7/98

Das einzige deutschsprachige Linux-Magazin

LINUX MAGAZIN

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

7/98

Linux News

## Neues aus dem Internet

Astronomie

## Bildverarbeitung in der Astronomie

Neue Serie

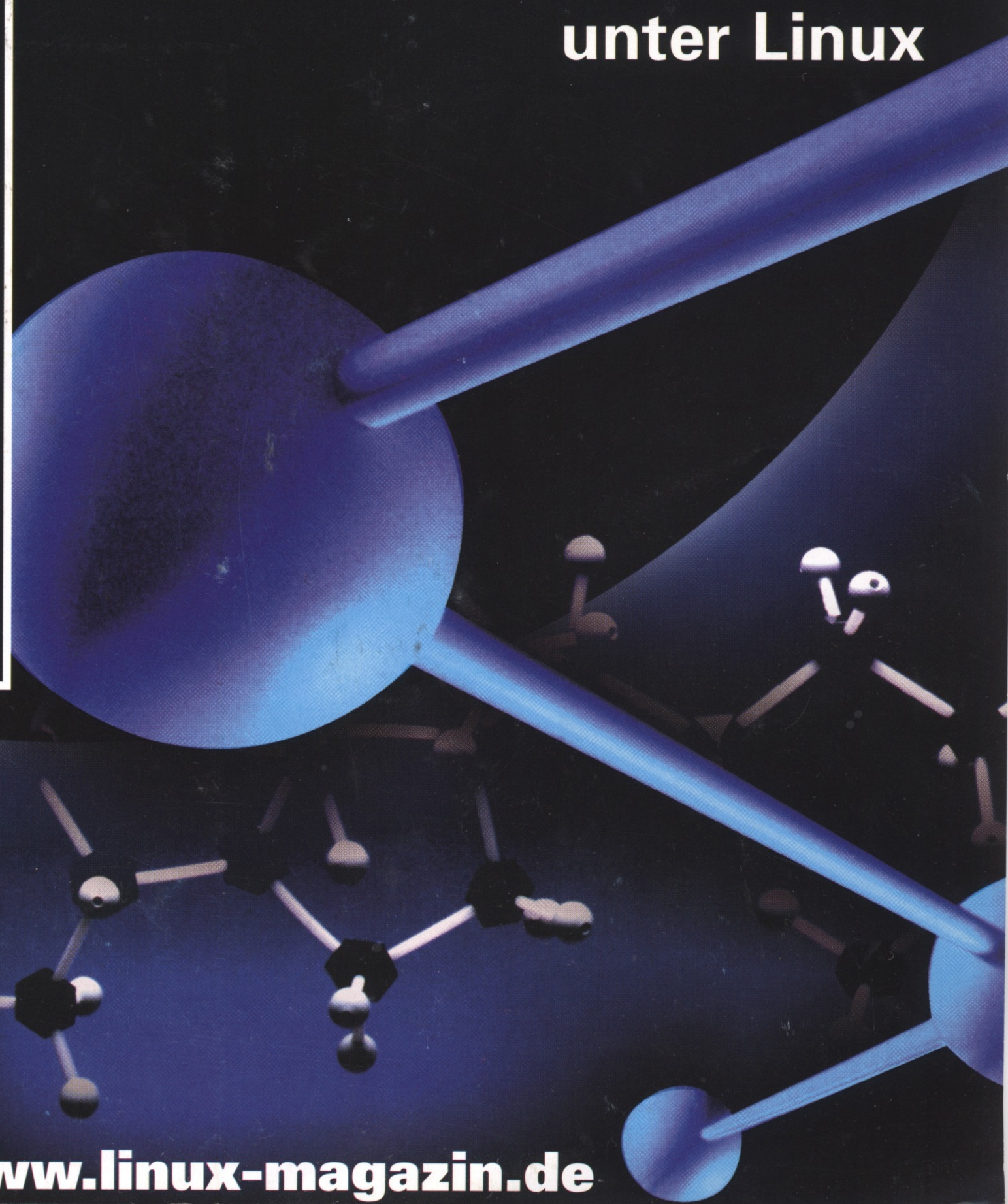
## Netzwerk-Management

Best of

## LinuxTag 98

# Interaktive Moleküle

## Chemie-Software unter Linux



www.linux-magazin.de

Perl Snapshot

# Clipping-Dienst

von Michael Schilli

**In der heutigen Folge unserer Perl-Reihe generieren wir mit einem Clipping-Script die tägliche Zusammenfassung unserer drei Lieblingsseiten.**

Liest der Bundeskanzler jeden Morgen alle Zeitungen? Nein, der feine Herr nutzt einen Clipping-Dienst. Das ist ein Stab von Leuten, die sich jeden Tag durch sämtliche bekannten Blätter wühlen, wichtige Artikel ausschneiden und eine Mappe zusammenstellen, mit der Creme de la Creme des Pressewesens, sozusagen.

*Mein* Arbeitspensum freilich läßt das des Kanzlers wie einen Hawaii-Aufenthalt erscheinen! Leider darf ich auch nicht soviel Geld verprassen, daß ich andere Leute zum Zeitunglesen anstellen könnte. Darum habe ich mir kurzerhand ein kleines Skript zusammengestellt, das die wichtigsten Neuigkeiten des Tages vom Internet pumpt: Den aktuellen Dilbert-Comic, die Schlagzeilen des Bayrischen Rundfunks und die aktuelle Erdbebenkarte der San-Francisco-Gegend, damit ich weiß, ob's letzte Nacht wirklich gerumpelt hat oder ob's doch nur wieder ein Bier zuviel war — *kommt leider vor*!

Die Unix-Kiste in der Arbeit, die eh die ganze Nacht läuft, ruft morgens um sieben das Skript *clip.pl* auf, das alles zusammensucht und Texte und Bilder auf eine einzige Webpage packt. So kann ich alle Daten auf einen Schlag einsehen, ohne unnötig Zeit mit Klicken und Warten zu verplempern.

## *CGI.pm* ist Dein Freund

Das Skript aus Listing *clip.pl* nutzt für die HTML-Ausgaben das schon ausführlich vorgestellte Modul *CGI.pm* von *Lincoln D. Stein*. Damit *CGI.pm*, falls das Skript vom *cron* ohne Parameter von der Kommandozeile aus aufgerufen wird, nicht endlos auf CGI-Parameter aus der Standardeingabe wartet, bietet das Modul ab Version *2.38* den Schalter *-noDebug* an. Der *use*-Befehl mit der angehängten Tag-Liste in Zeile 3 exportiert also diejenigen Funkionen aus *CGI.pm*, die Standard- und Tabellen-HTML-Tags erzeugen, läßt aber gleichzeitig das Skript normal von der Kommandozeile laufen.

Die Funktionen *get* und *getstore* aus dem Modul *LWP::Simple* aus der Bibliothek *libwww* von *Gisle Aas* holen Webseiten vom Netz. *get* liefert den Inhalt der betreffenden Webseite als String zurück, während *getstore* ihn gleich in einer angegebenen Datei auf der Festplatte ablegt. Zeile 4 importiert *getstore*, *get*, sowie das Fehlermakro *RC_OK* aus *LWP::Simple*. Die kompakten *LWP::Simple*-Funktionen kommen immer dann zum Einsatz, wenn keinerlei Redirects oder Authorisierungsmaßnahmen den URL-Zugriff erschweren - falls doch, müsste der *LWP::UserAgent* aus derselben Programmsammlung 'ran.

Die Zeilen 19 und 20 schreiben mit den praktischen Funktionen aus *CGI.pm* den HTML-Start-Tag und den Anfang einer Glossar-Liste, die später im Format

```
<DL>
 <DT>Quelle <DD>Inhalt
 <DT>Quelle <DD>Inhalt
 ...
</DL>
```

in der in Zeile 16 geöffneten Ausgabedatei *clip.html* stehen wird. Der *chdir*-Befehl aus Zeile 14 versetzt das Skript in das Verzeichnis, in dem später alle Ausgabedateien liegen sollen und sorgt so dafür, daß das Skript immer dorthin schreibt, auch wenn Dateien ohne Pfad angegeben werden. Mit den Konfigurationsparametern aus den Zeilen 9 und 10 läßt sich dieses Verzeichnis sowie der Name der Ergebnisdatei an die lokalen Erfordernisse anpassen.

## BR-Nachrichten

Der Bayerische Rundfunk aktualisiert stündlich eine

Webpage, die mit etwa fünf Schlagzeilen einschließlich kleiner Dreizeiler genau das Maß an Politik bietet, das ich noch vertragen kann, ohne mich zu langweilen. Da nicht die gesamte Seite einschließlich aller Logos und Werbung interessiert, schneidet *clip.pl* den HTML-Text zwischen den Tags *<A NAME="1"* und *</TT>* aus — irgendwann einmal habe ich herausgefunden, daß zwischen diesen Tags die Schlagzeilen stehen. Die Funktion *grab_and_grep*, die ab Zeile 63 definiert ist, nimmt als Argumente einen URL und einen regulären Ausdruck entgegen. Nachdem *grab_and_grep* die Seite mit der *get*-Funktion (*LWP::Simple*) geholt hat, prüft sie, ob deren Inhalt irgendwo auf den als String hereingereichten regulären Ausdruck paßt. Die *eval*-Anweisung aus Zeile 75 konstruiert zur Laufzeit die Befehlsfolge

```
$doc =~ /<A NAME="1".*<\/TT>/s; return $&
```

## Listing: clip.pl

```
1  #!/usr/bin/perl -w
2  ###################### Module ######################
3  use CGI qw/:standard :html3 -noDebug/;
4  use LWP::Simple qw/get getstore RC_OK/;
5  use HTML::TreeBuilder;              # HTML-Parser
6
7
8  ################## Konfiguration ###################
9  $clipdir  = „/home/mschilli/clip“;
10 $htmlfile = „clip.html“;
11
12
13 ################## Datei öffnen ####################
14 chdir($clipdir) || die „Cannot chdir to $clipdir“;
15
16 open(OUT, „>$htmlfile“) ||
17     die „Cannot open $htmlfile“;
18                                    # Titel
19 print OUT start_html('-title' => „Clipping-Dienst“);
20 print OUT dl;                      # Listenanfang
21
22
23 ##### Kurznachrichten des Bayrischen Rundfunks #####
24 $ret = grab_and_grep(
25            „http://www.br-online.de/news/aktuell“,
26            „/<A NAME=\“1\“.*<\\/TT>/s“);
27
28 print OUT dt(h1(„Bayrischer Rundfunk“)), dd($ret);
29
30
31 ############ Erbeben-Karte der Bay-Area ############
32 $url = „http://quake.wr.usgs.gov/recenteqs“ .
33        „/Maps/SF_Bay.gif“;
34 $localfile = „quake.gif“;
35
36 print OUT dt(h1(„Aktuelle Erbeben-Karte“));
37
38 if(getstore($url, $localfile) == RC_OK) {
39     print OUT dd(img({src => „quake.gif“}));
40 } else {
41     print OUT dd(„Cannot get $url“);
42 }
43
44
45 ############### Dilbert Comic-Strip ################
46 $url = „http://www.unitedmedia.com/comics/dilbert/“;
47 $localfile = „dilbert.gif“;
48
49 print OUT dt(h1(„Dilbert“));
50
51 if(dilbert_to_file($url, $localfile) == RC_OK) {
52     print OUT dd(img({src => „dilbert.gif“}));
53 } else {
54     print OUT dd(„Cannot get $url“);
55 }
56
57 ###################### Abschluß ####################
58 print OUT end_html;
59 close(OUT);
60
61
62 ########################################################
63 sub grab_and_grep {
64 ########################################################
65 # Webseite holen und Text nach angegebenen
66 # Muster extrahieren
67 ########################################################
68     my ($url, $regex) = @_;
69
70     my $doc;
71
72     ($doc = LWP::Simple::get $url) ||
73         return „Cannot get $url“;
74
75     eval „\$doc =~ $regex; return \$&“;
76 }
77
78 ########################################################
79 sub dilbert_to_file {
80 ########################################################
81 # Dilbert-Seite ($url) holen, Comic-URL extra-
82 # hieren, GIF holen und lokal in $file speichern
83 ########################################################
84     my ($url, $file) = @_;
85
86     my ($doc, $abslink, $link);
87
88     ($doc = get $url) || return 0;
89
90     my $tree = HTML::TreeBuilder->new->parse($doc);
91
92     for (@{$tree->extract_links(qw/img/)}) {
93         $link = URI::URL->new($_->[0]);
94         $abslink = $link->abs($url);
95         last if $abslink =~ /dilbert\d+\.gif$/;
96     }
97
98     $tree->delete();              # Parse-Baum löschen
99
100    getstore($abslink, $file); # Lokal speichern
101 }
```

und führt sie aus. Der reguläre Ausdruck strebt im Dokument *$doc* eine maximale Abdeckung (.*) zwischen den Tags *<A NAME="1"* und *</TT>* an, wegen des */s*-Modifizierers schluckt .* zeilenübergreifend Zeichen. Das Teildokument, auf das der reguläre Ausdruck paßt, liegt anschließend in der Spezial-Variablen *$&*, die die folgende *return*-Anweisung an das Hauptprogramm zurückgibt.

Dieses nutzt die Funktionen *dt* und *dd* aus *CGI.pm*, um die extrahierte Information schön als Glossarliste strukturiert in der Ergebnisdatei abzulegen.

Die HTML-Tags, auf die der Code anspringt, können sich natürlich kurzfristing ändern. Falls der Bayrische Rundfunk das Format der Webseite ändert, muß *clip.pl* entsprechend nachgezogen werden. Entsprechendes gilt für die nachfolgenden Funktionen: auch URLs können sich kurzfristig ändern.

## Hat's gerumpelt?

Graphisch aufbereitete Daten über die letzten Erbeben, die sich in der Bay-Area ereigneten, liegen als GIF-Bild auf einem Server der US-Regierung. Diese Datei vom Netz zu ziehen und lokal abzuspeichern, ist mit *LWP::Simple* und *getstore* ein Kinderspiel. Entspricht der Rückgabewert dem Wert des Fehlermakros *RC_OK*, ging alles gut. *RC_OK* ist nicht etwa ein Skalar, sondern eine von *LWP::Simple* exportierte *Funktion*. Im „Gutfall" fehlt nur noch, einen *IMG*-Link in unsere Clipping-Datei zu schreiben, der auf die Quake-Datei zeigt — fertig ist der Lack!

## Dilbert

United Media veröffentlicht täglich einen neuen Dilbert-Comic, den Leute wie ich, die in einem Großraumbüro mit Stellwand-Quadraten (``Cubicles'') arbeiten, natürlich unbedingt lesen müssen. Leider verunzieren die Komiker dort die Seite mit Werbung, und damit's nicht ganz so einfach ist, nur den Strip zu extrahieren, hängen sie an den Image-Namen das Datum und die (unvorhersagbare) Uhrzeit dran, z. B. *dilbert980118104253.gif*.

Da muß schweres Gerät ran, die Funktion *dilbert_to_file* zeigt ab Zeile 79, wie es geht: Die *get*-Funktion holt die Seite, die unter anderem irgendwo den Image-Link enthält, um sie anschließend mit dem *HTML::Treebuilder* zu parsen. Das Parse-Objekt verfügt über die Methode *extract_links*, die das *SRC*-Attribut aller Tags vom Format *<IMG SRC=...>* herausfiltert, falls, wie im Listing, die ihr übergebene Liste das Element *img* enthält. *extract_links* gibt dabei eine Referenz auf einen Array zurück, der als Elemente für jeden gefundenen *SRC*-Attributwert eine Referenz auf einen weiteren Array enthält, welcher wiederum als erstes Element den URL des Bildes führt, der wiederum ... kleiner Scherz, der URL ist, was wir brauchen :).

Dieser URL ist im Falle der United-Media-Seite relativ, also auf den URL der Seite bezogen. Um das Bild vom Netz zu holen, muß er absolut, also als sauberer *http://...*-URL vorliegen. Für die Umwandlung bietet sich die *abs*-Methode des *URI::URL*-Pakets an, also wird flugs ein neues *URI::URL*-Objekt mit dem relativen Link erzeugt und die *abs*-Methode mit dem Basis-URL der United-Media-Seite (*$url*) aufgerufen, worauf *$abslink* in Zeile 94 den absoluten URL enthält.

Da auf der Seite natürlich mehrere Links zu finden sind, extrahiert die *for*-Schleife einen nach dem anderen, bis einer daherkommt, der wie das Dilbert-Bild aussieht: */dilbert\d+\.gif$/* ist der passende reguläre Ausdruck, der auf Strings im Format *___dilbert980118104253.gif* wartet.

Was bleibt, ist nur, den Parse-Baum zu löschen, das Bild mit *getstore* vom Netz zu holen und auf der Platte zu speichern.

## Clipping um sieben in der Früh'

Nun muß noch ein Eintrag in die Tabelle des *cron*, damit dieser das Skript jeden Tag ausführt, sagen wir um sieben in der Frühe:

```
00 7 * * * /home/mschilli/bin/clip.pl
```

Dann enthält *clip.html* im eingestellten Verzeichnis kurze Zeit später alle gewünschten Informationen und auch die Zusatz-Dateien *quake.gif* und *dilbert.gif* liegen im gleichen Verzeichnis vor. Kommt der schlaftrunkene Anwender um neun ins Büro, zeigt der Browser, falls er ihn mit *File -> Open File* auf *clip.html* einstellt, eine schöne Clipping-Mappe an. Schon wieder kein Erdbeben? Muß wohl doch das Bier gewesen sein gestern nacht...


### DER AUTOR

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