öS 70,– sfr 8,50 **DM 8,50** H 10554

Januar 1999

1

Supercomputing:
512 x Linux im Cluster

c't

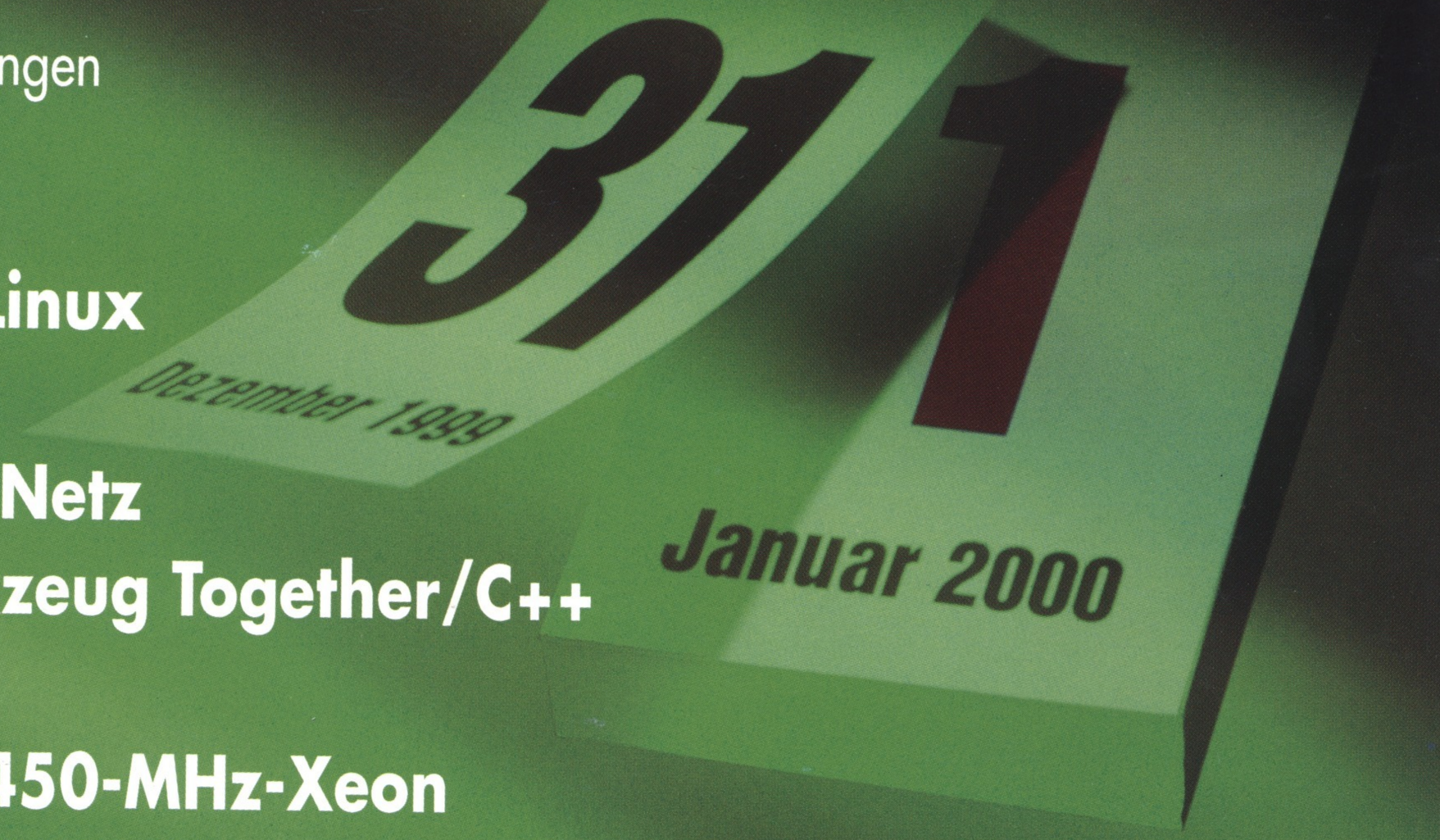
MAGAZIN FÜR PROFESSIONELLE INFORMATIONSTECHNIK

Mit Stellenmarkt

Windows 95/NT, Linux, Solaris, AIX, HP-UX im Praxistest:

Das Jahr-2000-Problem

Grundlagen und Lösungen



Getestet:

Empress 8 für Linux
Diascanner
CD-Brenner im Netz
OO-CASE-Werkzeug Together/C++

High-End 3D-Grafik:

HP Kayak mit 450-MHz-Xeon
IBM 43P 260 mit 200-MHz-Power 3

WWW für Unternehmensanwendungen:

Application Server

Sichere Firmennetze:

IPsec im Internet

Squid konfigurieren:

Schnelles Web durch Proxies

Vergleich:

5 TFT-Displays

ab 1280 × 1024 Auflösung

4 399148 508506 01

c't Magazin für professionelle Informationstechnik 1/99
http://www.heise.de/ix/

Anzeigen schalten im Web

Rotationsprinzip

Michael Schilli

Anwender grummeln, doch der Rubel rollt: Genau wie sich das Privatfernsehen durch Werbung finanziert, bestreiten sogenannte Portal-Sites ihren Umsatz mit Werbebildchen. Der hier vorgestellte Ad-Server bahnt auch kleineren Web-Unternehmen den Weg in den totalen Kommerz: Er schaltet Anzeigen.

Ein einzelnes Werbebildchen präsentiert sich auf einer HTML-Seite folgendermaßen:

```
<A HREF="http://werber.de">
<IMG SRC="http://werber.de/banners/pic.gif">
</A>
```

Dies läßt im Browser des Benutzers die Abbildung *http://werber.de/banners/pic.gif* erscheinen, die Anzeige des Anbieters. Klickt der Anwender darauf, springt der Browser zur Web-Seite des Werbenden, nach http://werber.de: ein potentieller Kunde mehr.

Für einen florierenden Handel mit mehreren Anzeigenkunden genügt diese statische Implementierung freilich nicht – bei eventuell Hunderten von Werbekunden muß ein Anzeigenwechsler her, der nach einem festgelegten Schema bei jedem neuen Aufruf der Web-Seite ein anderes Inserat schaltet und so eine feste Menge von Anzeigen über eine Vielzahl von Benutzern verstreut.

HTML sträubt sich

Ein mit einem CGI-Script verknüpftes Image-Tag

```
<IMG SRC="/cgi-bin/getimage.pl">
```

ließe in blankem HTML nun zwar wechselnde Bilder zu, jedoch sollen die einzelnen Anzeigen zu unterschiedlichen URLs verzweigen – und diese Dynamik läßt sich in HTML leider nicht formulieren. Die vollständige Dynamisierung

```
<A HREF="http://werber.de/cgi-bin/geturl.pl">
<IMG SRC="http://werber.de/cgi-bin/getimg.pl">
</A>
```

stellt die CGI-Scripts *geturl.pl* und *getimg.pl* vor das Problem, daß *geturl.pl* auf Mausklick zu genau dem URL verzweigen muß, dessen Werbebildchen zuvor *getimg.pl* aussandte. Mittels Cookies könnte der Server zwar *getimg.pl*-Benutzer in *geturl.pl* 'wiedererkennen' und entsprechend bedienen. Diese Lösung krankt jedoch vor allem daran, daß viele Browser solche Image-Tags, die den gleichen Namen tragen (hier den eines CGI-Scripts), gar nicht mehr anfordern, sondern einfach eine Kopie aus dem Cache verwenden.

Da beißt die Maus keinen Faden ab – es müssen dynamische Seiten her. Jede Seite durch ein CGI-Script auszuwechseln, ist in den seltensten Fällen tragbar, also wie wär's mit den fast in Vergessenheit geratenen Server Side Includes (SSI)?

Sind diese eingeschaltet (siehe 'Server Side Includes aktivieren'), genügt es, eine *.html*-Datei in *.shtml* umzubenennen, und der Eintrag

```
<!--exec cgi=/cgi-bin/adserver.pl -->
```

wirbelt mit Hilfe des Perl-Scripts *adserver.pl* ein ausgewähltes Werbebanner hervor, zusammen mit dem richtigen Link, der den Kunden auf die werbende Seite beamen wird. *adserver.pl* arbeitet mit einer Bannersequenzdatei (*adserver.seq*) der Art

```
# Bannersequenzdatei adserver.seq
# Format: IMGURL CLICKURL
/images/apache.gif                  http://www.apache.org
http://localhost/images/aol.gif     http://www.aol.com
http://perlmeister.com/images/perlmeister.jpg
                                    http://perlmeister.com
```

LISTING ADSERVER.PL

```
 1 #!/usr/bin/perl -w
 2 ################################################
 3 # Perl-Script zum rotierenden Ausliefern von
 4 # Werbebildern und -Links
 5 # 1998, schilli@perlmeister.com
 6 ################################################
 7
 8 use CGI qw(:standard);
 9                     # Send errors to Browser
10 use CGI::Carp 'fatalsToBrowser';
11
12                     # Ad banner sequence file
13 $BANNER_SEQ_FILE   = "adserver.seq";
14                     # Persistent counter
15 $BANNER_COUNT_FILE = "adserver.cnt";
16
17                     # Open banner sequence file
18 open(SEQ, "<$BANNER_SEQ_FILE") ||
19     die "Cannot open $BANNER_SEQ_FILE";
20
21 while(<SEQ>) {       # Read it and fill @banners
22     chomp;           # Remove Newlines
23     s/^\s*#.*//g;    # Remove comments
24     next if /^\s*$/; # Ignore blank lines
25                      # Extract two fields
26     my ($img, $url) = split(' ', $_);
27                      # Two fields, really?
28     die "Invalid line in $BANNER_SEQ_FILE: $_"
29         unless defined $url;
30     push(@banners, [$img, $url]);
31 }
32
33 close(SEQ);
34
35     # Get and increase persistent counter
36 my $counter = (inc_counter() % ($#banners + 1));
37
38     # Spit out CGI header and anchored image
39 print header(),
40       a({href => $banners[$counter]->[1] },
41         img({src => $banners[$counter]->[0]}));
42
43
44 ################################################
45 sub inc_counter {    # Get and increase
46                      # permanent counter
47 ################################################
48     my $counter = 0;
49
50             # Counter file exists?
51     if(-f $BANNER_COUNT_FILE) {
52
53             # Open it exclusively
54         open(COUNT, "+<$BANNER_COUNT_FILE") ||
55             die "Cannot r/w $BANNER_COUNT_FILE";
56
57             # Lock it
58         flock(COUNT, 2);
59
60             # Read out value
61         seek(COUNT, 0, 0);
62         $counter = <COUNT>;
63         chomp($counter);
64
65             # Truncate file
66         seek(COUNT, 0, 0);
67         truncate(COUNT, 0);
68
69     } else {
70             # File doesn't exist - create it
71         open(COUNT, ">$BANNER_COUNT_FILE") ||
72             die "Cannot init $BANNER_COUNT_FILE";
73     }
74
75         # Write new value to counter file
76     print COUNT $counter+1, "\n";
77
78         # Close, implicitly release lock
79     close(COUNT);
80
81         # Return old counter value
82     return $counter;
83 }
```

Installation

Um den Werbeserver zu aktivieren, müssen die Pfade zu den Dateien *adserver.seq* und *adserver.cnt* (Zeilen 13 und 15) in *adserver.pl* angepaßt und *adserver.pl* schließlich ins CGI-Verzeichnis des anzeigenden Web-Servers verschoben werden. Die Datei *adserver.seq* muß zeilenweise Wertepaare enthalten, die, durch Leerzeichen getrennt,

– einen relativen oder absoluten URL auf das anzuzeigende Bild und
– die anzuspringende URL enthalten.

Mit # beginnende Kommentarzeilen und Leerzeilen sind erlaubt. Produziert ein anschließender Aufruf von *http://localhost/cgi-bin/adserver.pl* die Fehlermeldung

```
Cannot init adserver.cnt at ... line 69.
```

konnte *adserver.pl* die Datei für den permanenten Zähler nicht anlegen, und der Pfad ist entweder auf ein beschreibbares Verzeichnis anzupassen oder eine leere Datei mit Schreibrechten unter dem angegebenen Pfad (ohne Pfadangabe: CGI-Verzeichnis) anzulegen. Erscheint andererseits

```
Cannot open adserver.seq at ... line 18.
```

wurde die Bannersequenzdatei nicht gefunden, und der Pfad in Zeile 13 ist zu korrigieren. Läuft alles korrekt und der Web-Server für SSI konfiguriert (Kasten SSI), reicht ein Eintrag vom Format

```
<!-exec cgi=/cgi-bin/adserver.pl ->
```

in einer *.shtml*-Datei, um bei jedem Web-Request das Script auszulösen und das Werbebanner einzupassen.

die es sequentiell abarbeitet und so dem ersten anfragenden Benutzer das Banner

```
<A HREF="http://www.apache.org">
<IMG SRC="/images/apache.gif"></A>
```

liefert, um bei der nächsten Anfrage mit der folgenden Image/URL-Kombination fortzufahren. Wie aus dem Beispiel ersichtlich, darf der URL des Bildchens relativ angegeben sein, wenn es sich auf dem anzeigenden Rechner befindet. Bilder von fremden Rechnern – Erlaubnis des dortigen Administrators vorausgesetzt – werden ebenfalls korrekt verarbeitet. Am Ende der Datei angelangt, startet *adserver.pl* von neuem am Anfang; dafür sorgt ein Zähler, der in der Datei *adserver.cnt* das kurzlebige CGI-Script *adserver.pl* überdauert. *adserver.cnt* enthält nur eine natürliche Zahl, die das Script bei jedem Aufruf erhöht. Die Modulo-Operation in Zeile 36 garantiert, daß das Script nur gültige Bannereinträge wählt, auch wenn der Zähler über die Anzahl der verfügbaren Werbebanner hinausschießt. Sollen bestimmte Anzeigen häufiger erscheinen als andere, trägt man sie mehrfach in *adserver.seq* ein.

Server Side Includes aktivieren

Beim Apache Server (getestete Version 1.3.1) aktivieren die beiden Einträge in *srm.conf*

```
AddType text/html .shtml
AddHandler server-parsed .shtml
```

das *mod_include*-Modul für *.shtml*-Dateien und ein Eintrag in *access.conf* wie

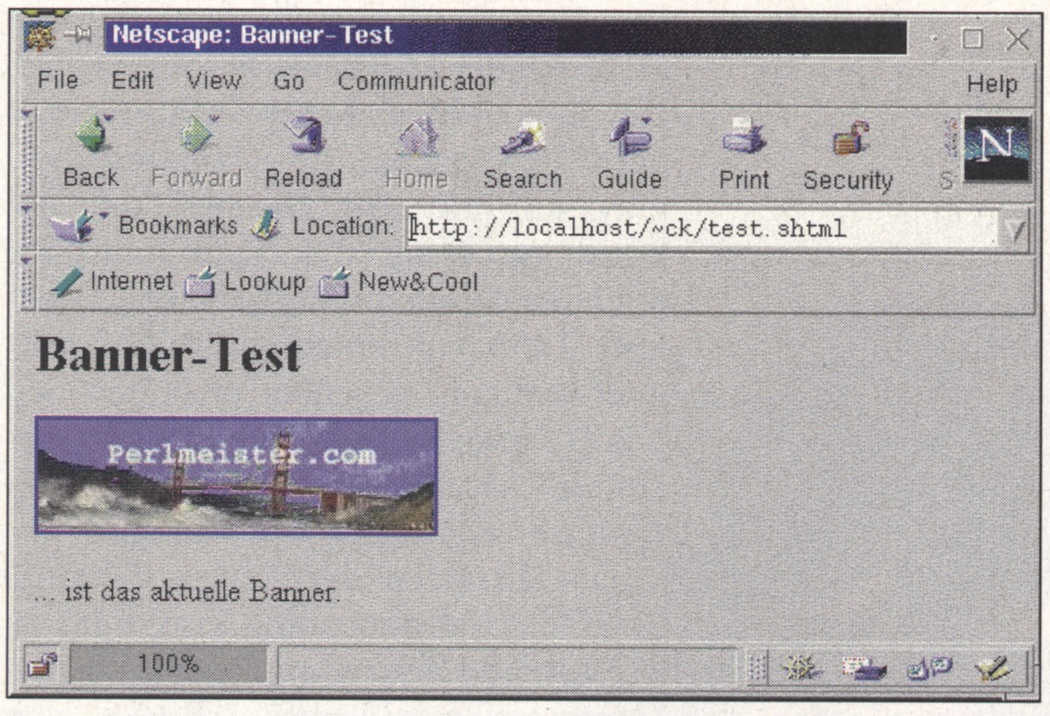
```
<Directory /services/http/share/apache/htdocs>
...
Options +Includes
...
</Directory>
```

veranlaßt den Server, Dateien im angegebenen Verzeichnis nicht einfach zum Client zu pumpen, sondern vorher die Server-Side-Includes auszuwerten und gegebenenfalls dynamisch Text einzusetzen.

Haltbarer Zähler per Datei

Als typisches CGI-Script verwendet *adserver.pl* Lincoln Steins CGI-Modul, das neueren Perl-Versionen standardmäßig beiliegt. Zeile 10 läßt es statt des verwirrenden Internal Server Error eine aufschlußreiche Fehlermeldung anzeigen, falls es nicht mehr weiter weiß und auf eine *die*-Anweisung läuft. Die Zeilen 18 bis 33 lesen die Werbebannerkonfigurationsdatei aus und legen die gefundenen Image-URL-Wertepaare im Array *@banners* ab. Zeile 36 ruft die Funktion *inc_counter* auf, die den gegenwärtigen Wert des persistenten Zählers ausgibt, und ihn gleichzeitig für den nächsten Aufruf hochzählt. Die Zeilen 39 bis 41 formatieren dann die Ausgabe in HTML und geben sie aus.

inc_counter() öffnet die Datei, deren Namen die globale Variablen *$BANNER_COUNT_FILE* enthält, liest einen eventuell dort vorhandenen Wert aus, erhöht ihn um eins und schreibt den neuen Wert zurück. Der Rückgabewert der Funktion ist der



Ein Klick auf 'Reload' zaubert das nächste Bild hervor.

LISTING TEST.SHTML

```
<HTML>
<HEAD> <TITLE>Banner-Test</TITLE> </HEAD>

<BODY>
<h1>Banner-Test</h1>

<p>
<!--#exec cgi="/cgi-bin/adserver.pl" -->

<p>
... ist das aktuelle Banner.

</BODY>
</HTML>
```

***test.shtml* benutzt das Perl-Script *adserver.pl* zur abwechselnden Darstellung von Werbebannern.**

alte Wert des Zählers. Da CGI-Scripts quasi-parallel auf die Datei zugreifen, sichert *inc_counter()* die Datei mit dem *flock*-Befehl und garantiert damit, daß kein weiterer Prozeß oder Thread zur selben Zeit darauf herumorgelt.

Listing *test.shtml* zeigt eine Beispielseite, die den Server dazu veranlaßt, vor der Auslieferung das aktuelle Werbebanner in den Text einzupassen. Abbildung 1 zeigt das Ergebnis. Findet eine Anzeige Interesse beim Kunden, klickt er also auf das entsprechende Werbebanner, läuft dieser Vorgang freilich völlig am Server des Unternehmens vorbei, das die Anzeige präsentiert: Der Browser verabschiedet sich grußlos und springt die neue Seite an. So freut sich zwar der Werbende über einen potentiellen neuen Kunden, doch auch den Betreiber der Portal-Site interessiert, wie viele Kaufwillige einen Ausflug zu seinen Werbekunden wagen: Informationen über den Erfolg einer Anzeige werden hoch gehandelt.

Die Fortsetzung im nächsten Heft präsentiert einen sogenannten Click-Through-Tracker, der nicht nur überwacht, wer wohin klickt, sondern diese wertvollen Informationen auch noch grafisch ansprechend aufbearbeitet. (ck)

MICHAEL SCHILLI

arbeitet als Web-Engineer für America Online Inc., San Mateo. Er ist Autor des bei Addison-Wesley erschienenen Buches 'GoTo Perl 5'.