Das einzige deutschsprachige Linux-Magazin

LINUX MAGAZIN 9/98

LINUX MAGAZIN

ISSN 1432 - 640 X DM 9,80 ÖS 75,- SFR 9,80

9/98

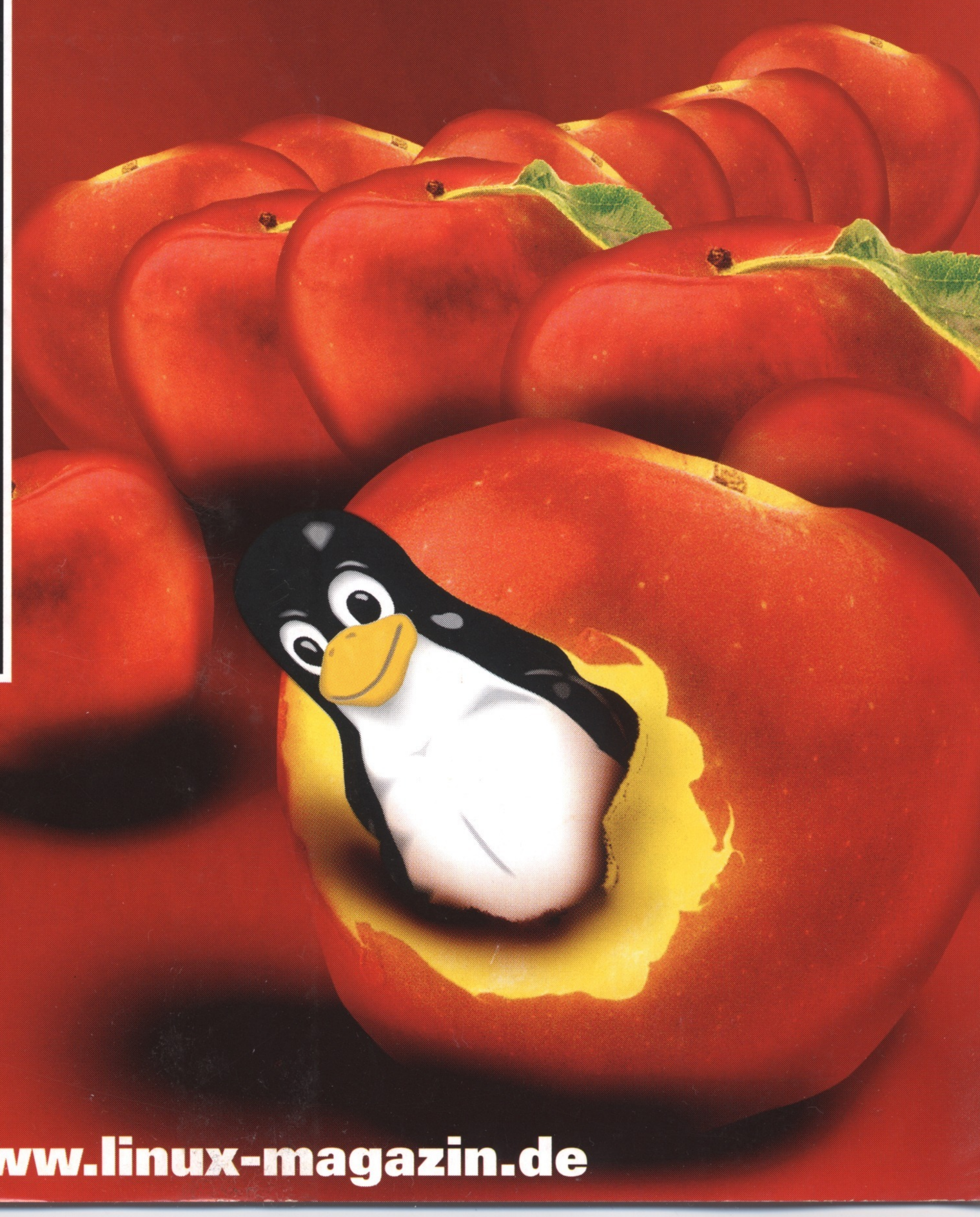
Knackig
Linux auf dem PowerPC

Linux News
Neues aus dem Internet

Informationsdienste
X.500 und LDAP

Netzwerke
NIS für Linux

Testbericht
Caldera OpenLinux 1.2



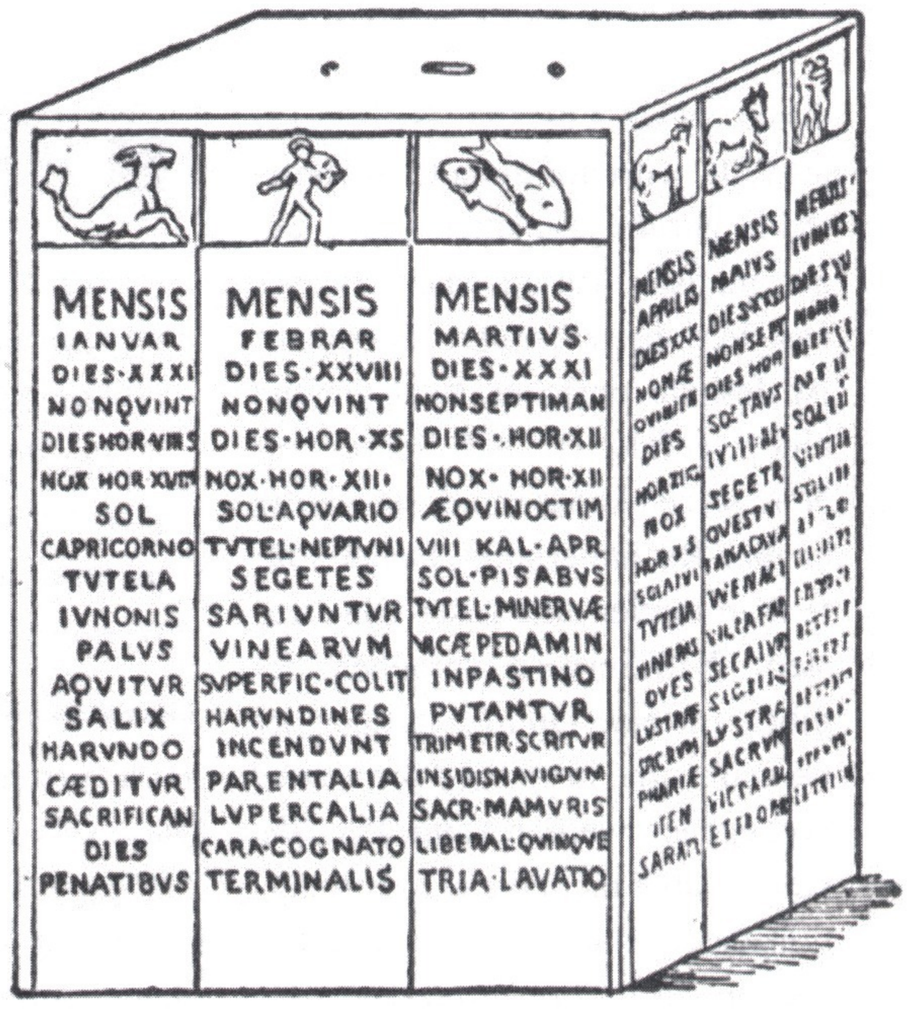
4 392587 309802

www.linux-magazin.de

Perl Snapshot

Kalender-rechnung

von Michael Schilli



Datumsberechnungen halten für den ungeübten Programmierer einige Stolpersteine bereit, doch mit dem neuen *Date::Calc*-Modul geht alles ganz einfach.

Ging es um Datumsberechnungen, war ich bislang ein unbekehrbarer *Date::Manip*-Jünger, so schön einfach zu bedienen schien mir *Sullivan Becks* Modul. Doch als ich neulich *Steffen Beyers* Erzeugnis *Date::Calc* in die Hände bekam, und feststellte, daß es ähnliche Funktionalität bietet und dabei viel, viel schneller läuft, ließ ich ab vom Gewohnten und gab mich voll diesem neuen Teufelszeug hin!

Date::Calc wird nach dem üblichen Verfahren installiert. Die aktuelle Version liegt auf dem CPAN unter *CPAN/modules/by-module/Date/Date-Calc-4.1.tar.gz* kostenlos zur Abholung bereit.

Die *Date::Calc*-Tour

Datumsangaben liegen in *Date::Calc* üblicherweise als Dreier-Array *($year, $month, $day)* vor. Das heutige Datum liefert die Funktion *Today()* folgendermaßen:

```
use Date::Calc qw(Today);
($year, $month, $day) = Today();
```

Hier zeigt sich, daß *Date::Calc* von sich aus noch keine Funktionen exportiert. So steht *Today()* erst zur Verfügung, nachdem es der Export-Liste, die der *use*-Anweisung anhängt, beiliegt. Das Jahr liegt heuer als ``1998'' vor, Monat und Tag entsprechen dem gesundem Menschenverstand, starten also beide bei *1*.

... bald ist wieder Weihnachtszeit!

Um nun zum Beispiel die Tage von heute bis Weihnachten zu zählen, muß man keine kalendertechnischen Klimmzüge absolvieren und mit Schaltjahren und ähnlichem Unbill herumjonglieren, sondern bemüht einfach *Delta_Days()*, wie in Listing *xmas.pl* vorgestellt: *Delta_Days()* nimmt zwei Datumsangaben entgegen, also insgesamt sechs Skalare. Nachdem das heutige Datum mit *Today()* ermittelt ist, steckt *xmas.pl* den 24.12. des gleichen Jahres in den Array *@xmas*, übergibt ihn *Delta_Days()* und — *schwupps!* — schon liefert es die Anzahl der Tage zwischen den zwei Datumsangaben zurück. (siehe Listing xmas.pl)

LISTING: xmas.pl

```
1 #!/usr/bin/perl -w
2
3 use Date::Calc qw(Delta_Days Today);
4
5 ($year, $month, $day) = Today();
6 @xmas = ($year, 12, 24);
7
8 $days = Delta_Days($year, $month, $day, @xmas);
9 print „Noch „, $days, „ Tage bis Weihnachten.\n";
```

Wochentage und Datumsarithmetik

Zeigt man (zum Beispiel mit dem in dieser Reihe vorgestellten Paket *Chart*) aktuelle Graphiken an, stellt sich zuweilen das Problem, daß man die Wochentage der hinter einem liegenden Woche auf die X-Achse zaubern muß — doch wo beginnen? Listing *week.pl* löst das Problem, indem es einen Array *@days* aufbaut, in dem es die Wochentags-Kürzel hintereinanderreiht. Da *Date::Calc* aus deutschen Landen stammt, bietet es eine vorbildliche Vielsprachen-Steuerung an, so daß *week.pl*, falls heute Donnerstag wäre, folgende Reihe ausgäbe:

```
Fr Sa So Mo Di Mi Do
```

Zunächst setzt die *Language()*-Funktion die verwendete Sprache auf *„Deutsch“*. *Language()* selbst verarbeitet aber keinen String als Parameter, sondern einen kleinen Integer, den kein Mensch kennt, außer der Funktion *Decode_Language()*, die den National-String in das interne Format umwandelt. Dann ermittelt *week.pl* das aktuelle Datum als Startwert und springt anschließend in eine Schleife über sieben Tage, in der es pro Durchgang um einen Tag zurückspringt.

Mit *Day_of_Week()* ermittelt es jeweils die Nummer des aktuellen Wochentags, verwandelt diese mit *Day_of_Week_to_Text()* in einen Wochentags-String voller Länge (z.B. *„Montag“*), kürzt diesen auf zwei Buchstaben und fügt ihn schließlich am Anfang (!) des Arrays *@days* ein. Um pro Schleifendurchgang einen Tag zurückzusetzen, kommt die Funktion *Add_Delta_YMD()* zum Zug, die ein dreiteiliges Startdatum und ein ebenfalls dreiteiliges Delta aus Jahren, Monaten und Tagen erwartet. Im vorliegenden Fall führt der letzte Parameter den Wert *-1*, was einem Zeitsprung von einem Tag in die Vergangenheit entspricht. (siehe Listing week.pl)

LISTING: week.pl

```
1 #!/usr/bin/perl -w
2
3 use Date::Calc qw(Today Decode_Language Language
4                    Day_of_Week Day_of_Week_to_Text
5                    Add_Delta_YMD);
6
7 Language(Decode_Language(„Deutsch“));
8
9 @date = Today();
10
11 foreach (1..7) {
12    $day = Day_of_Week_to_Text(Day_of_Week(@date));
13    unshift(@days, substr($day, 0, 2));
14    @date = Add_Delta_YMD(@date, 0, 0, -1);
15 }
16
17 print „@days\n“;
```

Tage der Woche im Monat

Die Funktion *Day_of_Week()* kalkuliert aus Jahr, Monat und Tag den Wochentag. Listing *first.pl* zeigt, wie sich die Monatsersten des Jahres 1998 berechnen lassen. Ohne explizit ausgewähltes Deutsch-Modul liefern Funktionen wie *Month_to_Text()*, die den Namen eines Monats, der als Zahl vorliegt, liefert, englische Wörter:

```
Thursday, January 1st, 1998
Sunday, February 1st, 1998
...
Sunday, November 1st, 1998
Tuesday, December 1st, 1998
```

LINUX USER GROUPS

Im folgenden die Liste der uns bekannten Linux User Groups im Deutschsprachigen Raum in Kurzfassung. Änderungen und Updates bitte der Redaktion (redaktion@linux-magazin.de) mitteilen (Name, Beschreibung, Treffpunkt, Adresse, Ansprechpartner, Homepage, Email, Tel, Fax, Mitgliederzahl,...).

Aachener Linux User Group (ALUG) http://aachen.heimat.de/alug/

Linux User Group Augsburg (LUGA) http://bsmux.rz.uni-augsburg.de/luga

Linux User Group Austria http://www.luga.or.at

Berliner Linux User Group (BeLUG) http://www.belug.org

Linux User Group Bochum (LugBo) http://www.ping.de/sites/lecter/lug.html

Linux Anwendergruppe der Universität Bochum
http://www.ruhr-uni-bochum.de/linux/

Linux User Group Bodensee (LLUGB) http://llugb.amsee.de/

Bonner Linux User Group (BoLUG) http://www.bonn.linux.de

Linux User Group Braunschweig http://www.tu-bs.de/initiativen/LUG

Chemnitz Linux User Group (CLUG) http://www.tu-chemnitz.de/linux/

Linux Users Group Darmstadt http://www.swb.de/personal/runner/dalug.html

Linux User Group Dortmund (LUGRUDO) lugrudo@outerspace.de

Linux Users Group Duisburg (LUG-DUI) http://mti26.mti.uni-duisburg.de

Düsseldorf Linux User Group (DLUG) http://www.hsp.de/~dlug/

Essener Linux-Stammtisch (ELiSta) http://www.uni-essen.de/initiative/elista/

Freiburger Linux User Group (FLUG) http://www.freiburg.linux.de

IN-Kompetent e.V. Fulda http://www.rhoen.de/

Linux User Grup Giessen (LUGG) http://lugg.tg.fh-giessen.de

Linux User Group Kassel (LUGK) jochen.hein@delphi.central.de

Unix-Gruppe der Hamburger MH e.V.
http://swt-www.informatik.uni-hamburg.de/~1willamo/hmh.html

Linux User Group Hannover (LUGH) http://www.han.de/lugh/

Projektgruppe Linux der UNIX-AG Kaiserslautern
http://www.unix-ag.uni-kl.de/~linux/

Karlsruher Linux User Group (KaLUG) http://www.karlsruhe.linux.de

Linux-Workshop Köln http://www.uni-koeln.de/themen/linux/

Leipziger Linux-Stammtisch http://rzaix340.rz.uni-leipzig.de/~linux/

Lüneburger Unix Stammtisch (LUSt) morningdew2@geocities.com

Münchner Linux Usergroup (MLUG)
http://www.muc.de/~cygnus/mlug/mlug.html

Linux Stammtisch Münster (MueSLi)
http://hottemax.uni-muenster.de/~grover/muesli.html

Linux User Group Nürnberg (LUGNü) http://www.linux.noris.de/

Oldenburger Linux Stammtisch (OLiS)
http://www.Infodrom.North.DE/Linux/Stammtisch/

Linux User Group Osnabrück (LUGO) http://www.lugo.de

Linux User Group Paderborn (LUG-PB) http://www.bowle.de/~lug-pb/

Linux User Group Saar (LUGSaar) http://www.lugs.linux.de/

UNIX-AG Siegen http://www.si.unix-ag.org

Linux User Group Stormarn
http://www.geocities.com/SiliconValley/Bay/7117/

Linux User Group Switzerland http://www.lugs.ch

Linux User Group Traunstein (LugTra) http://www.spotlight.de/lug/

Linux User Group Trier TriLUG http://www.trilug.fh-trier.de/

Linux User Group Tübingen (LUGTÜ) http://www.linux.de/lug/tuebingen/

Linux User Group TU Wien (LLL) lll@radawana.cg.tuwien.ac.at

Linux User Group Wolfsburg (LUGWo)
http://www.escape.de/user/laura/lug_wolfsburg.html

Brainstorm Computer-Club Würzburg (BraCCWü)
http://www.wuerzburg.de/brainstorm/ ???

LISTING: first.pl

```
1 #!/usr/bin/perl -w
2
3 use Date::Calc qw(Today Day_of_Week
4                   Day_of_Week_to_Text Month_to_Text);
5 $year = 1998;
6
7 foreach $month (1..12) {
8   $dow = Day_of_Week($year, $month, 1);
9
10   printf „%s, %s 1st, %d\n",
11      Day_of_Week_to_Text($dow),
12      Month_to_Text($month), $year;
13 }
```

``Laß uns regelmäßig jeden zweiten Freitag im Atzinger zusammensitzen und ein, zwei Augustiner trinken!'' — wer hat diesen Satz nicht so oder so ähnlich schon ausgesprochen. Listing *secfri.pl* zeigt, wie sich das in die Tat umsetzen läßt: Ausgehend vom aktuellen Datum, spult das Skript zwölfmal jeweils einen Monat vor, um für den aktuellen Monat mittels der Funktion *Nth_Weekday_of_Month_Year()* das Datum des zweiten Freitags zu bestimmen. Den Freitag nimmt die Funktion dabei über den dritten Parameter als *„5“* entgegen, dies enspricht der internen Numerierung von *Date::Calc* die am Montag mit *„1“* startet. Der vierte Parameter gibt an der wie*vielte* Freitag im Monat gemeint ist. Die Ausgabe ist verblüffend richtig:

```
11.09.1998
09.10.1998
...
09.07.1999
13.08.1999
```

LISTING: secfri.pl

```
1 #!/usr/bin/perl -w
2
3 use Date::Calc qw(Today Nth_Weekday_of_Month_Year
4                   Add_Delta_YMD);
5
6 my ($year, $month, $day) = Today();
7
8 foreach (1..12) { # Zwölf Monate lang
9
10   # Zum nächsten Monat vorspulen
11   ($year, $month, $day) =
12      Add_Delta_YMD($year, $month, $day, 0, 1, 0);
13
14   # 5: Freitag 2: 2. Freitag
15   ($y, $m, $d) = Nth_Weekday_of_Month_Year($year,
16                                            $month, 5, 2);
17   printf „%02d.%02d.%d\n", $d, $m, $y;
18 }
```

Logfile-Analyse des kleinen Mannes

Zum Abschluß noch ein Beispiel, das aufzeigt, wie einfach sich mit dem Kalendermodul die Log-Dateien von Webservern analysieren lassen: Eine Zeile des Access-Logs sieht üblicherweise in etwa so aus:

```
kunde.com - - [30/Jul/1998:02:31:06 -0700]
„GET /index.html HTTP/1.0“ 200 7304
```

Um festzustellen, wieviele Hits letzte Woche auf die einzelnen Wochentage verteilt eingegangen sind, iteriert man über die Zeilen der Datei, stellt fest, ob das angegebene Datum (in eckigen Klammern) in die letzte Woche fiel, und wenn, erhöht man einen Zähler für den entsprechenden Wochentag.

Ob ein angegebes Datum in einem festgelegten Zeitrahmen liegt, läßt sich dadurch bestimmen, daß man sämtliche Angaben in Tage umrechnet, die seit einem Zeitpunkt weit in der Vergangenheit vergangen sind. Die Funktion *Date_to_Days()* macht Nägel mit Köpfen und liefert zu einem vorgegeben Datum (Tag-Monat-Jahr) die Anzahl der Tage, die seit dem Urknall des Universums, nein, Spaß beiseite, seit dem 1.1. des Jahres 1 anno Domini vergangen sind. So ergibt sich für den 01.08.1998 etwa ein Wert von 729602 Tagen. Liegen Datumsangaben in solchen Zahlen vor, kann man Vergleiche einfach numerisch mit < und > durchführen.

In Listing *log.pl* drehen wir ein bißchen an der Schwierigkeitsschraube, bitte anschnallen und die Arme im Fahrzeug lassen!

Die zentrale Datenstruktur ist dort der Array *@result*, der als Elemente Referenzen auf kleine 2er-Listen enthält, die als erstes Element die Abkürzung des Wochentags und als zweites einen Zähler führen, der anzeigt, wieviele Hits am entsprechenden Wochentag schon eingegangen sind. *@result* führt genau *7* Einträge. Falls heute Donnerstag ist, steht dort für jeden Tag von Freitag letzter Woche bis heute jeweils ein Element, das den Wochentag mitsamt dem zugehörigen Zähler enthält.

Um diese Struktur anfangs aufzubauen, könnten wir wieder auf die Logik nach Listing *week.pl* zurückgreifen, aber nach dem Motto *Öfter mal was Neues!* legen wir diesmal in *@days* eine Liste mit *Mo* bis *So* ab, bestimmen das heutige Datum (*@date*) mitsamt der zugehörigen Wochentagszahl (*$dow*) und springen (Zeile 11) in eine Schleife mit 7 Durchgängen, die

LISTING: log.pl

```
1 #!/usr/bin/perl
2
3 use Date::Calc qw(Today Day_of_Week Decode_Month
4                    Add_Delta_YMD Date_to_Days);
5
6 @days = qw/Mo Di Mi Do Fr Sa So/;
7
8 @today = Today();                # Heutiges Datum
9 $dow   = Day_of_Week(@today);  # Wochentag (Nummer)
10
11 foreach ($dow..$dow+6) {
12     push(@result, [$days[$_ % 7], 0]);
13 }
14                          # 'Fenster' für Woche
15 @six_days_back = Add_Delta_YMD(@today, 0, 0, -6);
16 $window_from    = Date_to_Days(@six_days_back);
17 $window_to      = Date_to_Days(@today);
18
19                          # Access-Log-Datei bearbeiten
20 open(LOGFILE, „<access.log") ||
21     die „Cannot open file access.log";
22
23 while(<LOGFILE>) {    # Datum steht in [...] im
24                          # Format dd/Mon/yyyy:hh:mm:ss
25     if(m#\[(\d+)/(\w+)/(\d+)#) {
26         ($day, $monthname, $year) = ($1, $2, $3);
27         $month = Decode_Month($monthname);
28         @date = ($year, $month, $day);
29
30                          # Prüfen, ob Datum in
31                          # letzter Woche liegt
32         $days = Date_to_Days(@date);
33         if($days >= $window_from &&
34            $days <= $window_to) {
35                          # Wochentag-Zähler erhöhen
36             $result[$days-$window_from]->[1]++;
37         }
38     }
39 }
40 close(LOGFILE);
41
42                          # Wochen-Report ausgeben
43 foreach $result (@result) {
44     my ($weekday, $count) = @$result;
45     printf „$weekday: %d\n", $count;
46 }
```

jeweils den richtigen Wochentag aus *@days* holt und mitsamt eines auf *0* vorbesetzten Zählers in *@result* einfügt. Die Modulo-Logik *$_ % 7* setzt dabei den Index des aktuell in *@days* gelesenen Elements wieder an den Anfang zurück, falls der Zähler über das Ende des Arrays hinausschießt.

Die Zeilen 15-17 legen ein Zeitfenster für die vergangene Woche fest, *$window_from* und *$window_to* sind jeweils die zwischen Anno Tobak und dem Anfang bzw. Ende des Zeitfensters vergangenen Tage. Kommt also ein ebenfalls in diesem Format dargestelltes Datum daher, läßt sich leicht feststellen, ob es innerhalb oder außerhalb des Fensters liegt.

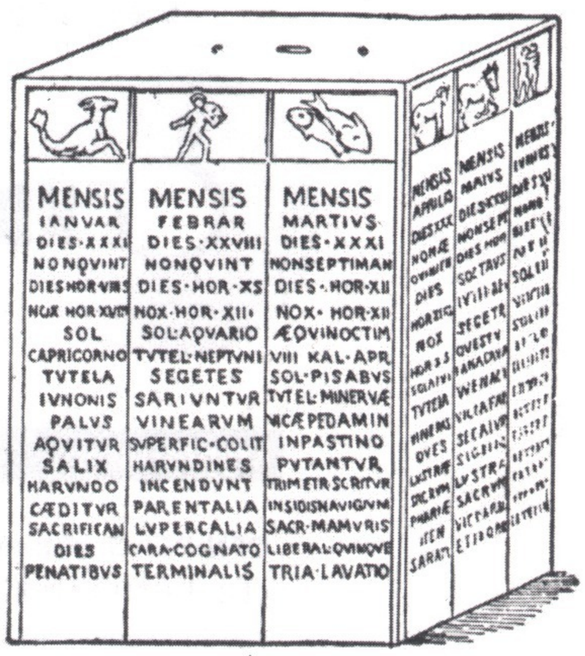
Zeile 20 öffnet die Log-Datei zum Lesen, der reguläre Ausdruck in Zeile 25 fördert für jede durchlaufene Zeile das Zugriffsdatum zutage. Nun hat *Date::Calc* zwar die Funktion *Parse_Date()*, die Datumsstrings im Format

```
Thu Jul  9 19:18:28 PDT 1998
```

importiert, doch für eine Funktion, die das in der Log-Datei verwendete Format *18/Aug/1997:20:58:42 -0700* beherrscht, suchte ich vergebens — egal, *log.pl* macht das *zu Fuß*. Da der Monat in dem oben einsehbaren dreibuchstabigen Kürzel vorliegt, muß er mit der Funktion *Decode_Month()* aus *Date::Calc* in eine Zahl von *1-12* umgewandelt werden, bevor *Date_to_Days()* zum Einsatz kommt und den Tageswert berechnet. Die Zeilen 33 und 34 stellen dann fest, ob das gefundene Datum im eingestellten Bereich liegt. Ist dies der Fall, greift Zeile 36 auf das *@result*-Element am Index *$days - $windows_from* zu (entspricht dem richtigen Wochentag in *@result*) und erhöht den Zähler (zweites Element der Unter-Liste) um Eins.

Bleibt nur noch, in den Zeilen 43-46 den Array *@result* zu durchlaufen und das Ergebnis auszugeben — man stelle sich vor, was man daraus mit David Bonners *Chart*-Paket zaubern könnte! (s. List.log.pl)

Wie jedem guten Modul liegt auch *Date::Calc* ausführliche Dokumentation bei, die nach der Installation mit *perldoc Date::Calc* zum Vorschein kommt. Viel Spaß damit!



Hurra, die Zwölf!

Ein ganzes Jahr Perl-Snapshot ist um, meine lieben Leser und Leserinnen: Zeit, ein bißchen in sich zu gehen und über die Zukunft nachzudenken. Hat's Euch gefallen? Was gibt's zu verbessern? Was interessiert Euch besonders? Rafft's Euch auf, schreibt's, und sagt's mir, ob Ihr noch weitere G'schichterln aus Perl-Land lesen wollt. Wenn ich genug Fanpost und Themenvorschläge kriege, häng' ich glatt noch ein Jahr dran, soviel Spaß hat's gemacht! Laßt was hören!

DER AUTOR

Michael Schilli arbeitet als Web-Engineer für America Online Inc., San Mateo. Er ist Autor des im Juni 1998 bei Addison-Wesley erschienenen Buches *„GoTo Perl 5"* und unter der EMail-Adresse *mschilli1@aol.com* oder *http://mission.base.com/mschilli* zu erreichen.

Autorenliste

Georg Basse

Marco Budde

Robert Gehring

Bernhard Kuhn

Thorsten Kukuk

Patrick Murmann

Christian Perle

Winfried Schüfer

Michael Schilli

Tom Schwaller

Mark Vogelsberger

Peter Wüchtler

Oliver Zendel

Service für DeLUG-Mitglieder

KDE 1.0 auf CD

Krypto-Info:

DFN-PCA Schlüssel: 2048/35DBF565 1997/04/16 DFN-PCA, CERTIFICATION ONLY KEY (Low-Level: 1997-1998) <not-for-mail>
Key fingerprint = 09 7C 09 19 D3 C3 86 DC 7A 30 15 11 12 95 8D E3

IN-Root-CA Schlüssel: 2048/19980101 SIGN EXPIRE:1999-12-31
Key fingerprint = B3 06 9A 8D 38 04 3C 75 41 32 EE DC 8B 7D 61 0D
http://www.in-ca.individual.net/
Hashwert des Ewigen Logfiles vom 27.07.1998 15:40:30
sha1=ca7b2e5cda1ef0a3563e298de271b2e66f6698f0
https://www.iks-jena.de/mitarb/lutz/logfile/

Inserentenverzeichnis

IMPRESSUM

Informationen, Berichte und Neuigkeiten aus der Linux-Welt

Herausgeber: Rudolf Strobl
Redaktion:
Email: redaktion@linux-magazin.de
Post: **Linux - Magazin Verlag**
Stefan-George-Ring 23
81929 München
Tel.: 089/99315-310
Fax: 089/99315-329
WWW: http://www.linux-magazin.de
ISSN: 1432 - 640 X
Chefredakteur: Tom Schwaller,
tom.schwaller@linux-magazin.de (ts)
Coverdesign: Tilo Wondollek, TW-Design
Layout : M &. S Art&Work
Mitarbeiter: siehe Autorenliste
Vertriebsmarketing: info@linux-magazin.de
Aboverwaltung: Monika Scheck
Tel: 089/99315-310
Email: info@linux-magazin.de
Anzeigen: Agentur Jaser
Plattenweg 4
86850 Fischach
Tel: 08204 90040
Fax: 08204 90050
Email: pjaser@t-online.de

Anzeigenpreise: Es gilt die Anzeigenpreisliste vom 01.12.1997

Das Linux - Magazin erscheint monatlich.
Einzelheft: **DM 9,80**
Preis des Jahresabonnements:
Incl. DeLUG-Mitgliedschaft **DM 180,-**
Ohne Mitgliedschaft **DM 103,-**
(Ausland: 115,-)

Für Schüler und Studenten 20% Ermäßigung (Vorlage Schüler- oder Studentenausweis)