einem Kreisdiagramm dargestellt, des Weiteren werden die absoluten Zahlen in einer Tabelle angezeigt. In der oberen Leiste gibt es neben der Möglichkeit, den Zeitraum individuell umzustellen und somit eine veränderte Statistik abzurufen, auch noch die Möglichkeit, zwischen mehreren *Piwik*-Datenbanken zu wechseln.



Statistik zu Bildschirmauflösungen im Android-App.

Geschichte von Piwik

Piwik entstand auf der Grundlage von phpMyVisites [8]. Nach der Einstellung des Projekts phpMyVisites wurde daraus *Piwik*. Die Version 0.1 wurde im März 2009 veröffentlicht und seitdem stetig weiterentwickelt. August 2010 erschien *Piwik* in Version 1.0, danach folgten zügig weitere Veröffentlichungen. Die aktuelle Version 1.5 wurde Mitte Juni veröffentlicht. *Piwik* wurde mittlerweile über 675.000 mal heruntergeladen [9] und wird von zahlreichen Webseitenbetreibern genutzt.

Neues in Piwik 1.5

Mit der neuen Version von *Piwik* kamen, neben Fehlerkorrekturen und weiteren kleineren Änderungen, auch neue Funktionen. Als neue Funktion wurde unter anderem E-commerce [10] und benutzerdefinierte Variablen [11] eingeführt. Des Weiteren wurde die Nutzung von Flash für die Anzeige von Graphen abgeschafft [12] Die Graphen werden nun mittels einer Kombination aus Canvas und JavaScript erzeugt. Einzig für die Weltkarte der Besucher wird noch Flash benötigt.

Ausblick

Nach der Roadmap [13] ist es die Mission von *Piwik* eine führende Open-Source-Webanalytik-

Software zu entwickeln, die den Zugang zur gesamten Funktionalität durch offene APIs und offenen Komponenten gewährleisten soll. Unter anderem ist geplant, dass man benutzerdefinierte Benachrichtigungen auf Aktionen erstellt, sodass der Administrator jedes Mal via E-Mail benachrichtigt wird, wenn dieser Fall eintritt. Des Weiteren soll die Länder- und Städte-Lokalisation mittels der GeoIP-Datenbank eingebunden werden. Alle weiteren Funktionen, die bis zur Version 2.0 eingefügt werden sollen, können in der Roadmap nachgelesen werden.

LINKS

[1] http://de.piwik.org/

[2] http://www.google.de/analytics/

[3] http://www.internet-law.de/2011/06/datenschutz-ip-adressen-als-personenbezogene-daten.html

[4] http://de.piwik.org/blog/2011/03/unabhangiges-landeszentrum-datenschutz-uld-piwik-datenschutzkonform-einsetzbar/

[5] https://www.datenschutzzentrum.de/tracking/piwik/

[6] https://market.android.com/details?id=org.piwik.mobile

[7] http://itunes.apple.com/us/app/piwikmobile/id385536442?mt=8

[8] http://www.phpmyvisites.us/

[9] http://piwik.org/download/counter/

[10] http://piwik.org/docs/ecommerce-analytics/

[11] http://piwik.org/docs/custom-variables/

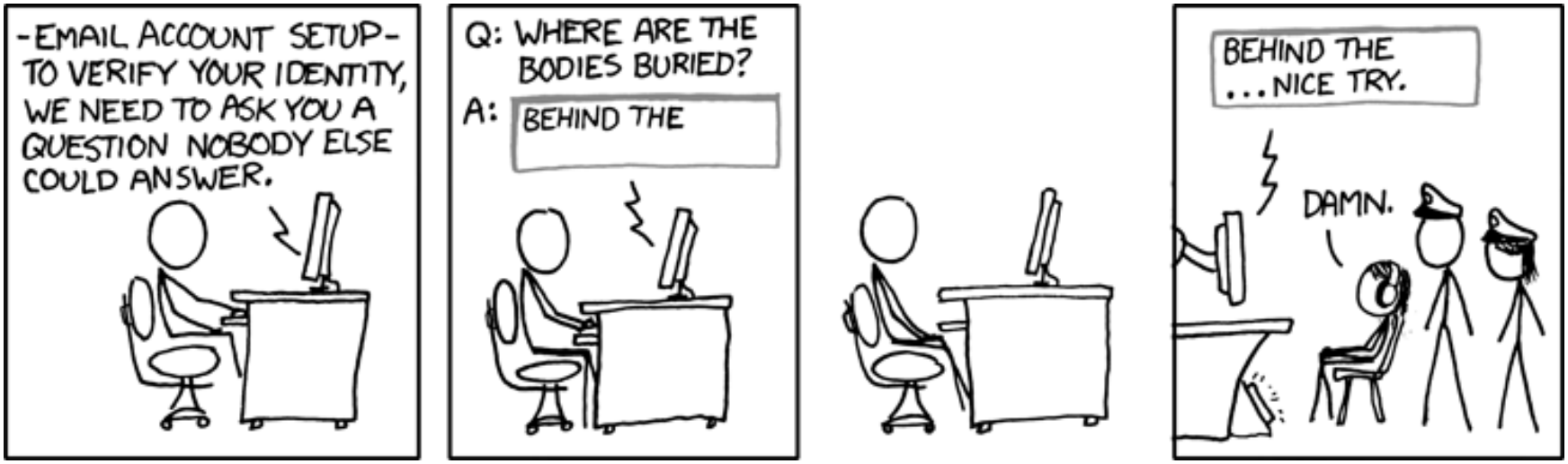
[12] http://piwik.org/blog/2011/06/piwik-innovative-with-javascript-canvas-chart-and-contributing-by-jqplot-creator/

[13] http://piwik.org/roadmap/

Autoreninformation

Sujeevan Vijayakumaran (Webseite) testet gerne verschiedene Content-Management-System und setzt *Piwik* auch auf seiner eigenen Homepage ein.

Diesen Artikel kommentieren



„Security Question“ © by Randall Munroe (CC-BY-NC-2.5), http://xkcd.com/565

Rezension: LPI 301

von Hans-Joachim Baader

Überraschenderweise ist auch vier Jahre nach dem Start der LPIC-3-Zertifizierung das Buchangebot dazu sehr übersichtlich. Mit „LPI 301“ stößt der IT-Trainer Thorsten Robers in diese Lücke vor.

Redaktioneller Hinweis: *Der Artikel „LPI 301“ erschien erstmals bei Pro-Linux [1].*

Vorwort

Das Linux Professional Institute (LPI) [2] bietet umfassende Linux-Zertifizierungsprogramme in bisher drei Stufen an. Die Besonderheit der Zertifizierungen des LPI ist, dass sie in Gemeinschaftsarbeit interessierter Freiwilliger entwickelt werden und hersteller- sowie distributionsunabhängig sind.

Nach den Stufen LPIC-1 („Junior Level Linux Professional“) und LPIC-2 („Advanced Level Linux Professional“), die jeweils zwei Prüfungen umfassen, bietet LPI seit 2007 auch als dritte Stufe LPIC-3 („Senior Level Linux Professional“) als bisher anspruchsvollste Zertifizierung an. LPIC-3 [3] wird anders als die ersten beiden Stufen mit nur einer einzigen Prüfung erworben, die wahlweise durch zusätzliche Prüfungen ergänzt werden kann. Die verpflichtende Prüfung 301 nennt sich „Core“. Inzwischen stehen als Spezialisierungs-Prüfungen „Mixed Environments“ (302), „Security“ (303), „Virtualization and High Availability“ (304, momentan Betaversion) zur Verfügung.

Da die Stufen aufeinander aufbauen, kann man die Zertifizierung als LPIC-3 erst erhalten, wenn man diejenigen für LPIC-2 und LPIC-1 absolviert hat. Die Prüfungen umfassen Multiple-Choice- und frei auszufüllende Fragen. LPIC-3 macht hier keine Ausnahme. Ablegen kann man die Prüfungen in einem Prüfungszentrum, das in der Regel von einem Unternehmen betrieben wird. Weltweit stehen einige tausend Zentren zur Verfügung.

Trotz der über vier Jahre, die die Prüfung 301 bereits existiert, sind Prüfungsunterlagen kaum zu finden. „LPI 301“ ist das erste deutschsprachige Buch zu der Prüfung und auch weltweit eines der ersten, wenn nicht gar das einzige.

Die Prüfung 301 umfasst laut Beschreibung die Themen Authentifikation, Problemlösung, Netzwerk-Integration und Kapazitätsplanung. In der Praxis bedeutet dies fast ausschließlich LDAP.

Das Buch

Das vorliegende Buch widmet sich ausschließlich der Vorbereitung auf die Prüfung 301. Wie bereits erwähnt geht es in dieser Prüfung vorwiegend um LDAP. Es sei aber betont, dass „LPI 301“ deswegen trotzdem kein LDAP-Lehrbuch ist. Wenn es so wäre, hätte es wohl den Begriff „LDAP“ im Titel. Zur Vorbereitung auf die Prüfung 301 ist nicht unbedingt ein Buch über LDAP notwendig, umfassende Kenntnisse von OpenLDAP sind aber unerlässlich. Zur Not genügt hierfür aber auch die Online-Dokumentation von OpenLDAP [4] einschließlich der FAQs sowie praktisches Arbeiten an einem selbst aufgesetzten Server.

Das Buch des erfahrenen Systemadministrators und IT-Trainers Thorsten Robers orientiert sich in seiner Kapitelstruktur an den detaillierten Lernzielen [5] der Prüfung 301. Nach einem kurzen Vorwort von nur drei Seiten beginnt sofort die Vorstellung und Behandlung der Prüfungsinhalte in sechs Kapiteln. Dabei werden keinerlei konkrete Prüfungsfragen vorgestellt. Es ist also leider nicht möglich, eine Testprüfung mit einigermaßen realistischen Fragen zu absolvieren. Entsprechende Fragen, die zwar nicht identisch, aber ähnlich zu den tatsächlich in der Prüfung vorkommenden Fragen sind, findet man eventuell bei anderen Anbietern oder im Web, allerdings ist mir derzeit keine konkrete Quelle bekannt. Stattdessen profitieren die Leser von der Erfahrung des Autors, der die Prüfung bereits kennt und daher Angaben machen kann, worauf man sich besonders vorbereiten muss.

Die Kapitel sind nicht ab 1 fortlaufend nummeriert, sondern von 301 bis 306, was den so nummerierten Themengebieten der Prüfung entspricht. Alle sechs Kapitel sind in Unterkapitel gegliedert, die alle den gleichen Aufbau besitzen. Am Anfang steht jeweils eine kurze Übersicht mit der Gewichtung (1-5), Beschreibung und einem Satz, wichtigen Wissensgebieten und wichtigen

Dateien, Begriffen und Hilfsprogrammen. Diese Information ist mehr oder weniger direkt den Prüfungszielen entnommen. Danach bietet jedes Kapitel eine umfassende Erläuterung zu den Themen und Aufgaben. Es gibt Beispiele zur Konfiguration und Verwendung der Programme, Hinweise auf besonders beachtenswerte Sachverhalte und Hinweise, was für die Prüfung relevant ist. An manchen Stellen finden sich darüber hinausgehende Hinweise. Jedes Kapitel wird mit einem kurzen Abschnitt abgeschlossen, der nochmals darauf hinweist, was zur Vorbereitung auf die Prüfung zu beherrschen ist.

Kapitel 301 „Konzepte, Architektur und Design“ beschäftigt sich mit LDAP-Konzepten und -Architektur, dem Entwerfen eines LDAP-Verzeichnisbaums und Schemata. Kapitel 302 „Installation und Entwicklung“ besitzt nur zwei Unterkapitel, das Kompilieren und Installieren von OpenLDAP und die Softwareentwicklung für LDAP unter Perl oder C++. Letzteres kann man notfalls auch vernachlässigen.

Kapitel 303 „Konfiguration“ weist Unterkapitel zu ACLs in LDAP, LDAP-Replikation, Absicherung des LDAP-Verzeichnisses, Performance-Tuning des LDAP-Servers und Konfiguration des OpenLDAP-Daemons auf. Das nächste Kapitel 304 „Bedienung“ beschreibt das Durchsuchen des Verzeichnisses, LDAP-Kommandozeilenwerkzeuge und Whitepages. Während man in der Praxis meist mit grafischen Oberflächen arbeitet, wenn es nicht gerade um Massenoperationen geht, ist dieser Teil für die Prüfung ziemlich wichtig, da sie sich auf die Kommandozeilenwerkzeuge beschränkt.

Kapitel 305 „Integration und Migration“ ist das letzte Kapitel zum Thema LDAP. Themen sind die LDAP-Integration mit PAM und NSS, Migration von NIS nach LDAP, Integration von LDAP und Unix-Dienste, Integration von LDAP mit Samba, Integration von LDAP mit Active Directory und Integration von LDAP mit E-Mail-Diensten. Es folgt ein abschließendes Kapitel 306 „Kapazitätsplanung“, das einzige Thema, das nichts mit LDAP zu tun hat. Die Themen Messen des Ressourcenverbrauchs, Fehlerbehebung bei Ressourcenproblemen, Analyse der Leistungsanforderungen und Vorhersagen zukünftigen Ressourcenverbrauchs dürften keinem erfahrenen Linux-Administrator Probleme bereiten.

Ein Anhang und ein nützliches Inhaltsverzeichnis runden das Buch ab. Einige Schreib- und Satzfehler erwecken den Eindruck, dass das Buch schnell fertiggestellt werden musste. Fachliche Fehler sind allerdings nicht zu entdecken.

Fazit

„LPI 301“ eignet sich hervorragend zum Sebststudium und ist allen zu empfehlen, die sich auf die Prüfung LPI 301 vorbereiten wollen. Ob man es unbedingt benötigt, hängt sicher von der individuellen Arbeitsweise ab. Die Investition ist aber wesentlich günstiger als eine Wiederholung der Prüfung, falls man es beim ersten Mal nicht schaffen sollte, da LPI 301 mit 250 US-Dollar zu Buche schlägt. Andere professionelle Schulungsunterlagen, soweit überhaupt auf Deutsch verfügbar, sind deutlich teurer und eine mehrtägige Schulung ist noch einmal eine ganz andere Kategorie.

Buchinformationen	
Titel	LPI 301
Autor	Thorsten Robers
Verlag	Open Source Press
Umfang	343 Seiten
ISBN	978-3-941841-36-9
Preis	34,90 EUR

Links

[1] http://www.pro-linux.de/artikel/2/1513/lpi-301.html

[2] http://www.lpi.org/

[3] http://www.lpi.org/eng/certification/the_lpic_program/lpic_3

[4] http://www.openldap.org/

[5] http://www.lpice.eu/de/lpi-zertifizierungsinhalte.html

Autoreninformation

Hans-Joachim Baader (Webseite) befasst sich seit 1993 mit Linux. 1994 schloss er sein Informatikstudium erfolgreich ab, machte die Softwareentwicklung zum Beruf und ist einer der Betreiber von Pro-Linux.de.

Diesen Artikel kommentieren

Veranstaltungskalender

Messen				
Veranstaltung	**Ort**	**Datum**	**Eintritt**	**Link**
Desktop Summit 2011	Berlin	06.08.-12.08.2011	-	https://www.desktopsummit.org
FrOSCon	Sankt Augustin	20.08.-21.08.2011	5 EUR	http://www.froscon.de
LibreOffice Hackfest	München	02.09.2011	-	http://wiki.documentfoundation.org/Hackfest2011
openSUSE Conference	Nürnberg	11.09.-14.09.2011	frei	http://en.opensuse.org/Portal:Conference
Software Freedom Day	Weltweit	17.09.2011	frei	http://softwarefreedomday.org
Ubucon	Leipzig	14.10.-16.10.2011	-	http://www.ubucon.de

(Alle Angaben ohne Gewähr!)

Sie kennen eine Linux-Messe, welche noch nicht auf der Liste zu finden ist? Dann schreiben Sie eine E-Mail mit den Informationen zu Datum und Ort an redaktion@freiesMagazin.de.

Vorschau

freiesMagazin erscheint immer am ersten Sonntag eines Monats. Die September-Ausgabe wird voraussichtlich am 4. September unter anderem mit folgenden Themen veröffentlicht:

- Trinity – Desktop ohne Zukunft

Es kann leider vorkommen, dass wir aus internen Gründen angekündigte Artikel verschieben müssen. Wir bitten dafür um Verständnis.

Konventionen

An einigen Stellen benutzen wir Sonderzeichen mit einer bestimmten Bedeutung. Diese sind hier zusammengefasst:

$:	Shell-Prompt
#:	Prompt einer Root-Shell – Ubuntu-Nutzer können hier auch einfach in einer normalen Shell ein **sudo** vor die Befehle setzen.
↷:	Kennzeichnet einen aus satztechnischen Gründen eingefügten Zeilenumbruch, der nicht eingegeben werden soll.
~:	Abkürzung für das eigene Benutzerverzeichnis **/home/BENUTZERNAME**
:	Kennzeichnet einen Link, der auf eine englischsprachige Seite führt.
:	Öffnet eine höher aufgelöste Version der Abbildung in einem Browserfenster.

Impressum

freiesMagazin erscheint als PDF und HTML einmal monatlich.

ISSN 1867-7991
Erscheinungsdatum: 7. August 2011

Kontakt

E-Mail redaktion@freiesMagazin.de
Postanschrift freiesMagazin
c/o Dominik Wagenführ
Beethovenstr. 9/1
71277 Rutesheim
Webpräsenz http://www.freiesmagazin.de/

Autoren dieser Ausgabe

Redaktion

Frank Brungräber Thorsten Schmidt
Dominik Wagenführ (Verantwortlicher Redakteur)

Satz und Layout

Ralf Damaschke Nico Maikowski
Matthias Sitte

Korrektur

Daniel Braun Stefan Fangmeier
Mathias Menzer Karsten Schuldt
Stephan Walter

Veranstaltungen

Ronny Fischer

Logo-Design

Arne Weinberg (GNU FDL)