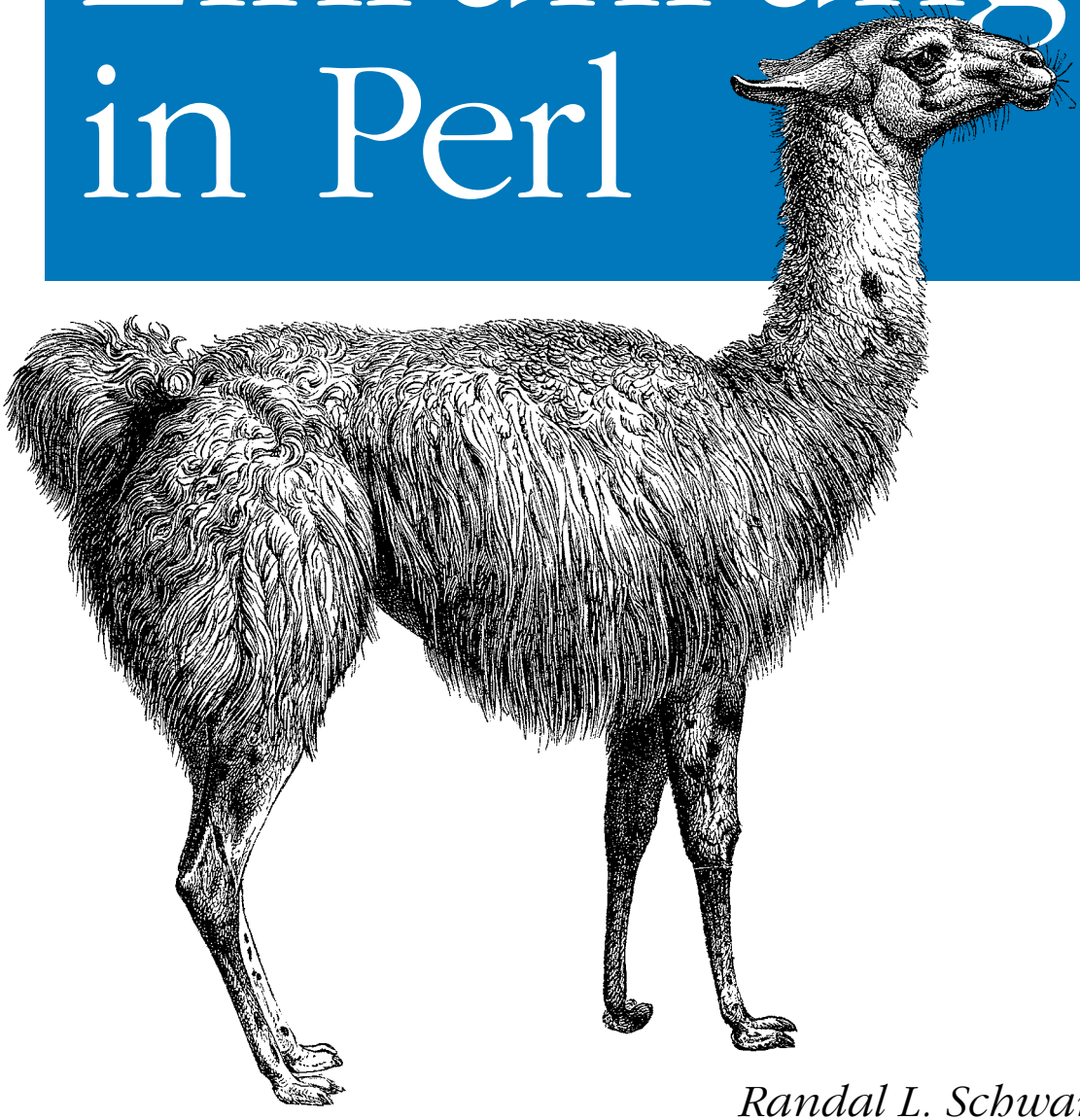
Behandelt Perl 5.14

Deutsche Ausgabe der 6. Auflage

Einführung in Perl



Randal L. Schwartz,
brian d foy & Tom Phoenix

Deutsche Übersetzung von Jørgen W. Lang,
aktualisiert von Eike Nitz

O'REILLY®

Einführung in Perl

Randal L. Schwartz, Tom Phoenix & brian d foy

Deutsche Übersetzung von Jørgen W. Lang

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

Die Informationen in diesem Buch wurden mit größter Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Verlag, Autoren und Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für eventuell verbliebene Fehler und deren Folgen.
Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen. Der Verlag richtet sich im wesentlichen nach den Schreibweisen der Hersteller. Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Alle Rechte vorbehalten einschließlich der Vervielfältigung, Übersetzung, Mikroverfilmung sowie Einspeicherung und Verarbeitung in elektronischen Systemen.
Kommentare und Fragen können Sie gerne an uns richten:

O'Reilly Verlag
Balthasarstr. 81
50670 Köln
E-Mail: kommentar@oreilly.de

Copyright:
© 2012 by O'Reilly Verlag GmbH & Co. KG
6. Auflage 2012

Die Darstellung eines Lamas im Zusammenhang mit dem Thema Perl ist ein Warenzeichen von O'Reilly Media, Inc.

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

Lektorat: Volker Bombien, Köln
Korrektorat: Astrid Sander, Köln
Satz: III-Satz, Husby; www.drei-satz.de
Umschlaggestaltung: Michael Oreal, Köln
Produktion: Andrea Miß, Köln
Belichtung, Druck und buchbinderische Verarbeitung:
Druckerei Kösel; Krugzell, www.koeselbuch.de

ISBN 978-3-86899-145-1

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Inhalt

Vorwort

Willkommen zur 6. Auflage von *Einführung in Perl*, aktualisiert auf Perl 5.14. Sie können dieses Buch aber auch problemlos benutzen, wenn Sie noch mit Perl 5.8 arbeiten (wobei seine Veröffentlichung mittlerweile schon so lange her ist, dass Sie vielleicht über ein Update nachdenken sollten).

Wenn Sie nach der besten Möglichkeit suchen, die nächsten 30 bis 45 Stunden mit der Programmiersprache Perl zu verbringen, hat Ihre Suche hier ein Ende. Auf den folgenden Seiten finden Sie eine sorgfältig abgestimmte Einleitung in die Sprache, die als das »Arbeitspferd des Internet« sowie als die Sprache der Wahl bei Systemadministratoren, Web-Hackern und Gelegenheitsprogrammierern auf der ganzen Welt gilt.

Wir können Ihnen natürlich nicht den gesamten Umfang von Perl in ein paar Stunden beibringen. Bücher, die Ihnen das versprechen, übertreiben ein bisschen. Stattdessen haben wir sorgfältig einen nützlichen Querschnitt von Perl für Sie ausgewählt. Dieser reicht aus, um Anwendungen mit einer Länge von einer bis zu 128 Zeilen zu schreiben, die ungefähr 90% aller Programme ausmachen. Wenn Sie dieses Buch durchgearbeitet haben, können Sie sich das Alpaka-Buch besorgen, das dort anschließt, wo das Ihnen hier vorliegende Lama-Buch aufhört. Wir haben außerdem eine Reihe von weiterführenden Hinweisen mit aufgenommen, die Ihnen weiterhelfen sollen, wenn Sie sich mit spezielleren Perl-Themen befassen wollen.

Jedes Kapitel ist kurz genug, um in einer bis zwei Stunden gelesen zu werden, und endet mit einer Reihe von Übungen, die Ihnen helfen sollen, das Gelernte anzuwenden. Die Antworten haben wir für Sie zur Überprüfung in Anhang A zusammengestellt. Dieses Buch ist also ideal für Perl-Einführungskurse und -seminare geeignet. Wir wissen dies aus erster Hand, da das Material für dieses Buch fast Wort für Wort aus dem Konzept für unseren Perl-Einführungskurs stammt, den mittlerweile Tausende von Kursteilnehmern in aller Welt absolviert haben. Dennoch ist dieses Buch natürlich auch für das Selbststudium geeignet.

Perl wird oft als der »Werkzeugkasten für Unix« bezeichnet, aber Sie müssen kein Unix-Guru sein, nicht einmal Unix-Benutzer, um mit diesem Buch Perl lernen zu können.

Sofern es nicht extra vermerkt ist, gilt alles, was wir sagen, auch für die Windows-Version ActivePerl von ActiveState und für andere moderne Implementierungen von Perl.

Auch wenn Sie zu Beginn der Lektüre dieses Buchs keinerlei Ahnung von Perl haben müssen, ist es empfehlenswert, sich bereits mit grundsätzlichen Konzepten des Programmierens auszukennen. Hierzu zählen etwa Variablen, Schleifen, Unterroutinen, Arrays und das ganz wichtige »eine Quellcode-Datei mit Ihrem Lieblings-Texteditor bearbeiten«. Diese Konzepte setzen wir voraus. Auch wenn es uns freut zu hören, dass der eine oder andere mit diesem Buch Perl als erste Programmiersprache gelernt hat, können wir nicht allen Lesern versprechen, dass sie das ebenfalls hinbekommen.

Typografische Konventionen

Die folgenden typografischen Konventionen werden in diesem Buch verwendet:

`Nichtproportionalschrift`
: wird für Namen von Methoden, Funktionen, Variablen und Attributen sowie für Codebeispiele verwendet.

`Nichtproportionalschrift fett`
: wird verwendet, wenn User-Eingaben dargestellt werden sollen.

`Nichtproportionalschrift kursiv`
: steht für austauschbare Codeelemente (zum Beispiel *`Dateiname`*, wenn der tatsächliche Dateiname eingesetzt werden soll).

Kursivschrift
: wird benutzt für Dateinamen, URLs, Rechnernamen, wichtige Begriffe bei ihrer ersten Erwähnung und zur Hervorhebung von Textteilen.

Fußnoten
: enthalten zusätzliche Anmerkungen, die Sie beim ersten Lesen dieses Buchs *nicht mitlesen* sollten (sondern vielleicht beim zweiten oder dritten Mal). Manchmal wird im Text bewusst ein wenig die Unwahrheit gesagt, um die Erklärungen zu vereinfachen, und eine Fußnote stellt dann alles wieder richtig. Oft enthalten Fußnoten auch Hinweise auf weiterführendes Material, das in diesem Buch gar nicht weiter behandelt wird.

[2], [5] usw.
: am Anfang der Texte zu den Übungen stellen grobe Schätzungen dar, wie viele Minuten Sie voraussichtlich mit der jeweiligen Übung verbringen werden.

Verwendung der Codebeispiele

Dieses Buch soll Ihnen dabei helfen, Ihre Arbeit zu erledigen. Sie können den Code aus diesem Buch benutzen und an Ihre eigenen Bedürfnisse anpassen. Wir empfehlen Ihnen

allerdings, ihn nicht von Hand zu kopieren, sondern unter *http://www.learning-perl.com* herunterzuladen.

Den Code, den wir hier zeigen, dürfen Sie im Allgemeinen in Ihren Programmen und in Ihrer Dokumentation verwenden. Sie brauchen uns nicht um Genehmigung bitten, sofern Sie nicht große Teile des Codes reproduzieren. Wenn Sie zum Beispiel ein Programm schreiben, das mehrere Codestücke aus diesem Buch wiederverwendet, brauchen Sie uns nicht um Erlaubnis zu bitten, wenn Sie jedoch eine CD mit Codebeispielen aus O'Reilly-Büchern verkaufen oder vertreiben wollen, müssen Sie eine Genehmigung einholen. Eine Frage mit einem Zitat aus diesem Buch und seinen Codebeispielen zu beantworten erfordert keine Erlaubnis, aber es ist ohne Genehmigung nicht gestattet, große Teile unseres Textes oder Codes in eine eigene Produktdokumentation aufzunehmen.

Wir freuen uns über eine Quellenangabe, verlangen sie aber nicht unbedingt. Zu einer Quellenangabe gehören normalerweise Autor, Titel, Verlag und ISBN, zum Beispiel so: »Randal L. Schwartz, Tom Phoenix & brian d foy: *Einführung in Perl*, 6. Auflage, O'Reilly Verlag 2011, ISBN 978-3-868 99-145-1«.

Die Geschichte dieses Buchs

Für die Neugierigen unter Ihnen folgt nun Randals Version der Entstehungsgeschichte dieses Buchs.

Nachdem Larry Wall und ich 1991 mit der Arbeit an der 1. Auflage von *Programming perl* fertig waren, wurde ich von der Firma Taos Mountain Software im Silicon Valley angesprochen, einen Perl-Kurs zu entwickeln. Hierzu gehörte auch das erste Dutzend Kurse, um Ausbilder dieser Firma auszubilden, diesen Kurs zu halten. Ich schrieb also diesen Kurs und gab ihn, wie vereinbart, an die Firma[1] weiter.

Als ich den Kurs das dritte oder vierte Mal hielt (Ende 1991), kam jemand auf mich zu und sagte: »Weißt du, ich finde ja *Programming Perl* wirklich gut, aber die Art, wie das Material hier im Kurs aufbereitet wird, ist viel leichter nachzuvollziehen. Du solltest ein Buch schreiben, das wie dieser Kurs aufgebaut ist.« Das schien mir eine gute Idee, und so begann ich, darüber nachzudenken.

Ich schrieb an Tim O'Reilly und schlug ihm ein Buch vor, das sich in den Grundzügen an dem Kurs orientierte, den ich für Taos entwickelt hatte. Allerdings habe ich einige Kapitel, nach Beobachtungen im Kursraum, etwas umstrukturiert und geändert. Ich glaube, schneller ist noch nie ein Buchvorschlag von mir angenommen worden: Bereits 15 Minuten später erhielt ich eine Nachricht von Tim: »Wir haben schon darauf gewartet, dass

1 In dem Vertrag behielt ich mir die Rechte an den Übungen vor, in der Hoffnung, diese eines Tages woanders, etwa in den Zeitschriftenkolumnen, die ich zu der Zeit schrieb, wiederverwenden zu können. Die Übungen sind das Einzige, das aus dem Taos-Kurs in diesem Buch verwendet wird.

du ein zweites Buch einreichst – *Programming perl* verkauft sich wie verrückt.« Die folgenden 18 Monate verbrachte ich dann damit, die 1. Auflage von *Learning Perl* fertig zu stellen.

Während dieser Zeit begann ich, eine Gelegenheit zu sehen, Perl auch außerhalb des Silicon Valley[2] zu lehren. Also entwickelte ich einen Kurs, der auf dem Text basierte, den ich für *Learning Perl* geschrieben hatte. Ich gab ein Dutzend Kurse für verschiedene Kunden (unter anderem auch für meinen Hauptauftraggeber, Intel Oregon) und benutzte die Rückmeldungen, die ich bekam, um der Rohfassung des Buchs den letzten Schliff zu geben.

Die 1. Auflage kam am 1. November 1993[3] in die Läden und wurde zu einem überwältigenden Erfolg, dessen Verkaufszahlen oft sogar die von *Programming perl* übertrafen.

Auf der Rückseite der 1. Auflage stand zu lesen: »Geschrieben von einem der führenden Perl-Trainer«. Das wurde schnell zu einer sich selbst erfüllenden Prophezeiung. Innerhalb von wenigen Monaten bekam ich E-Mails von Leuten überall aus den USA, die mich darum baten, für ihre Firmen Kurse zu geben. Während der folgenden sieben Jahre entwickelte sich meine Firma zum weltweiten Marktführer für Perl-Training vor Ort, und ich sammelte eine Million Vielflieger-Meilen. Es kam auch nicht ungelegen, dass es während dieser Zeit so richtig mit dem Web losging. Bei Webmastern und -meisterinnen wurde Perl die bevorzugte Sprache für Content-Management, Benutzerinteraktion mit CGI und die Pflege von Sites.

Zwei Jahre arbeitete ich bei Stonehenge ziemlich eng mit Tom Phoenix zusammen. Ich gab ihm Freiraum, mit dem »Lama«-Kurs herumzuexperimentieren, einige Dinge umzustellen und anders aufzuteilen. Als wir mit dem fertig waren, was wir als die beste große Revision dieses Kurses ansahen, rief ich Tim O'Reilly an und sagte: »Zeit für ein neues Buch!« Das Ergebnis war die 3. Auflage dieses Buchs.

Zwei Jahre nach der 3. Auflage des Lama-Buchs beschlossen Tom und ich, dass es an der Zeit sei, auch unseren Fortgeschrittenen-Kurs als Buch herauszubringen. Und so entstand das Alpaka-Buch, *Einführung in Perl-Objekte, Referenzen & Module*, das 2004 erschien (*www.oreilly.de/catalog/lrnperlormger/*).

Zu dieser Zeit kam unser Kollege brian d foy vom Einsatz im Golfkonflikt zurück; ihm war aufgefallen, dass beide Bücher ein wenig Anpassung gebrauchen konnten, um die veränderten Bedürfnisse der Leser bzw. Lernenden besser zu berücksichtigen. Also schlug er O'Reilly vor, Lama und Alpaka noch ein (hoffentlich) letztes Mal vor Perl 6 in einer neuen Auflage herauszugeben. Die vorliegende Auflage von *Einführung in Perl* spie-

2 Mein Vertrag mit Taos enthielt eine Klausel, nach der ich meinem Auftraggeber keine Konkurrenz machen durfte. Ich musste mich also vom Silicon Valley fern halten, was ich auch viele Jahre lang respektiert habe.

3 An diesen Tag erinnere ich mich gut – es war der Tag, an dem man mich zu Hause wegen Computeraktivitäten im Zusammenhang mit meinem Intel-Vertrag verhaftete. Es wurde eine Reihe von Anklagen erhoben, für die ich später auch verurteilt wurde. Details finden Sie unter *http://www.lightlink.com/fors/*.

gelt diese Veränderungen wider. brian war dabei der Hauptautor und hat großartige Arbeit geleistet, das Team aus mehreren Autoren zu organisieren.

Am 18. Dezember 2007 veröffentlichten die Perl-Portierer Perl 5.10, eine bedeutsame neue Version mit zahlreichen neuen Features. Bei der Vorgängerversion 5.8 war es hauptsächlich um Unicode-Unterstützung gegangen. Die neue Version, die auf der stabilen 5.8 aufbaute, hatte ganz neue Features zu bieten, von denen einige schon aus der Entwicklung des noch nicht veröffentlichten Perl 6 stammten. Ein paar dieser Features, zum Beispiel benannte Captures in regulären Ausdrücken, sind in ihrer Funktionsweise den alten haushoch überlegen und dadurch ideal für Perl-Anfänger. Wir hatten nie eine fünfte Auflage dieses Buches geplant, aber Perl 5.10 hat so viele neue Features zu bieten, dass wir einfach nicht widerstehen konnten.

Seitdem ist Perl fortlaufend verbessert worden und hat einen regelmäßigen Release-Zyklus aufzuweisen. Wir hatten keine Gelegenheit dazu, das Buch für Perl 5.12 zu überarbeiten, weil die Entwicklung zu schnell voranging. Wir freuen uns, Ihnen die vorliegende Überarbeitung für Perl 5.14 präsentieren zu können, und können selbst kaum fassen, dass es jetzt schon die 6. Auflage gibt.

Veränderungen in dieser Auflage

Ihnen werden vermutlich einige Unterschiede zur letzten Auflage auffallen. Der Text wurde auf Perl 5.14 aktualisiert. Einiger Code in diesem Buch funktioniert nur mit dieser Version. Wir weisen im Text darauf hin, wenn wir über Perl 5.14-Features sprechen, und die entsprechenden Codeabschnitte sind mit einem use-Statement markiert, damit Sie auch die richtige Version verwenden:

```
use 5.014; # dieses Skript setzt Perl 5.14 oder höher voraus
```

Wenn Sie nicht diesen Hinweis vorfinden, sollte der Code bis hinunter zu Perl 5.8 lauffähig sein. Um in Erfahrung zu bringen, mit welcher Perl-Version Sie arbeiten, verwenden Sie folgenden Befehl:

```
$ perl -v
```

Hier ist eine Liste mit einigen der neuen Features in Perl 5.14, die behandelt werden. Wo es angebracht ist, erklären wir aber auch, wie man dasselbe in älteren Versionen bewerkstelligt.

- Wir geben an geeigneten Stellen auch Unicode-Beispiele und -Features mit an. Für den Fall, dass Sie noch nicht damit begonnen haben, mit Unicode herumzuprobieren, finden Sie in Anhang C eine kleine Einführung in das Thema. Irgendwann müssen Sie sowieso in den sauren Apfel beißen, also warum nicht einfach jetzt? Um Unicode geht es hier und da überall in diesem Buch, vor allem aber in den Kapiteln über Skalare (Kapitel 2), Input/Output (Kapitel 5) und Sortieren (Kapitel 14).

- Die Kapitel über reguläre Ausdrücke sind ausführlicher als zuvor und umfassen die neuen Features in Perl 5.14, die Case-Folding in Unicode betreffen. Bei den Operatoren in regulären Ausdrücken gibt es die neuen Switches `/a`, `/u` und `/l`. Wir behandeln jetzt auch das Matching nach Unicode-Eigenschaften mit den Features `\p{}` und `\P{}` von regulären Ausdrücken.
- In Perl 5.14 wurde ein nichtdestruktiver Ersetzungsoperator hinzugefügt (Kapitel 9), der sich als wirklich praktisch erwiesen hat.
- Intelligentes Matching und `given-when` haben sich seit ihrer Einführung mit Perl 5.10 etwas verändert. Deshalb haben wir Kapitel 15 ein wenig verändert, um auch die neuen Regeln abzudecken.
- Wir haben Kapitel 11, Perl-Module, aktualisiert und erweitert, um die neuesten Änderungen mit hineinzunehmen, unter anderem das Zeroconf-Werkzeug *cpanm*. Außerdem wurden einige Modulbeispiele hinzugefügt.
- Einige der Punkte, die bislang in Anhang B bei den »fortgeschrittenen, aber nicht beschriebenen Features« zu finden waren, sind in den Haupttext aufgenommen worden. Besonders erwähnenswert sind hier die Verwendung des »dicken Pfeils« `=>` in Hashes (Kapitel 6) und die Verwendung von `splice` in Listen und Arrays (Kapitel 3).

Danksagungen

Von Randal

Ich möchte mich bei den früheren und jetzigen Stonehenge-Trainern (Joseph Hall, Tom Phoenix, Chip Salzenberg, brian d foy und Tad McClellan) für ihre Bereitschaft bedanken, hinauszugehen und Woche für Woche vor den Kursteilnehmern zu stehen, um mit ihren Notizen zurückzukommen und zu berichten, was funktioniert hat und was nicht, damit wir das Material für dieses Buch feiner abstimmen konnten. Besonders hervorheben möchte ich meinen Mitautor und Geschäftspartner Tom Phoenix dafür, dass er viele Stunden daran gearbeitet hat, den Lama-Kurs zu verbessern und das hervorragende Grundgerüst für den größten Teil dieses Buchs zu erstellen. Danke an brian d foy, der ab der 4. Auflage Hauptautor wurde und mich damit von diesem ewigen To-do-Eintrag in meiner Inbox befreit hat.

Ein Dankeschön auch an alle bei O'Reilly, insbesondere an unsere geduldige Lektorin Allison Randal (mit der ich nicht verwandt bin, aber sie hat natürlich einen wunderbaren Nachnamen), den derzeit verantwortlichen Lektor Simon St. Laurent sowie Tim O'Reilly, der mir überhaupt die Gelegenheit zu dem Kamel- und dem Lama-Buch eröffnet hatte.

Ich stehe außerdem in der Schuld bei den Tausenden von Leuten, die die letzten Auflagen des Lama-Buchs gekauft haben, so dass ich weder unter der Brücke noch im

Gefängnis gelandet bin. Außerdem danke ich meinen Kursteilnehmern, die mir beigebracht haben, ein besserer Kursleiter zu sein. Des Weiteren danke ich der erstaunlichen Anzahl von Fortune-1000-Kunden, die unsere Kurse in der Vergangenheit gebucht haben und dies auch in Zukunft tun werden.

Wie immer, ein besonderes Dankeschön an Lyle und Jack dafür, dass sie mir fast alles beigebracht haben, was ich über das Schreiben weiß.

Von Tom

Ich schließe mich Randals Dank an alle bei O'Reilly an. Für die 3. Auflage war Linda Mui unsere Lektorin, und ich bin ihr immer noch dankbar für die Geduld, die sie bewies, während sie uns dazu brachte, die allzu übertriebenen Witze und Fußnoten rauszuwerfen; gleichzeitig muss ich betonen, dass sie keinerlei Schuld für die verbliebenen Albernheiten trifft. Linda und Randal führten mich durch den Schreibprozess, wofür ich ihnen dankbar bin. Später übernahm dann Allison Randal die Aufgabe der Lektorin, und mittlerweile ist es Simon St. Laurent. Diesen beiden gilt mein Dank für ihre ganz besonderen Beiträge.

Außerdem schließe ich mich Randal an, was die anderen Stonehenge-Trainer angeht, aber auch ihn selbst: Sie haben sich so gut wie nie darüber beklagt, wenn ich unerwartet das Kursmaterial verändert habe, um eine neue Lehrmethode auszuprobieren. Ihr habt eine Menge verschiedene Perspektiven hinsichtlich der Lehrmethoden beigetragen, die ich ohne euch nie kennengelernt hätte.

Ich arbeitete viele Jahre am Oregon Museum of Science and Industry (OMSI) und ich möchte euch allen danken, dass ich meine Lehrmethoden verbessern durfte, indem ich lernte, einen oder zwei Witze in jede nur mögliche Tätigkeit einzubauen.

Danke auch an die vielen Leute im Usenet, die ihr mir eure Wertschätzung und Aufmunterungen für meine Beiträge dort gezeigt habt. Wie immer hoffe ich, es hilft.

Auch den vielen Studenten gilt Dank, die mir mit ihren Fragen (und ihren verdatterten Gesichtern) gezeigt haben, wann ich nach einer anderen Möglichkeit suchen musste, ein bestimmtes Konzept zu erklären. Ich hoffe, dass das vorliegende Buch jede noch verbliebene Verwirrung auflöst.

Selbstverständlich gilt meine tiefe Dankbarkeit insbesondere meinem Koautor Randal, der mir die Freiheit gab, verschiedene Arten auszuprobieren, das Material sowohl im Kurs als auch hier im Buch zu präsentieren. Außerdem danke ich ihm für seine Initiative, dieses Material überhaupt als Buch herauszubringen. Ich bin schwer beeindruckt von deinen Anstrengungen sicherzustellen, dass niemand anders in die rechtlichen Schwierigkeiten gerät, die so viel von deiner Zeit und Energie gestohlen haben; du bist ein gutes Vorbild.

Von brian

Ich habe zuerst Randal zu danken, da ich Perl mithilfe der 1. Auflage dieses Buchs erlernte. 1998 musste ich es noch einmal erlernen, als er mich bat, als Lehrer für Stonehenge zu arbeiten. Lehren ist oft der beste Weg, etwas zu lernen. Seit dieser Zeit ist Randal mein Mentor für Perl und viele andere Dinge, von denen er meinte, dass ich sie lernen sollte – zum Beispiel zu der Zeit, als er entschied, dass wir für eine Demonstration auf einer Webkonferenz Smalltalk an Stelle von Perl verwenden sollten. Ich bin immer wieder überrascht über die Vielfalt seines Wissens. Er hat mir Perl beigebracht, und jetzt helfe ich selbst bei dem Buch mit, mit dem ich einst begonnen hatte. Ich fühle mich sehr geehrt, Randal.

Während der ganzen Zeit, als ich für Stonehenge arbeitete, habe ich Tom Phoenix wahrscheinlich insgesamt höchstens zwei Wochen gesehen, aber ich unterrichte schon seit Jahren seine Version des Perl-Einführungskurses. Diese Version stellte die Grundlage für die 3. Auflage dieses Buchs dar. Dadurch, dass ich diesen Kurs unterrichte, habe ich neue Möglichkeiten gefunden, so gut wie alles zu erklären, und sogar noch die eine oder andere Ecke von Perl genauer kennengelernt.

Als ich Randal davon überzeugte, dass meine Mitarbeit bei der Aktualisierung des Lama-Buchs von Vorteil sein würde, wurde mir die Verantwortung für das Proposal an den Verlag, die Gliederung der neuen Auflage und das Managen der Manuskriptversionen übertragen. Unsere Lektorin Allison Randal half mir bei all dem sehr und ertrug meine ständigen E-Mails ohne Klagen. Nach ihrem Aufbruch zu neuen Ufern hat Simon St. Laurent uns als Lektor und Insider bei O'Reilly sehr geholfen und immer die richtige Mondphase abgewartet, um eine weitere Überarbeitung anzuregen.

Von uns allen

Danke unseren Fachkorrektoren David H. Adler, Alan Haggai Alavi, Andy Armstrong, Dave Cross, Chris Devers, Paul Fenwick, Stephen Jenkins, Matthew Musgrove, Jacinta Richardson, Steve Peters, Peter Scott, Wil Wheaton und Karl Williamson für ihre Anmerkungen zu unserem Manuskript.

Danke auch an die vielen Kursteilnehmer, die uns mitgeteilt haben, welche Teile des Kursmaterials über die Jahre verbesserungswürdig waren. Euretwegen sind wir heute so stolz darauf.

Danke auch an die vielen Perl Mongers, die uns ein Zuhause gegeben haben, wenn wir ihre Städte besucht haben. Lasst es uns irgendwann wiederholen.

Und schließlich gilt unser aufrichtigster Dank unserem Freund Larry Wall dafür, dass er die Weisheit besaß, seine wirklich coolen und mächtigen Werkzeuge dem Rest der Welt zugänglich zu machen, damit wir alle unsere Arbeit ein bisschen schneller, leichter und mit mehr Spaß erledigen können.

KAPITEL 1

Einleitung

Willkommen zum Lama-Buch!

Dies ist die 6. Auflage eines Buchs, in dessen Genuss seit 1993 über eine halbe Million Leser gekommen sind. Wir hoffen zumindest, dass sie es genossen haben. Uns hat es jedenfalls Spaß gemacht, es zu schreiben.[1]

Fragen und Antworten

Wahrscheinlich haben Sie einige Fragen zu Perl, und vielleicht auch einige zu diesem Buch, besonders wenn Sie schon ein wenig herumgeblättert haben, um zu sehen, was dieses Buch so alles bietet. Wir werden daher dieses Kapitel dazu benutzen, Ihre Fragen zu beantworten bzw. Ihnen zu erzählen, wo Sie Antworten herbekommen, die nicht in diesem Buch stehen.

Ist dies das richtige Buch für Sie?

Wenn Sie und wir uns auch nur ein wenig ähnlich sind, dann stehen Sie vermutlich jetzt in einem Buchladen[2] und fragen sich, ob Sie dieses Lama-Buch kaufen und Perl lernen sollen oder das Buch da drüben, um eine Sprache zu lernen, die nach einer Schlange[3]

1 Um sicherzugehen: Die 1. Auflage wurde von Randal L. Schwartz geschrieben, die 2. von Randal und Tom Christiansen, die 3. Auflage von Randal und Tom Phoenix und die letzten drei von Randal, Tom Phoenix und brian d foy. Wenn wir also »wir« sagen, meinen wir die letzte Gruppe. Wenn Sie sich nun fragen, warum es uns Spaß *gemacht hat* (in der Vergangenheitsform), dieses Buch zu schreiben, obwohl wir uns noch auf der ersten Seite befinden, ist die Antwort leicht: Wir haben hinten angefangen und uns dann nach vorne vorgearbeitet. Das mag jetzt zwar etwas seltsam klingen, aber um ehrlich zu sein, war der Rest einfach, als der Index erst einmal fertig war.

2 Wenn Sie und wir uns tatsächlich ähnlich sind, stehen Sie jetzt eher in einer *Bücherei* und nicht in einem Buchladen. Aber wir wollen ja nicht kleinlich sein.

3 Bevor Sie uns jetzt schreiben, um uns mitzuteilen, dass es sich eigentlich um eine Comedy-Truppe handelt, sollten wir Ihnen vielleicht erklären, dass wir – mit einem Buchstabendreher – eigentlich CORBA meinen.

benannt ist, nach einem Getränk oder einem Buchstaben des Alphabets. Sie haben genau zwei Minuten Zeit, bevor der Verkäufer herüberkommt, um Ihnen zu sagen, dass das hier keine Bibliothek sei,[4] und Sie auffordert, entweder etwas zu kaufen oder den Laden zu verlassen. Vielleicht wollen Sie die zwei Minuten nutzen, um ein kurzes Perl-Programm zu sehen, anhand dessen Sie erkennen können, was Perl so alles kann. In diesem Fall sollten Sie die Perl-Blitztour weiter hinten in diesem Kapitel mitmachen.

Das hier ist kein Nachschlagewerk. Es ist eine Einführung in die Grundlagen von Perl, das Ihnen gerade das Nötigste vermittelt, um einfache Programme für den Hausgebrauch zu schreiben. Wir behandeln nicht alle Details eines jeden Themas, und wir verteilen einige der Themen über mehrere Kapitel, so dass Sie mit bestimmten Konzepten jeweils dort vertraut gemacht werden, wo Sie sie auch brauchen.

Wir schreiben für Leser, die schon das eine oder andere übers Programmieren wissen und jetzt Perl lernen möchten. Wir gehen davon aus, dass Sie sich schon einigermaßen damit auskennen, wie man ein Terminal benutzt, Dateien editiert und Programme laufen lässt – aber eben nicht mit Perl-Programmen. Sie wissen, was Variablen, Subroutinen usw. sind, aber Sie wollen wissen, wie so etwas in Perl funktioniert.

Das heißt nicht, dass Sie als blutiger Anfänger, der noch nie ein Terminal-Programm angesehen oder eine Codezeile geschrieben hat, durchgehend Bahnhof verstehen werden. Sie werden vielleicht nicht alles verstehen, was wir schreiben, wenn Sie das Buch das erste Mal durchgehen, aber eine Menge Anfänger haben dieses Buch schon zum Lernen benutzt, ohne völlig zu verzweifeln. Der Trick besteht darin, sich keine Gedanken um all das zu machen, was man vielleicht nicht versteht, sondern sich stattdessen einfach auf die grundlegenden Konzepte zu konzentrieren, die wir vorstellen. Sie werden vielleicht etwas länger brauchen als ein erfahrener Programmierer, aber irgendwo müssen Sie ja anfangen.

Und: Das hier sollte nicht das einzige Perl-Buch sein, das Sie jemals lesen. Es ist nur eine Einführung und bei Weitem nicht umfassend. Es ist eine erste Starthilfe, um Sie auf den richtigen Weg zu bringen, damit Sie dann, wenn Sie so weit sind, mit unseren anderen Büchern weitermachen können: *Intermediate Perl* (deutsch: *Einführung in Perl-Objekte*, *Referenzen & Module*, zurzeit nur als E-Book erhältlich – bekannt als das »Alpaka-Buch«) und *Mastering Perl*. Das maßgebliche Nachschlagewerk zu Perl ist *Programming Perl*, das auch das »Kamel-Buch« genannt wird (deutsch: *Programmieren mit Perl*; zurzeit nur als E-Book erhältlich).

Wir möchten außerdem anmerken, dass dieses Buch zwar Perl bis Version 5.14 behandelt, aber auch nutzbringend verwendet werden kann, wenn Sie eine frühere Version benutzen. Dann verpassen Sie vielleicht ein paar coole neue Features, aber auf jeden Fall lernen Sie die grundlegende Verwendung von Perl kennen. Die älteste Version, die wir in

4 Es sei denn, es ist wirklich eine.

unsere Überlegungen mit einbeziehen werden, ist Perl 5.8, obwohl es vor fast zehn Jahren veröffentlicht wurde.

Warum gibt es so viele Fußnoten?

Danke, dass Sie das bemerkt haben. Es gibt *eine Menge* Fußnoten in diesem Buch. Ignorieren Sie sie. Die Fußnoten werden gebraucht, weil Perl eine ganze Reihe von Ausnahmen zu seinen Regeln hat. Das ist eine gute Sache, da das reale Leben auch voller Ausnahmen ist.

Das hat zur Folge, dass wir nicht ohne zu lügen sagen können: »Der *fitzbin*-Operator frobniziert die husistatische Variable.«, ohne die Ausnahmen[5] in einer Fußnote zu erläutern. Da wir ziemlich ehrliche Leute sind, müssen wir also eine Menge Fußnoten schreiben. Sie dagegen können ehrlich sein, ohne die Fußnoten lesen zu müssen (witzig, wie sich das ergibt). Die Fußnoten bieten zusätzliche Informationen, die Sie für die Grundkonzepte nicht benötigen.

Viele der Ausnahmen haben mit der Portierbarkeit des Codes zu tun. Die Geschichte von Perl begann auf Unix-Systemen und ist dort auch heute noch ziemlich tief verwurzelt. Wo immer es möglich war, haben wir versucht, auf unerwartetes Verhalten hinzuweisen, egal ob es daher rührt, dass ein Programm auf einem Nicht-Unix-System laufen soll, oder auf einem anderen Grund beruht. Wir hoffen, dass dieses Buch auch für unsere Leser, die nichts über Unix wissen, eine gute Einführung in Perl darstellt. (Und die werden nebenbei noch etwas über Unix lernen – ohne Extrakosten, versteht sich.)

Außerdem folgen die meisten Ausnahmen der alten »80/20«-Regel, die besagt, dass 80% des Verhaltens von Perl in 20% der Dokumentation beschrieben werden kann. Die übrigen 20% des Verhaltens nehmen dafür die verbleibenden 80% der Dokumentation ein. Um dieses Buch also übersichtlich zu halten, besprechen wir im Haupttext die gängigsten und leicht erklärbaren Anwendungsformen. In den Fußnoten (die außerdem eine kleinere Schrift verwenden, so dass wir bei gleichem Platzverbrauch mehr sagen können)[6] gehen wir dann auf die anderen Dinge ein. Wenn Sie das Buch zum ersten Mal ganz durchgelesen haben, ohne dabei die Fußnoten mitzulesen, werden Sie sich vermutlich einige Abschnitte zur Vertiefung noch einmal ansehen wollen. Wenn Sie an diesem Punkt angekommen sind oder die Neugier beim Lesen einfach zu groß wird, können Sie die Fußnoten lesen. Ein Großteil sind sowieso nur Informatikerwitze.

---

5 Außer dienstags während eines Stromausfalls, wenn Sie bei Tag-und-Nacht-Gleiche Ihren Ellenbogen verdrehen oder wenn bei einer Perl-Version vor 5.12 `use integer` innerhalb eines von einer Prototyp-Subroutine aufgerufenen Schleifenblocks benutzt wird.

6 Wir haben uns sogar überlegt, das ganze Buch als Fußnote zu konzipieren, um die Seitenzahl klein zu halten. Wir haben diese Idee jedoch wieder verworfen, weil uns Fußnoten mit Fußnoten dann doch ein bisschen zu verrückt vorkamen.

Was ist mit den Übungen und ihren Lösungen?

Die Übungen finden Sie jeweils am Ende jedes Kapitels. Sie basieren auf den Erfahrungen, die wir mit diesem Kursmaterial bereits vor tausenden[7] von Kursteilnehmern gemacht haben. Wir haben diese Übungen sorgfältig zusammengestellt, um Ihnen die Gelegenheit zu geben, auch einmal Fehler zu machen.

Nicht dass wir *wollen*, dass Sie Fehler machen, aber Sie brauchen die *Gelegenheit* dazu. Die meisten dieser Fehler werden Ihnen während Ihrer Perl-Karriere begegnen, warum also nicht gleich jetzt? Jeden Fehler, den Sie beim Lesen dieses Buchs begehen, machen Sie nicht noch einmal, wenn Sie unter Zeitdruck ein Programm schreiben müssen. Außerdem sind wir, falls einmal etwas schiefgehen sollte, die ganze Zeit bei Ihnen. Anhang A, *Lösungen zu den Übungen*, enthält unsere Lösungen für jede Übung und bespricht die Fehler, die Sie gemacht haben, sowie ein paar Fehler, die Sie nicht gemacht haben. Sehen Sie sich diese Lösungen und Erläuterungen an, wenn Sie die Übungen erledigt haben.

Aber schlagen Sie die Antwort nicht nach, bevor Sie nicht ernsthaft versucht haben, selbst auf die Lösung zu kommen. Ihr Lernerfolg wird besser sein, wenn Sie die Lösung von sich aus finden, als wenn Sie sie einfach nachlesen. Schlagen Sie Ihren Kopf nicht gegen die Wand, wenn Sie mal eine Aufgabe nicht lösen können. Ärgern Sie sich nicht und machen Sie einfach mit dem nächsten Kapitel weiter.

Selbst wenn Sie überhaupt keine Fehler machen, sollten Sie sich die Antworten ansehen, wenn Sie mit der Übung fertig sind. Der Begleittext zeigt einige Details der Übungsprogramme auf, die auf den ersten Blick vielleicht nicht ganz so offensichtlich sind.

Wenn Sie gern zusätzliche Übungen machen möchten, probieren Sie es mit dem englischsprachigen *Learning Perl Student Workbook*, in dem zu jedem der Kapitel mehrere weitere Übungen angefügt sind.

Was bedeuten die Zahlen am Anfang der Übungen?

Jeder Übung ist vor dem Text eine Zahl in eckigen Klammern vorangestellt:

1. [2] Was bedeutet die Zahl 2, die am Anfang einer Übung steht?

Diese Zahl ist unsere (grobe) Schätzung, wie viele Minuten Sie in etwa für das Absolvieren einer Übung brauchen werden. Seien Sie bitte nicht überrascht, wenn Sie (inklusive des Schreibens, Testens und Debuggens) nur die halbe Zeit benötigen oder auch nach mehr als der doppelten Zeit noch nicht fertig sind. Wenn Sie allerdings wirklich nicht weiterkommen, werden wir natürlich nicht verraten, dass Sie in Anhang A nachgesehen haben, wie die Antwort lautet.

7 Natürlich nicht gleichzeitig.

Wie soll ich als Perl-Dozent vorgehen?

Wenn Sie Perl-Kurse leiten und sich entschieden haben, dieses Buch im Kurs einzusetzen (wie es schon viele andere getan haben), sollten Sie wissen, dass wir jeden Übungsblock so konzipiert haben, dass die meisten Schüler in 45 bis 60 Minuten damit fertig sind und noch etwas Zeit für eine Pause bleibt. Die Übungen einiger Kapitel brauchen weniger Zeit, andere dauern womöglich etwas länger. Nachdem wir den Übungen diese kleinen Zahlen in den eckigen Klammern vorangestellt hatten, wussten wir plötzlich einfach nicht mehr, wie wir sie zusammenzählen sollten. (Glücklicherweise wissen wir, wie wir das von Computern erledigen lassen können.)

Es gibt von uns ein begleitendes englischsprachiges Buch, das *Learning Perl Student Workbook*, in dem Sie zu jedem der Kapitel zusätzliche Übungen finden. Wenn Sie sich das Workbook zur vierten Auflage besorgen, müssen Sie für sich die Kapitelreihenfolge ändern, weil wir in der vorliegenden Auflage ein Kapitel hinzugefügt und ein anderes verschoben haben.

Was bedeutet »Perl«?

Perl wird manchmal die »*P*ractical *E*xtraction and *R*eport *L*anguage« genannt, auch wenn es schon mal als »Pathologically Eclectic Rubbish Lister« (krankhaft zusammengeschustertes Auflistungsprogramm für wirres Zeug) bezeichnet wird. Hierbei handelt es sich eigentlich um ein Retronym und nicht um ein Akronym. Soll heißen, Larry Wall, der Erfinder von Perl, hatte zunächst den Namen und erst später die Interpretation der Buchstaben. Deshalb schreibt man »Perl« auch nicht komplett in Großbuchstaben. Es besteht kein Anlass, darüber zu streiten, welche Interpretation die richtige ist – beide Versionen werden von Larry Wall unterstützt.

Sie werden manchmal auch die Schreibweise »perl« mit kleinem »p« sehen. Im Allgemeinen bezieht sich die Schreibweise »Perl« auf die Sprache, während mit »perl« der Interpreter gemeint ist, der Ihre Perl-Programme kompiliert und ausführt. Intern schreiben wir Programmnamen so: *perl*.

Warum schuf Larry Perl?

Larry entwickelte Perl Mitte der achtziger Jahre, um aus Usenet-News-artigen Dateistrukturen eine Reihe von Berichten zu erstellen, weil *awk* bei dieser Aufgabe die Luft ausging. Faul, wie er war,[8] versuchte Larry das Problem mit einem Mehrzweckwerk-

8 Es ist keine Beleidigung für Larry, wenn wir sagen, er sei faul – Faulheit ist eine Tugend. Die Schubkarre wurde von jemandem erfunden, der zu faul war, Dinge zu tragen, und das Schreiben wurde von jemandem erfunden, der zu faul war, sich alles zu merken. Perl wurde von jemandem erfunden, der zu faul war, sich für seine Arbeit eine komplett neue Computersprache auszudenken.

zeug zu erschlagen, das er noch für mindestens einen anderen Zweck benutzen konnte. Das Ergebnis war Perl Version Null.

Warum hat Larry nicht einfach eine andere Sprache benutzt?

Es gibt doch schon genug Computersprachen, oder? Als Larry Perl erfand, konnte er jedoch keine finden, die seinen Bedürfnissen entsprach. Wäre eine der heutigen Sprachen damals verfügbar gewesen, hätte er vermutlich eine davon benutzt. Er benötigte eine Sprache, in der er, wie auf der Shell oder in *awk*, schnell mal etwas zusammentippen konnte. Dabei sollte sie allerdings die Mächtigkeit von höher entwickelten Werkzeugen, wie etwa *grep*, *cut*, *sort* und *sed*,[9] haben, ohne dabei auf eine Sprache wie C zurückgreifen zu müssen.

Perl versucht, die Lücke zwischen Low-Level-Programmierung (wie zum Beispiel in C, C++ oder Assembler) und High-Level-Programmierung (zum Beispiel Shell-Programmierung) zu schließen. Low-Level-Programme sind in der Regel schwer zu schreiben und hässlich, dafür aber schnell und unbegrenzt einsetzbar. Gut geschriebene Low-Level-Programme sind auf der richtigen Maschine schwer zu schlagen, und es gibt kaum etwas, das Sie hier nicht machen können. Das andere Extrem, die High-Level-Programme, tendiert dazu, langsam, schwer, hässlich und begrenzt zu sein. Viele Dinge lassen sich mit Shell- oder Batch-Programmierung nur erledigen, wenn es auf Ihrem System auch ein Kommando gibt, das diese Funktionalität bereitstellt. Perl ist einfach, fast unbegrenzt, meistens schnell und ein bisschen hässlich.

Lassen Sie uns diese vier Behauptungen über Perl noch einmal einzeln betrachten:

Perl ist einfach. Wie Sie sehen werden, bedeutet das, dass Perl zwar einfach zu *benutzen* ist, wodurch es aber nicht unbedingt einfach zu *lernen* ist. Bevor Sie Auto fahren können, verbringen Sie viele Wochen oder Monate damit, es zu lernen. Aber heute fällt es Ihnen leicht. Wenn Sie so viel Zeit, wie Sie gebraucht haben, um fahren zu lernen, investieren, um Perl zu lernen, wird auch Perl Ihnen leicht fallen.[10]

Perl ist fast unbegrenzt. Es gibt wenige Dinge, die Sie mit Perl nicht machen können. Perl ist nicht gerade die geeignete Sprache, um einen Interrupt-gesteuerten Gerätetreiber auf Basis eines Mikrokernels zu schreiben (obwohl auch das schon gemacht wurde). Die meisten Durchschnittsanwendungen bewältigt Perl ohne Probleme, von schnell zusammengetippten Einmalprogrammen bis hin zu Anwendungen von industriellem Ausmaß.

Perl ist meistens schnell. Das liegt daran, dass niemand Perl weiterentwickelt, ohne es selbst zu benutzen – also wollen wir alle, dass es schnell ist. Bringt jemand ein neues Feature ein, das zwar cool wäre, andere Programme aber verlangsamen könnte, ist es ziem-

9 Machen Sie sich keine Sorgen, wenn Sie nicht wissen, worum es sich dabei handelt. Wichtig ist nur, dass es Programme sind, die Larry in seiner Unix-Werkzeugkiste hatte, die aber für die anfallenden Arbeiten nicht ausreichten.

10 Wir hoffen allerdings, dass Ihr Auto nicht so oft abstürzt.

lich sicher, dass Larry es zurückweist, bis wir einen Weg gefunden haben, es schnell genug zu machen.

Perl ist ein bisschen hässlich. Das stimmt. Perls Symbol ist das Kamel, das Tier auf dem Cover des hoch geschätzten Kamel-Buchs (auch bekannt als *Programming Perl* bzw. *Programmieren mit Perl* [O'Reilly] von Larry Wall, Tom Christiansen und John Orwant), dem Cousin dieses Lama-Buchs (und seiner Schwester, des Alpaka-Buchs). Kamele sind auch ein bisschen hässlich, aber sie arbeiten hart, selbst unter schwierigen Bedingungen. Kamele erledigen die Arbeit trotz aller Schwierigkeiten, selbst wenn sie schlecht aussehen, noch schlechter riechen und einen manchmal anspucken. Perl ist so ähnlich.

Ist Perl nun schwer oder leicht?

Perl ist einfach zu benutzen, manchmal aber schwer zu lernen. Das ist selbstverständlich eine Verallgemeinerung. Aber als Larry Perl entwickelte, musste er eine Menge Kompromisse eingehen. Wo es möglich war, dem Programmierer etwas zu erleichtern, auch wenn es dadurch für den Schüler schwerer wurde, hat Larry sich fast immer zugunsten des Programmierers entschieden. Der Grund besteht darin, dass Sie Perl nur einmal lernen, aber immer wieder benutzen.[11] Perl hat eine Reihe von Annehmlichkeiten, die dem Programmierer viel Zeit sparen helfen. So haben die meisten Funktionen ein Standardverhalten. Der Standard ist die Art, auf die Sie diese Funktion üblicherweise verwenden. Betrachten Sie einmal diese Zeilen Perl-Code:

```
while (<>) {
  chomp;
  print join("\t", (split /:/)[0, 2, 1, 5] ), "\n";
}
```[12]

Komplett ausgeschrieben, also ohne das Standardverhalten und die Abkürzungen von Perl zu benutzen, wäre dieser Codeschnipsel etwa zehn- bis zwölfmal länger. Sie würden also auch länger brauchen, um ihn zu lesen und zu schreiben. Es wäre außerdem schwieriger, den Code zu debuggen, und es müssten mehr Variablen verwendet werden. Wenn Sie etwas Perl kennen und die Variablen im Code gar nicht sehen, so ist das ein Teil dessen, was wir Ihnen hier klarzumachen versuchen. Die Variablen werden durch das Standardverhalten definiert. Den Preis für diese Erleichterung beim Programmieren zahlen Sie beim Lernen, weil Sie das Standardverhalten und die Abkürzungen ebenfalls lernen müssen.

Eine gute Analogie ist die richtige und häufige Verwendung von Kurzformen im Englischen. »Will not« bedeutet zwar dasselbe wie »won't«, aber die meisten verwenden eher

11 Sofern Sie eine Programmiersprache nur ein paar Minuten in der Woche oder im Monat benutzen, werden Sie vermutlich eine Sprache vorziehen, die sich leichter lernen lässt, da Sie das meiste, was Sie gelernt haben, zwischenzeitlich wieder vergessen. Perl ist für Leute gedacht, die mindestens zwanzig Minuten täglich programmieren.

12 Wir werden das Programm hier jetzt nicht komplett erklären, aber dieses Beispiel liest Daten aus einer Eingabedatei mit einem bestimmten Format und gibt einige der Daten in einem anderen Format wieder aus. Sämtliche hier verwendeten Sprachmerkmale werden im Buch behandelt.

die Kurzform. Sie spart Zeit, und da sie jeder kennt, ist sie auch sinnvoll. Auf vergleichbare Weise kürzen die »Kurzformen« von Perl verbreitete »Wendungen« ab, so dass sie schneller »gesprochen« werden und als ein einzelner Ausdruck statt als Reihe voneinander unabhängiger Schritte verstanden werden können.

Sobald Sie einmal mit Perl vertraut sind, werden Sie weniger Zeit benötigen, das Shell-Quoting (oder C-Deklarationen) richtig hinzubekommen, und mehr Zeit haben, um im Internet zu surfen. Die klaren Konstrukte von Perl ermöglichen Ihnen, (mit minimalem Aufwand) coole Einmallösungen oder auch allgemein verwendbare Werkzeuge zu erstellen. Sie können diese Werkzeuge zu Ihrer nächsten Arbeitsstätte mitnehmen, da Perl hochportabel und dort oft bereits installiert ist, so dass Sie noch mehr Zeit haben, um im Web zu surfen.

Perl besitzt ein sehr hohes Sprachniveau. Das bedeutet, dass der Code eine ziemlich hohe »Dichte« aufweist. Ein Perl-Programm ist nur etwa 30 bis 70 Prozent so lang wie ein entsprechendes C-Programm. Dadurch ist Perl schneller zu schreiben, zu lesen und zu debuggen, sowie leichter zu pflegen. Sie brauchen nicht viel Programmiererfahrung, um zu sehen, dass die gesamte Subroutine klein genug ist, um vollständig auf den Bildschirm zu passen. Sie müssen also nicht ständig hin- und herscrollen, um zu sehen, was gerade passiert. Und da sich die Anzahl der Bugs in einem Programm ungefähr proportional zur Länge des Quellcodes verhält[13] (und nicht proportional zur Funktionalität), bedeutet der kürzere Quellcode bei Perl im Schnitt auch weniger Fehler.

Wie in jeder Sprache ist es auch in Perl möglich, so zu programmieren (»write-only«), dass das Programm hinterher nicht mehr lesbar ist. Wenn Sie allerdings sorgfältig vorgehen, können Sie diesen oft gehörten Vorwurf vermeiden. Für Außenstehende sieht Perl aus wie ein »Rauschen in der CPAN-Leitung«, für Eingeweihte jedoch sieht es aus wie die Noten einer großartigen Sinfonie. Wenn Sie sich an die Richtlinien in diesem Buch halten, sollten Ihre Programme leicht zu lesen und ebenso leicht zu pflegen sein. Dafür werden Sie aber vermutlich den »Obfuscated Perl Contest« leider nicht gewinnen.

Warum ist Perl so beliebt?

Nachdem Larry ein bisschen mit Perl herumgespielt und es um ein paar Dinge ergänzt hatte, veröffentlichte er es in der Gemeinschaft der Usenet-Leserschaft (auch als das »Net« bezeichnet). Die Benutzer dieses zusammengewürfelten Haufens von Systemen auf der ganzen Welt (schon damals mehrere zehntausend) gaben ihm Rückmeldungen und fragten nach Möglichkeiten, dieses oder jenes zu erledigen, wovon Larry eigentlich nie gedacht hatte, dass sein kleines Perl es übernehmen sollte.

Das Resultat war, dass Perl wuchs und wuchs und wuchs. Es wuchs in seinen Möglichkeiten. Es wuchs in seiner Portabilität. Was früher einmal eine kleine Sprache war, die nur auf ein paar Unix-Systemen verfügbar war, ist zu einer Größe herangewachsen, die

13 Mit einem großen Sprung, sobald ein Programmteil nicht mehr auf den Bildschirm passt.

tausende von Seiten frei erhältlicher Dokumentation beinhaltet, Dutzende von Büchern hervorgebracht hat und in einer großen Anzahl von Mainstream-Usenet-Newsgruppen (und einem Dutzend Newsgruppen und Mailinglisten, die nicht so »Mainstream« sind) diskutiert wird, die eine kaum zu zählende Anzahl von Lesern haben. Perl ist auf fast jedem Betriebssystem, das heute benutzt wird, zu Hause. Und vergessen Sie auch nicht das Lama-Buch.

Was geschieht in Zukunft mit Perl?

Zwar schreibt Larry den Code heutzutage nicht mehr selbst, aber er leitet die Entwicklung und trifft die großen Entscheidungen. Perl wird zum großen Teil von einer kühnen Gruppe von Leuten gewartet, die die »Perl 5 Porters«[14] genannt werden. Sie können ihre Arbeit und Diskussionen auf der Mailingliste *perl5-porters@perl.org* mitverfolgen.

Während wir diesen Text schreiben (März 2011), passiert eine Menge mit Perl. In den letzten paar Jahren haben viele Leute an der nächsten größeren Version von Perl gearbeitet: Perl 6.

Kurz gesagt, ist Perl 6 heute eine völlig andere Sprache. Das geht so weit, dass seine wichtigste Implementierung jetzt Rakudo heißt. Perl 6 nahm seinen Anfang im Jahr 2000 als etwas, das vielleicht eines Tages Perl 5 ersetzen sollte, das ja aufgrund der langen Lag-Zeiten bei den Releases 5.6, 5.8 und 5.10 eine ziemliche Flaute erlebte. Schließlich ergab es sich durch diverse Zufälle und Berührungspunkte, dass Perl 5 wieder auflebte, während Perl 6 immer schleppender vorankam. So eine Ironie ...

Die Entwicklung von Perl 5 ist wieder in Schwung gekommen: Derzeit gibt es monatliche Releases von experimentellen Versionen und etwa eine neue Wartungsversion pro Jahr. Die letzte Auflage dieses Buchs beschäftigte sich mit 5.10, und wir hatten keine Zeit, es zu überarbeiten, bis Perl 5.12 herauskam. Das vorliegende Buch erscheint etwa zu der Zeit, zu der Perl 5.14 veröffentlicht werden soll. Und gleichzeitig beschäftigen sich die Perl-5-Portierer schon mit Perl 5.16.

Wofür ist Perl eigentlich geeignet?

Perl ist gut für schnell zusammengeschusterte Programme, die Sie kurz mal in drei Minuten herunterhacken. Perl ist aber auch für lange und ausgedehnte Programme gut, für die ein Team von einem Dutzend Programmierern drei Jahre braucht. Sie werden allerdings eine Menge Programme schreiben, für die Sie von der Konzeption bis zum vollständig getesteten Code nicht mal eine Stunde brauchen.

Perl ist für Probleme optimiert, die zu ungefähr 90% mit Text und ungefähr 10% mit anderen Dingen zu tun haben. Diese Beschreibung scheint auf die meisten Programmieraufgaben zu passen, die sich heutzutage stellen. In einer perfekten Welt würden alle Pro-

14 Portierer, nicht Portiers!

grammierer alle Sprachen kennen und dadurch in der Lage sein, immer die beste Sprache für ein bestimmtes Projekt zu wählen. Meistens würde man sich wohl für Perl entscheiden.[15]

Obwohl das World Wide Web noch nicht einmal ein Glänzen in Tim Berners-Lees Augen war, als Larry sich Perl ausdachte, war es doch eine Heirat im (Use-)Net. Manche Leute sagen, dass es durch den Einsatz von Perl Anfang der neunziger Jahre möglich wurde, eine große Menge an Inhalten sehr schnell in das HTML-Format umzuwandeln. Das war wichtig, da das Web ohne Inhalte nicht existieren konnte. Selbstverständlich war Perl auch die Sprache der Wahl für kleine CGI-Skripten (Programme, die von einem Webserver ausgeführt werden). Das ging sogar so weit, dass schlecht informierte Leute immer wieder fragten, »Ist Perl nicht sowieso nur CGI?« oder »Warum sollte man Perl überhaupt für etwas anderes als CGI einsetzen?« Wir finden solche Fragen sehr amüsant.

Wofür ist Perl nicht geeignet?

Wofür ist Perl denn *nicht* gut, wo es doch für so viele Dinge gut geeignet ist? Nun, Sie sollten Perl nicht benutzen, wenn Sie *opaken Binärcode* erzeugen wollen. Das ist ein Programm, das Sie an Leute weitergeben oder verkaufen, ohne dass diese Ihre geheimen Algorithmen im Quellcode sehen können. Sie können Ihnen also auch nicht helfen, Ihren Code zu debuggen oder zu pflegen. Wenn Sie jemandem Ihr Perl-Programm weitergeben, geben Sie in der Regel den Quellcode weiter, nicht opaken Binärcode.

Wenn Sie opaken Binärcode haben wollen, müssen wir Ihnen leider sagen, dass es so etwas nicht gibt. Wenn man Ihr Programm installieren und ausführen kann, kann das Programm auch wieder in Quellcode in jeder beliebigen Sprache zurückverwandelt werden. Das ist nicht unbedingt derselbe Quellcode, mit dem Sie angefangen haben, aber es ist auf jeden Fall Quellcode. Der korrekte Weg, Ihre geheimen Algorithmen weiterhin geheim zu halten, besteht also vielmehr darin, eine angemessene Anzahl von Rechtsanwälten zu beschäftigen. Diese können dann eine Lizenz aufsetzen, in der steht: »Sie können *das* mit dem Code machen, *jenes* aber nicht. Sollten Sie entgegen unseren Regeln *jenes* trotzdem tun, verfügen wir über eine angemessene Anzahl von Rechtsanwälten, um sicherzustellen, dass Sie es bereuen werden.«

Wo kann ich Perl bekommen?

Vermutlich haben Sie es bereits. Da, wo *wir* hingehen, ist Perl zumindest immer zu finden. Es wird mit vielen Betriebssystemen bereits ausgeliefert, und Systemadministratoren installieren es oft auf jeder Maschine ihrer Site. Wenn sich Perl jedoch noch nicht auf

15 Nehmen Sie uns das nicht einfach so ab. Wenn Sie wissen wollen, ob Perl besser ist als Sprache X, lernen Sie beide Sprachen, probieren Sie beide aus und finden Sie heraus, welche von beiden Sie öfter benutzen. Das ist dann die Sprache, die für Sie am besten ist. Letztendlich werden Sie Perl besser verstehen, weil Sie die Sprache X erforscht haben, und umgekehrt. Sie haben die Zeit also sinnvoll genutzt.

Ihrem System befindet, können Sie es sich kostenlos besorgen. Auf den meisten Linux- und *BSD-Systemen, Mac OS X und einigen anderen ist es vorinstalliert. Firmen wie ActiveState (*http://www.activestate.com*) bieten fertige, verbesserte Distributionen für verschiedene Plattformen einschließlich Windows an. Sie können sich für Windows auch Strawberry Perl besorgen (*http://www.strawberryperl.com*), das alles mitbringt, was beim normalen Perl dabei ist, und dann noch zusätzliche Werkzeuge, um Module von Drittanbietern zu kompilieren und installieren.

Perl wird unter zwei verschiedenen Lizenzvereinbarungen verteilt. Wenn Sie – wie die meisten Leute – Perl nur *benutzen*, sind für Sie beide Lizenzen gleich gut. Wenn Sie Perl selbst verändern wollen, sollten Sie die Lizenzen jedoch etwas genauer lesen, da es einige Beschränkungen gibt, was die Verteilung von modifiziertem Code angeht. Für Leute, die Perl nicht modifizieren wollen, besagen beide Lizenzen im Prinzip: »Perl ist freie Software – viel Spaß damit!«

Perl ist nicht nur frei erhältlich, sondern läuft auch auf so ziemlich allem, was sich Unix nennt und einen C-Compiler besitzt. Sie müssen es nur herunterladen und ein oder zwei Kommandos eingeben, und schon beginnt es, sich selbst zu konfigurieren und zu kompilieren. Oder noch besser: Bringen Sie Ihren Systemadministrator dazu, es für Sie zu installieren.[16] Abgesehen von Benutzern von Unix- und Unix-artigen Systemen gab es sogar Leute, die süchtig genug nach Perl waren, um es auch für andere Systeme zu portieren, wie MacOS X, VMS, OS/2 und sogar MS/DOS und jede Spezies von Windows. Und während Sie das hier lesen, sind vermutlich schon wieder einige dazugekommen.[17] Viele dieser *Portierungen* von Perl werden mit einem Installationsprogramm verteilt, was die Installation von Perl auf diesen Systemen sogar noch leichter macht als unter Unix. Sehen Sie sich die Links im »ports«-Abschnitt des CPAN an.

Was ist das CPAN?

Das *CPAN* ist das Comprehensive Perl Archive Network, Ihre erste Anlaufstelle, wenn es um Perl geht. Hier bekommen Sie den Quellcode von Perl selbst, Portierungen für alle möglichen Nicht-Unix-Systeme, die Sie nur noch zu installieren brauchen,[18] Beispiele, Dokumentation und Erweiterungen zu Perl sowie Nachrichtenarchive, die sich mit Perl beschäftigen. Kurz gesagt: Das CPAN ist »comprehensive« (umfassend).

16 Wofür sind Systemadministratoren gut, wenn sie keine Software installieren können? Wenn Sie Schwierigkeiten haben, Ihren Administrator davon zu überzeugen, Perl zu installieren, laden Sie ihn auf eine Pizza ein. Wir haben noch nie einen Systemadministrator getroffen, der nein zu einer Pizza sagen konnte, oder zumindest zu etwas Ähnlichem, das genauso leicht zu beschaffen war.

17 Und nein, es passt (noch) nicht auf Ihren Blackberry – es ist einfach zu groß, selbst wenn es auf das Wesentliche beschränkt ist. Wir haben aber schon Gerüchte gehört, dass es unter WinCE funktioniert.

18 Auf Unix-Systemen ist es fast immer besser, Perl aus dem Quellcode selbst zu kompilieren. Auf anderen Systemen stehen eventuell C-Compiler oder andere für die Kompilierung benötigte Werkzeuge nicht zur Verfügung. Für diese Systeme finden Sie die bereits kompilierten Binärdateien im CPAN.

Das CPAN wird auf hunderten von Rechnern weltweit gespiegelt. Beginnen Sie einfach bei *http://search.cpan.org/*, um das Archiv zu durchsuchen. Falls Sie keinen Internetzugang haben, hat Ihr EDV-Buchladen vielleicht eine CD- oder DVD-ROM vorrätig, auf der die nützlichen Teile des CPAN zu finden sind. Achten Sie jedoch darauf, dass es sich um eine aktuelle Kopie des Archivs handelt, da das CPAN täglich aktualisiert wird – ein zwei Jahre altes Archiv ist eine Antiquität. Noch besser ist es, wenn Sie jemanden mit einer schnellen Netzanbindung finden, der Ihnen das aktuelle CPAN auf eine CD brennt.

Wo bekomme ich Support für Perl?

Tja, Sie haben doch den kompletten Quellcode – also können Sie die Bugs auch selbst entfernen!

Klingt nicht so gut, oder? Ist es aber. Da es bei Perl keine »Quellcodehinterlegung« (source code escrow) gibt, kann im Prinzip jeder einen Fehler beheben. Wenn Sie einen Fehler gefunden und verifiziert haben, hat ihn jemand anderes vielleicht schon behoben. Tausende von Leuten weltweit helfen mit, Perl zu pflegen.

Damit wollen wir jetzt nicht sagen, Perl sei voller Bugs. Aber es handelt sich um ein Programm, und jedes Programm hat mindestens einen Bug.

Um zu erkennen, warum es so nützlich ist, den Quellcode von Perl zu besitzen, stellen Sie sich einmal vor, Sie hätten es stattdessen mit einem gigantischen, mächtigen Konzern zu tun, dessen Besitzer ein Zillionär mit einem schlechten Haarschnitt ist. Von diesem hätten Sie nun eine Lizenz für eine Programmiersprache namens Forehead erworben. (Das ist natürlich alles hypothetisch. Jeder weiß, dass es keine Sprache mit dem Namen Forehead gibt.) Jetzt überlegen Sie einmal, was Sie tun können, wenn Sie einen Fehler in Forehead entdecken. Zuerst einmal können Sie diesen Fehler melden. Danach können Sie hoffen – hoffen, dass der Fehler behoben wird, hoffen, dass der Fehler *bald* behoben wird, und hoffen, dass der Konzern für die neue Version keinen zu hohen Preis verlangt. Sie können hoffen, dass in der neuen Version keine neuen Features vorkommen, die neue Fehler enthalten, und Sie können hoffen, dass der gigantische Konzern nicht durch ein Gerichtsverfahren wegen Ausnutzung der Monopolstellung in viele kleine Teile zerschlagen wird.

Perl hingegen wird zusammen mit seinem Quellcode verteilt. Tritt der seltene Fall auf, dass sich der Fehler nicht auf eine andere Art beseitigen lässt, können Sie notfalls einen oder zehn Programmierer engagieren und an die Arbeit gehen. Falls Sie in diesem Fall einen neuen Rechner kaufen, auf dem Perl noch nicht unterstützt wird, können Sie Ihre eigene Portierung schreiben. Und wenn Sie ein Feature brauchen, das es noch nicht gibt, wissen Sie ja nun, was zu tun ist.

Gibt es auch noch andere Arten von Support?

Sicher. Einer unserer Favoriten sind die »Perl Mongers«. Das ist ein weltweiter Zusammenschluss von Perl-Benutzergruppen. Weitere Informationen dazu finden Sie unter