11/1997
ISSN 1432 - 640 X

Informationen, Berichte und Neuigkeiten aus der LINUX-Welt

LINUX MAGAZIN

11/1997

DM 9,- ÖS 75,- SFR 9,-

www.linux-magazin.de

Titel story

HAWKEYE

Netzwerkdienste integriert verwalten

Erstes & einziges deutschsprachiges LINUX MAGAZIN

RC4-Beispielimplementierung

```
#include <stdio.h>
#include <string.h>

unsigned char sbox[256], k[256], key[6];

void init()
{
        unsigned i, j;
        unsigned char *kptr = key, trans;
        for(i=0;i<256;i++)
                sbox[i] = (unsigned char)i;
        for(i=0;i<256;i++) {
                k[i] = *kptr++;
                if(!*kptr)
                        kptr = key;
        }
        for(i=0,j=0;i<256;i++) {
                j = (j + sbox[i] + k[i]) % 256;
                trans = sbox[i];
                sbox[i] = sbox[j];
                sbox[j] = trans;
        }
}

void crypt(unsigned char *s)
{
        unsigned i, j, t;
        unsigned char trans;
        init();
        for(i=0,j=0;*s;*s++) {
                i = (i + 1) % 256;
                j = (j + sbox[i]) % 256;
                trans = sbox[i];
                sbox[i] = sbox[j];
                sbox[j] = trans;
                t = (sbox[i] + sbox[j]) % 256;
                *s ^= sbox[t];
        }
}

int main()
{
        unsigned char *msg = "Linux Magazin";
        unsigned n, len = strlen(msg);
        printf("Sch_ssel    : ");
        for(n=0;n<5;n++) {
                key[n] = getch();
                putchar('*');
        }
        key[5] = '\0';
        printf("\n\nKlartext    : %s\n", msg);
        crypt(msg);
        printf("Chiffretext: ");
        for(n=0;n<len;n++)
                printf("%0x ", msg[n]);
        crypt(msg);
        printf("\nKlartext 2 : %s\n", msg);
        return 0;
}
```

Soviel für heute. Nächstes Mal werden die mathematischen Grundlagen von Public-Key-Verfahren wie *pgp* besprechen.

WEITERE INFOS

[1] Schneier, Bruce: Angewandte Kryptographie. Bonn, u. a.: Addison-Wesley 1996

DER AUTOR

Oliver Müller ist hauptsächlich als Autor und Software-Entwickler tätig. Seine bevorzugten "Spielwiesen" sind die Systemprogrammierung, der Compilerbau, die Kryptologie und der Software-Entwurf, sowie zwischendurch ein bißchen Künstliche Intelligenz. Im Zuge seiner Firma "OMSE - Oliver Müller Software-Entwicklung" befaßt er sich derzeit hauptsächlich mit der Mustererkennung, plattformübergreifenden Tools & Utilities und dem Jahr-2000-Problem. Zu erreichen ist er unter *ogmueller@t-online.de*.

Indizieren und Suchen mit WAIS

Nadel im Heuhaufen

von Michael Schilli

Auf dem Comprehensive Perl Archive Network (CPAN) lagert eine schier unglaubliche Fülle an Perl-Modulen. Was man mit ihnen alles anstellen kann, *zeigt unsere neue Reihe. In der heutigen Folge geht* es um das Indizieren und Suchen mit WAIS.

Wie arbeiten AltaVista, Hotbot, Yahoo und Konsorten? Sie hetzen Herden von Robots, Spiders und ähnlichem Getier 24 Stunden am Tag quer durchs Internet und schnappen sich HTML-Seiten. Daß eine Search-Engine jedoch auf die Frage *Welche Dokumente enthalten die Daten, die ich suche?* kluge Antworten geben kann, erfordert mehr: Ausgefeilte Software muß die Dokumente analysieren und eine Schlagwort-Datenbank aufbauen. Auf eine Suchanfrage (Query) hin spuckt ein derartiges System die Namen der Dokumente aus, die die verlangten Schlagworte möglichst oft enthalten und - so jedenfalls die Theorie - inhaltlich am ehesten dem Gesuchten entsprechen.

Die private Search-Engine

Auch wenn man nicht gerade den Ehrgeiz hat, das gesamte Internet zu durchstöbern, sondern nur eine größere Website mit einer intelligenten Suchfunktion ausstatten will, steht man früher oder später vor dem Problem der Indizierung. Zum Glück gibt's hierfür bereits funktionie-

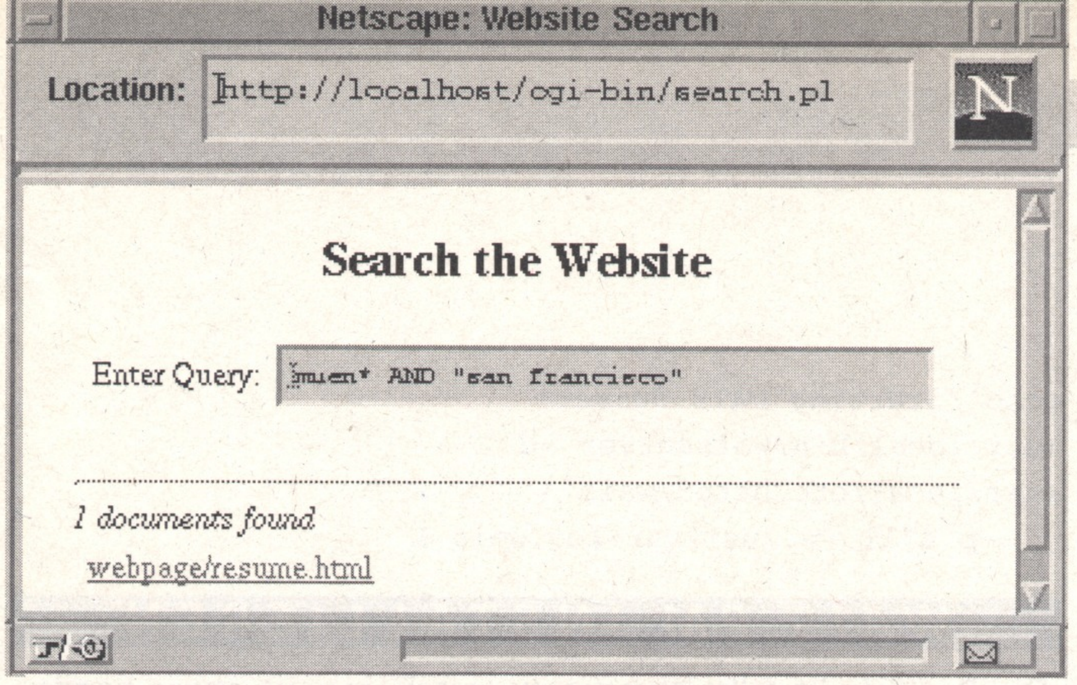

Abb.1: Stichwortsuche mit dem CGI-Skript

rende Software! Und sogar kostenlos!

freeWAIS-sf basiert auf einem Produkt der (mittlerweile nicht mehr existierenden) Firma WAIS zur Indizierung von frei formatierten Texten bzw. der Stichwortsuche in denselben. *freeWAIS-sf* läuft unter Linux und anderen gängigen Unix-Plattformen und ist ruck-zuck installiert (siehe Kasten: freeWAIS-sf*: Installation und Indizierung*).

Lauscht der neue WAIS-Server einmal auf dem eingestellten Port, dürfen Clients wie das nachfolgend vorgestellte CGI-Skript *search.pl* Suchanfragen stellen und bekommen passende Dokumente als Antwort.

In Aktion

Abbildung 1 zeigt *search.pl* in Aktion. Es liegt im *cgi-bin* des Webservers und gibt gerade darüber Auskunft, welche Dokumente auf meiner lokalen Test-Website die Begriffe *muen** (könnte auf die bayrische Landeshauptstadt passen) und *san francisco* enthalten, und, siehe da, das einzig passende ist - mein Lebenslauf, der, relativ zum Webserver im Unterverzeichnis *webpage* hängt. Selbstverständlich fördert ein Mausklick auf den angezeigten Namen das entsprechende Dokument zutage.

Die WAIS-Engine versteht - und das ist der Vorteil gegenüber einfachen Suchprogrammen wie *grep* - auch komplexere Suchbegriffe wie

```
a AND b      # Dokumente, die 'a' und 'b' enthalten
a OR b       # Dokumente, die 'a' oder 'b' enthalten
a NOT b      # Dokumente, die 'a', aber nicht 'b' enthalten
a b          # Dokumente, die 'a' oder 'b' enthalten
"a b"        # Dokumente, die die Zeichenkette
             # "a b" enthalten
a*           # Dokumente, die Wörter enthalten,
             # die mit 'a' beginnen
```

und liefert wirklich nur die Dokumente, deren Inhalt allen Bedingungen genügt.

Die Perl-Schnittstelle *Wais.pm* von *Ulrich Pfeifer*, vom CPAN geholt und installiert (siehe Kasten Wais.pm *installieren*), bietet derart komfortablen High-Level Zugriff auf den WAIS-Server, daß *search.pl* mit nur 77 Zeilen Code auskommt. Es nimmt einen Query-String vom Benutzer entgegen, sendet ihn an den WAIS-Server, erhält von dort die Namen passender Dokumente und zeigt diese anschließend als HTML-Hyperlinks an.

So wird's gemacht

Wie aus Listing 1 hervorgeht, greift sich *search.pl* in Zeile 6 das für CGI-Skripts unverzichtbare *CGI.pm* (kommt mit *perl5.004*) und importiert auch gleich - entgegen meiner sonstigen Maxime - die wichtigsten Funktionen. Dies hat zur Folge, dass auch ohne ein explizit erzeugtes CGI-Objekt Funktionen wie *header()* (gibt den HTTP-Header aus) oder *h1()* (schreibt eine HTML-Überschrift) zur Verfügung stehen. Zeile 7 holt das frisch installierte *Wais.pm*-Modul herein. Die Zeilen 12 bis 15 definieren Parameterwerte, die an lokale Gegebenheiten anzupassen sind. Da der WAIS-Server Zugriffspfade von Dokumenten entsprechend dem File-System spezifiziert, muß das Skript zur Umwandlung von lokalen File-System-Pfaden in globale URLs wissen, wo das Dokumentenverzeichnis des HTTP-Servers relativ zur Unix-Wurzel liegt (*$htdocs_path*). *$hostname* ist der Name des Rechners, auf dem der WAIS-Server läuft, *$port* der zugehörige Port, *$dbname* der Name der Datenbank - alles Parameter, die bei der Installation bzw. der Indizierung festgelegt werden.

Zeile 18 ruft die Funktion *print_form()* auf, die ab Zeile 59 den HTTP-Header sendet und das Eingabeformular in den Browser zaubert. Liegt, wie beim ersten Aufruf von *search.pl*, keine Query-Eingabe vor, beendet Zeile 20 das Skript ab. Trägt der Benutzer dann den Suchstring in das angezeigte Textfeld ein und drückt die Eingabetaste, startet *search.pl* erneut - und diesmal liefert *param("query")* den Query entsprechend dem Inhalt des Textfeldes.

Die Funktion *Wais::Search* in Zeile 23 nimmt Kontakt zum WAIS-Server auf, der wiederum seine Datenbank durchsucht und das Ergebnis wieder über die aufgebaute TCP-Verbindung zurückschickt. *search.pl* sieht nichts von alledem - nur *$result*, ein Objekt der Klasse *Wais::Result*. Die Methode *header()* gibt ein Array von Referenzen zurück, deren jede wiederum auf ein Array mit den Elementen

```
$tag, $score, $lines, $bytes, $headline
```

verweist. *$tag* ist hierbei der Name der WAIS-Datenbank, die für das Ergebnis verantwortlich zeichnet, *$score* die Trefferquote, *$lines* und *$bytes* die Länge des gefundenen Dokuments in Zeilen bzw. Bytes. In *$headline* liegen der Dateiname des Dokuments und, durch Leerzeichen getrennt, der Zugriffspfad relativ zum Dateisystem.

Zeile 33 ermittelt die Länge des Ergebnis-Arrays - die Anzahl der Treffer. Doch, halt, eine Ausnahme gibt's: Für den Fall, daß nichts gefunden wurde oder ein Fehler aufgetreten ist, kommt ein Eintrag zurück, dessen Score-Feld auf Null gesetzt ist.

Ist, wie Zeile 28 prüft, die Anzahl der Ergebnisse gleich Null (*$#results* liefert Eins weniger als die Länge von *@results*), muß ein Fehler aufgetreten sein, wahrscheinlich läuft der WAIS-Server nicht. *search.pl* zeigt daraufhin eine Fehlermeldung in roter Schrift an und bricht ab.

Sonst gibt Zeile 40 die (korrigierte) Anzahl der Treffer aus und ab Zeile 42 beginnt eine *for*-Schleife, die über alle Treffer iteriert, die Pfad-Datei-Informationen vom WAIS-Server in URLs umwandelt und als HTML-Links in einer Tabelle anzeigt. Die *td()*, *TR()*, *a()* und *table()*-Funktionen des *CGI.pm*-Moduls sehen zwar kryptisch aus, verkürzen jedoch den Code beträchtlich, und verleihen einem, hat man sie mal verstanden - grenzenlooose Maaaacht, huaaah ...

freeWAIS-sf: Installation und Indizierung

freeWAIS-sf von *Ulrich Pfeifer* zeichnet sich gegenüber dem originalen WAIS-Engine durch die Einbindung sogenannter *structured fields* aus: in Dokumenten eingebettete Felder, die dem Indizierer ein wenig von der Dokument-Struktur vermitteln sollen, statt nur auf den Inhalt loszugehen. Die im Artikel besprochene

Anwendung macht von diesem Feature keinen Gebrauch, schaden tut's jedoch nicht.
Die neueste Version von *freeWAIS-sf* ist *freeWAIS-sf-2.1.2.tar.gz* und liegt auf

```
ftp://ftp.wsct.wsc.com/pub/freeWAIS-sf/freeWAIS-sf-2.1/
```

oder einem der deutschen Spiegel, z.B.

```
ftp://ftp.leo.org/pub/comp/infosys/wais/freeWAIS/free-WAIS-sf-2.1/
```

zur Abholung bereit. Ausgepackt, konfiguriert, ge-maked und installiert wird folgendermaßen:

```
tar zxfv freeWAIS-sf-2.1.2.tar.gz
cd freeWAIS-sf-2.1.2
./configure
make
make install
```

./configure wirft eine Reihe von Fragen auf, die jedoch durch stetes Hämmern auf die Return-Taste schnell verschwinden. *make install* erfordert - im allgemeinen - Root-Rechte. Der fertige Build wird später noch benötigt.
Dafür, daß der noch zu startende WAIS-Server die Dokumente der Website in seinem Index findet, sorgt das folgendes Shell-Script, welches das mit der *freeWAIS-sf*-Installation kommende Programm *waisindex* aufruft und ihm den Pfad zur Website, sowie das Verzeichnis, indem der Index abgelegt wird, angibt:

```
WAISDIR=/usr/local/etc/httpd/wais
WEBSITE=/usr/local/etc/httpd/htdocs
WAISINDEX=/usr/local/bin/waisindex

# Verzeichnis für WAIS-Datenbank anlegen
if [ ! -d $WAISDIR ]
then
    mkdir $WAISDIR
fi
cd $WAISDIR

# Indizierung starten
$WAISINDEX -r -d website
$WEBSITE
```

Die *WEBSITE*-Variable gibt den Pfad an, unter dem das Dokumentenverzeichnis der zu indizierenden Website liegt. Soll die Datenbank, wie in *WAISDIR* angegeben, in einem Nebenverzeichnis des Webservers liegen, benötigt das Skript natürlich die entsprechenden Rechte, sonst tut's auch jedes andere Verzeichnis. *waisindex* ackert dabei rekursiv die gesamte Website durch, dieser Vorgang kann, entsprechend der zu bearbeiteten Dokumentenzahl, etwas dauern. Der eigentliche Server-Prozeß läßt sich über folgende Kommandozeile starten:

```
/usr/local/bin/waisserver -d
/usr/local/etc/httpd/wais -p 4711 .
```

Die angegebene Port-Nummer muß - klarerweise - mit der im CGI-Script *search.pl* angegebenen übereinstimmen, das über die *-d*-Option spezifizierte Verzeichnis entspricht dem *waisindex* vorher mitgeteilten. Damit der WAIS-Server zukünftig gleich beim Booten des Rechners startet, empfiehlt es sich, folgende Zeilen in */etc/rc.d/rc.local* zu packen:

```
echo "Starting WAIS Server"
/usr/local/bin/waisserver -d
/usr/local/etc/httpd/wais \
   -p 4711 -e /usr/var/log.wais &
```

WAIS.PM INSTALLIEREN

Die Perl-Schnittstelle *Wais.pm* zu installieren, ist etwas haarig, aber machbar. Zunächst geht alles wie gewohnt: Ans CPAN, wahlweise

```
ftp://ftp.leo.org/pub/comp/programming/
      languages/perl/CPAN/
ftp://ftp.rz.ruhr-uni-bochum.de/pub/CPAN/
ftp://ftp.uni-hamburg.de/pub/soft/lang/perl/CPAN/
```

angedockt, findet sich unter *modules/by-module/Wais* die Distribution

```
Wais-2.304.tar.gz
```

Diese entpackt sich wie gewohnt - und zwar am geschicktesten im selben Verzeichnis wie vorher die *freeWAIS-sf*-Distribution - mit

```
tar zxfv Wais-2.304.tar.gz
```

Doch vor dem *make* sind einige wilde Aktionen mit der *freeWAIS-sf*-Distribution notwendig: So benötigt *Wais.pm* die Include-Datei *wais.h* sowie die Bibliothek *libwais.a*. Beide Dateien entstehen unter Mitwirkung des *freeWAIS-sf*-Source-Codes.
Stehen, wie vorgeschlagen, *freeWAIS-sf-2.1.2/* und *Wais-2.304/* im gleichen Verzeichnis, geht das so:

```
cd freeWAIS-sf-2.1.2/ir
perl ../../Wais-2.304/mkinc -I../ctype ui.h cutil.h
irext.h \
  irfiles.h irsearch.h irtfiles.h weight.h \
  docid.h >/tmp/wais.h

cd ../../Wais-2.304
mkdir tmp
cd tmp
ar x ../../freeWAIS-sf-2.1.2/ir/libwais.a
ar x ../../freeWAIS-sf-2.1.2/regexp/libregexp.a
ar x ../../freeWAIS-sf-2.1.2/lib/libftw.a
cp ../../freeWAIS-sf-2.1.2/ctype/ctype.o .
ar rc libwais.a *.o
ranlib libwais.a
cp libwais.a /tmp/libwais.a
cd ..
```

Dann müssen *wais.h* und *libwais.a*, die noch unter */tmp* liegen, dorthin kopiert werden, wo der Compiler sie findet, also z.B. (als *root*)

```
cp /tmp/wais.h /usr/local/include
cp /tmp/libwais.a /usr/local/lib
```

Dann ist *Wais-2.304* endlich startbereit:

```
cd Wais-2.304
perl Makefile.PL
make
```

Der *make*-Lauf wirft eine Reihe von unkritischen Warnings auf, läuft aber erfolgreich. Schließlich plaziert ein unter *root* aufgerufenes

```
make install
```

das Modul *Wais.pm* in den Perl-Modul-Pfad. Es gab schon einfachere Installationen!

LISTING *SEARCH.PL*

```
1  #!/usr/bin/perl -wT
2  ##############################################################################
3  # search.pl - Stichwortsuche in einer WAIS Datenbank
4  ##############################################################################
5
6  use CGI qw/:form :html param header/;
7  use Wais;
8
9  ##############################################################################
10 # Konfiguration
11 ##############################################################################
12 $htdocs_path = "/usr/local/etc/httpd/htdocs";
13 $dbname = "website";
14 $hostname     = "m1";
15 $port         = 4711;
16 ##############################################################################
17
18 print_form();                    # Überschrift und Suchformular ausgeben
19
20 exit 0 unless param("query"); # Ohne spezifizierten Query ist hier Schluß
21
22                                  # Query ist angegeben, WAIS befragen
23 $result = Wais::Search({'query' => param("query"), 'database' => $dbname,
24                         'host'   => $hostname,      'port'     => $port});
25
26 @results = $result->header(); # Ergebnisse aufbereiten
27
28 if($#results < 0) {             # Fehler aufgetreten?
29     print pre(font({color=>"red"}, "Error - WAIS server down?\n"));
30     return;
31 }
32
33 $nof_hits = $#results + 1;     # Anzahl der Ergebnisse
34
35 if($nof_hits == 1) {           # Ein Ergebnis mit Score 0 => Kein Treffer
36     my ($tag, $score) = @{$results[0]};
37     $nof_hits = 0 unless $score;
38 }
39
40 print hr, i($nof_hits, " documents found\n"); # Anzahl Treffer
41
42 for(@results) {        # Über Ergebnisse iterieren
43
44     my ($tag, $score, $lines, $bytes, $headline, $types, $docid) = @$_;
45
46     next unless $score;        # Bogus-Ergebnis vergessen
47
48                                  # WAIS-Pfad in URL-Pfad umwandeln
49     my ($file, $path) = split(' ', $headline);
50     $path =~ s,^$htdocs_path/,,g;         # File-System-prefix wegwerfen
51                                  # URL in Tabelle hängen
52     $rows .= TR(td( a({href=>"http://$hostname/$path$file"},
53                       "$path$file")));
54 }
55
56 print table($rows);            # Tabelle ausgeben
57
58 ##############################################################################
59 sub print_form {
60 ##############################################################################
61
62     print header,
63           start_html(-title    => 'Website Search',
64                      -BGCOLOR => 'white'),
65           center(h1("Search the Website")),
66           center(table(TR(
67               td(
68                   p("Enter Query:")),
69               td(
70                   start_form,
71                   textfield(-name  => 'query',
72                             -value => (param('query') || ""),
73                             -size  => 40),
74                   end_form,
75           )))),
76           end_html;
77 }
```

DER AUTOR

Michael Schilli arbeitet als Web-Engineer bei AOL Productions, San Mateo. Er ist Autor des bei Addison-Wesley erschienenen Buches *Effektives Programmieren mit Perl 5* und unter *mschilli1@aol.com* zu erreichen. Seine Homepage findet man unter *http://mission.base.com/mschilli*.