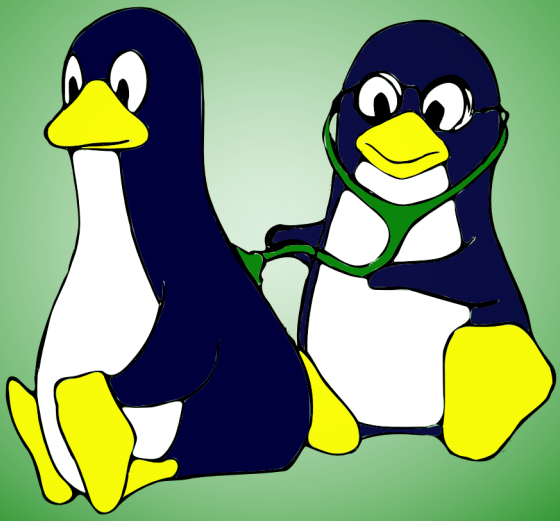
Fehlersuche

bei Linux-Servern und in IP-Netzwerken



Mathias Weidner

Fehlersuche bei Linux Servern und Netzwerken

Mathias Weidner

Dieses Buch können Sie hier kaufen
http://leanpub.com/troubleshoot-linux-server-and-networks

Diese Version wurde auf 2014-11-01 veröffentlicht

Das ist ein Leanpub-Buch. Leanpub bietet Autoren und Verlagen mit Hilfe des Lean-Publishing-Prozesses ganz neue Möglichkeiten des Publizierens. Lean Publishing bedeutet die permanente, iterative Veröffentlichung neuer Beta-Versionen eines E-Books unter der Zuhilfenahme schlanker Werkzeuge. Das Feedback der Erstleser hilft dem Autor bei der Finalisierung und der anschließenden Vermarktung des Buches. Lean Publishing unterstützt de Autor darin ein Buch zu schreiben, das auch gelesen wird.

Ebenfalls von Mathias Weidner

Linux headless

Using the Leanpub API with Perl

Inhaltsverzeichnis

Vorwort

Johannes Loxen sagte in seiner Keynote zum Frühjahrsfachgespräch 2012 der GUUG über den Unterschied von Closed-Source und Open-Source Software:

> "Wenn es nicht funktioniert, hörst Du bei Closed-Source auf zu denken und rufst den Hersteller-Support an. Bei Open-Source fängt das Denken dann erst richtig an".

Dieses Buch ist für alle, die nicht nur intensiv sondern vor allem auch strukturiert nachdenken wollen.

Ich habe bereits viel Zeit mit Fehlersuche verbracht. Schon als Kind beim Basteln, wenn etwas nicht funktionierte. Dann in meinem ersten Beruf, bei der Reparatur von Telefonen und Telefonanlagen, die damals noch mit Relais funktionierten, so dass ich ihnen im wahrsten Sinne des Wortes bei der Arbeit zusehen konnte. Nach dem Studium, als Softwareentwickler, beim Debuggen meiner oder fremder Programme. Später dann als Systemadministrator für UNIX und Netzwerke.

Dabei war ich vor allem auf drei Dinge angewiesen um zum Erfolg zu kommen. Das erste und wichtigste ist ein grundlegendes Verständnis der Materie und die Fähigkeit und Möglichkeit, weitere nötige Kenntnisse zu erwerben.

Das zweite ist eine einigermaßen strukturierte Vorgehensweise, die mich in den meisten Fällen schneller zum Ziel führt als andere Vorgehensweisen.

Die dritte Voraussetzung ist die eigene Einstellung zum Problem. Ich musste sehr oft und werde vermutlich noch oft die Erfahrung machen, dass ich mir selbst bei der Lösung eines Problems im Weg stehe:

- Sei es, dass ich mir bereits vor Kenntnis aller relevanten Tatsachen eine, eventuell falsche, Meinung bilde.
- Sei es, dass ich durch Druck von außen oder selbst erzeugten Druck besonders schnell sein will und dann beim oberflächlichen Hinsehen wichtige Details übersehe.
- Sei es, dass ich angebotene Hilfe nicht annehme, weil ich es selbst schaffen will oder so sehr im Problem gefangen bin, dass ich das Hilfsangebot nicht wahrnehme.
- Sei es, dass ich kurz vor dem Ziel aufgebe und einfach nicht den letzten Meter gehe.

Allein diese Hindernisse zu überwinden ist eine Aufgabe für ein ganzes Leben.

In diesem Buch bereite ich meine in mehr als zwanzig Jahren gesammelten Erfahrungen mit der Fehlersuche bei Linux-Servern und in IP-Netzwerken auf.

Ich habe diese beiden Themen kombiniert, weil einerseits Linux-Server fast immer über IP-Netzwerke angesprochen werden und daher auf ein gut funktionierendes Netzwerk angewiesen sind, andererseits Linux-Rechner für mich ideal zur Analyse von Problemen im Netzwerk geeignet sind.

Zwar benutze ich fast täglich Linux auf dem Desktop, allerdings verwende ich auch hier am häufigsten Terminal und Webbrowser, die in den meisten Fällen vorzüglich funktionieren, so dass ich kaum Gelegenheit hatte, Erfahrungen bei der Fehlersuche am Linux-Desktop zu sammeln. Aus diesem Grund lasse ich diesen Bereich außen vor.

Das gleiche gilt für IPv6 und WLAN, die ich zwar privat einsetze, aber noch nicht im beruflichen Umfeld, so dass meine Erfahrungen damit nicht ausreichen für eine fundierte und strukturierte Darstellung.

In der Keynote von Johannes Loxen ging es so weiter:

> "Manchmal wünscht man sich von den Closed-Source-Admins, dass Sie versuchen, weiter zu denken, indem sie sich der typischen Mittel der Open-Source-Welt bedienen, also Netzwerk-Sniffer oder Debuggern. Allerdings wünscht man sich auch umgekehrt manchmal, dass Open-Source-Admins nicht weiter denken - vor allem, wenn dies Zeit und Geld verbrennt und die entstehenden Kosten in keinem Verhältnis zum Wert des Problems stehen".

Hier gilt es eine Balance zu finden. Bei Problemen, die weit über meinen Horizont gehen, habe ich kein Problem damit, mir sofort Hilfe zu suchen. Einfache Probleme sind in der Regel im Nu geklärt. Anders ist es bei Problemen, die nur wenig schwieriger sind, als mein bisheriger Kenntnisstand. Das sind die Probleme, die am meisten Spaß machen, bei denen ich den Flow-Zustand erreiche, viel dabei lerne, besser werde und die Zeit vergesse. Ich nutze in diesem Fall gern Timeboxing, stelle einen Kurzzeitwecker auf eine knappe halbe Stunde und nutze die Unterbrechung, um ein wenig Bewegung zu bekommen und mich zu überzeugen, dass ich noch auf dem richtigen Weg bin.

Für wen ist dieses Buch

Die Zielgruppe sind angehende und gestandene Systemadministratoren für Linux und/oder IP-Netzwerke.

Ich setze grundlegende Kenntnisse der Kommandozeile voraus und dass der Leser weiß, wie er sie erreicht. Zum Teil setze ich für die Analyse von Problemen spezielle Software ein, die nicht auf jedem System installiert ist. Ich gehe davon aus, dass der Leser weiß, wie er Software auf seinem Rechner installiert.

Dieses Buch soll Anfängern in der Systemadministration über die ersten Hürden bei der Fehlersuche helfen und sie bei ihren ersten Schritten begleiten. Gestandene Administratoren werden vielleicht die eine oder andere Anregung entnehmen können.

Wie ist das Buch aufgebaut

Dieses Buch ist besteht aus drei Hauptteilen.

Teil eins beschäftigt sich mit allgemeinen Themen der Fehlersuche, die auch in anderen Bereichen nützlich sein können.

Im ersten Kapitel geht es um Methoden, Heuristiken und Modelle, also um strukturelle Hilfsmittel, die mir bei bestimmten Problemen der Fehlersuche weiterhelfen.

Im zweiten Kapitel geht es um die Herangehensweise an ein Problem. Wann und wie setze ich die Hilfsmittel aus dem vorigen Kapitel ein? Wie bereite ich mich auf die Fehlersuche vor? Wie kann ich generell besser bei der Fehlersuche werden?

Das dritte Kapitel geht auf die Nachbearbeitung eines Problems ein, die immer auch eine Vorbereitung auf das nächste Problem ist. Welche Erkenntnisse kann ich aus der letzten Fehlersuche ziehen? Kann ich diesen oder ähnliche Fehler in Zukunft vermeiden?

Die beiden folgenden Teile des Buches sind ähnlich strukturiert. Zuerst führe ich in Grundlagen ein, die als Hintergrundwissen für die folgenden Kapitel benötigt werden. Als

nächstes widme ich mich dem Totalausfall, dann Teilausfällen und was ich darunter verstehe und schließlich Problemen mit der Performance. Am Schluß der Teile zwei und drei stelle ich jeweils einige Programme vor, die ich bei der Fehlersuche häufig einsetze.

Teil zwei beschäftigt sich mit lokalen Problemen einzelner Linux-Server.

Das erste Kapitel dieses Teiles stellt Grundlagenwissen für die lokale Fehlersuche auf Linux-Systemen vor.

Im nächsten Kapitel geht es um Totalausfälle. Das können Hardwarefehler sein, Bootprobleme, ein Rechner der scheinbar oder wirklich nicht mehr reagiert oder ein Rechner, der durch Überlast so langsam geworden ist, dass ich nicht einmal mehr eine Shell zur Diagnose bekommen kann.

Das folgende Kapitel behandelt Teilausfälle bei Linux-Servern. Diese äußern sich meistens darin, dass Dienste nicht starten oder nicht korrekt arbeiten, oder dass Dateien nicht gespeichert werden können.

Im anschließenden Kapitel geht es um die Performance von Linux-Servern. Alles funktioniert, aber richtig schnell ist es nicht. Hier geht es darum, herauszufinden, was der Rechner macht, wenn es so lange dauert und wie man das beschleunigen kann.

Das letzte Kapitel des zweiten Teils schließlich stellt Werkzeuge vor, die ich häufig einsetze. Zusätzlich zu den wichtigsten Optionen zeige ich einige Anwendungsfälle oder verweise auf Kapitel, in denen das Programm verwendet wird.

Der **dritte Teil** des Buches behandelt die Fehlersuche im IP-Netzwerk.

Das erste Kapitel dieses Teiles geht auf Totalausfälle im Netz ein. Ein Totalausfall im Netzwerk ist für mich gegeben,

wenn ein Rechner überhaupt keinen Kontakt zum Netzwerk bekommt oder, wenn in einem Netzsegment die Basisdienste so weit ausgefallen sind, dass die Rechner ihren Netzanschluß nicht produktiv nutzen können.

Im nächsten Kapitel gehe ich auf Teilausfälle im Netzwerk ein. Dabei unterscheide ich, ob ganze Netzsegmente nicht erreichbar sind oder nur einzelne Dienste.

Danach geht es um Performance-Probleme im Netzwerk. Wie analysiere ich diese? Welche möglichen Abhilfen gibt es?

Und im letzten Kapitel stelle ich wiederum Werkzeuge vor, die mir bei der Fehlersuche im Netz helfen.

Übungen

Ich habe mir in diesem Buch große Mühe gegeben, eine Hilfe bei der Fehlersuche zu bieten. Dennoch kann es keine eigene Erfahrungen ersetzen. Insbesondere, wenn man vor einem schwierigen neuen Problem steht, ist es hilfreich, wenn die aus diesem Buch gewonnenen Erkenntnisse und die eigenen Erfahrungen abrufbereit sind. Aus diesem Grund habe ich einige Übungen im Buch eingestreut. Bei diesen kommt es nicht immer auf das konkrete Ergebnis an, sondern dass man sie in einer freien Minute ausführt und beobachtet, was passiert.

Übungen sind mit diesem Symbol gekennzeichnet.

Zur Schreibweise

Für Programmbeispiele und Eingaben auf der Kommandozeile verwende ich eine `dicktengleiche Schrift`, diese nehme ich

auch im Fließtext, wenn ich Optionen oder Befehle wortgetreu verwende. Bei Kommandozeilen stehen Begriffe, die mit $ eingeleitet werden, wie zum Beispiel `$gateway`, für variable Angaben, die je nach Kontext ersetzt werden müssen.

Ansonsten verwende ich einen *kursiven Font* für Hervorhebungen.

Danksagung

Ohne die vielen Autoren der freien Software, die ich in meiner täglichen Arbeit nutze, wäre dieses Buch nicht möglich gewesen. Ich würde meine Arbeit auch nicht so ausüben können, wie ich es heute tue.

Auch der Gemeinschaft, die das Internet möglich gemacht hat und die es täglich aufs Neue mit Ideen, Anregungen und Problemlösungen füllt, die es nur zur rechten Zeit zu finden gilt, hätte ich viele der hier vorgestellten Kenntnisse selbst nicht erwerben können.

Mein besonderer Dank gebührt Denise Betzendörfer und Stefan Naumann, die mir etliche Hinweise zu diesem Buch gaben und mich auf Dinge aufmerksam machten, die ich im Laufe der langen Beschäftigung mit dem Text gar nicht mehr wahrgenommen hatte. Auch Knut Schmädt, hat sich die Mühe gemacht, eine frühe Version dieses Buches zu lesen und gab mir Feedback dazu.

I Allgemeine Themen

1. Methoden, Heuristiken und Modelle

Um eine gemeinsame Sprache zu finden, lege ich zunächst kurz dar, was ich in diesem Buch unter *Methoden*, *Heuristiken* und *Modellen* verstehe.

Methode

Spreche ich von einer Methode, meine ich ein auf Regeln aufbauendes Verfahren um konkrete Erkenntnisse oder praktische Ergebnisse zu erlangen, ähnlich einem Algorithmus. Eine Methode kann ich nur unter bestimmten Voraussetzungen sinnvoll einsetzen. Sie liefert mir dann jedoch klar definierte Ergebnisse beziehungsweise Erkenntnisse.

Heuristik

Das Lexikon online für Psychologie und Pädagogik[1] beschreibt eine Heuristik wie folgt:

"Als Heuristik oder heuristisches Vorgehen bezeichnet man in der Psychologie eine einfache Denkstrategie für effizientere Urteile und Problemlösungen, die meist schneller, aber auch fehleranfälliger ist als ein Algorithmus."

Diese Definition kommt dem recht nahe, wie ich Heuristik in diesem Buch verwende. Im Gegensatz zur Methode bekomme ich keine klar definierten Ergebnisse, sondern bestenfalls Anhaltspunkte, die mir eine Richtung für die weitere Untersuchung anzeigen.

Modell

Unter einem Modell verstehe ich ein vereinfachtes Abbild der Wirklichkeit, dass bestimmte, für die Betrachtung wesentliche Merkmale korrekt abbildet und andere ignoriert. Ein und dieselbe Situation kann ich mit verschiedenen Modellen beschreiben. Welches Modell ich wähle, hängt davon ab, welche Eigenschaften ich untersuchen will.

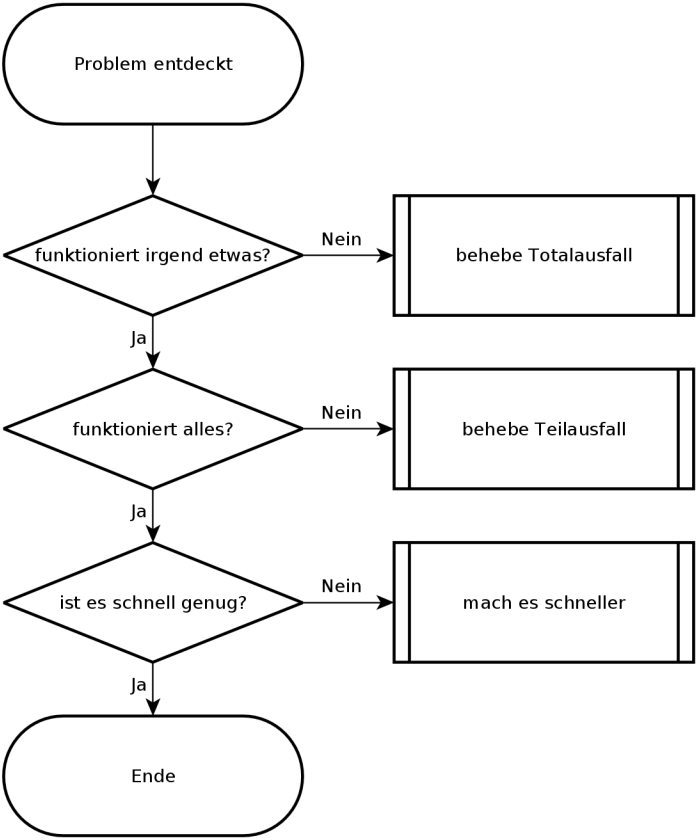
Bei der Auswahl oder Gestaltung eines Modells gilt es zwei Gefahren zu vermeiden. Bilde ich ein unwesentliches Merkmal mit ab, dann wird das Modell zu komplex und macht es schwerer die wesentlichen Punkte zu erkennen. Lasse ich wesentlichen Aspekte aus, ist es möglich, dass ich die falschen Schlüsse ziehe.

Entscheidungsbaum

Wenn ich auf ein neues Problem treffe, versuche ich es so schnell wie möglich zu charakterisieren, um die nächsten Schritte zu seiner Behebung herauszufinden. Dabei helfen mir Entscheidungsbäume. Programmierer kennen so etwas auch als Programmablaufplan.

Mein grundlegender Entscheidungsbaum bei der Fehlersuche sieht so aus, wie im folgenden Bild. Grundsätzlich werde ich nur tätig, wenn eine der Fragen mit nein beantwortet wird. Damit decke ich fast alle Probleme ab, lediglich intermittierende Probleme erfasst der Entscheidungsbaum nicht.

[1] http://lexikon.stangl.eu/1963/heuristik/



Allgemeiner Entscheidungsbaum

Die erste Frage geht danach, ob überhaupt noch etwas funktioniert oder ob es sich um einen Totalausfall handelt. Diese Frage erscheint vielleicht trivial, aber das ist letztendlich der Zweck eines Entscheidungsbaumes: das die richtigen Fragen im richtigen Moment gestellt werden. Manchmal berichtet ein

Anwender, dass eine Funktion eines Programmes nicht funktioniert und irgendwann stellt sich heraus, dass der ganze Rechner eingefroren ist und zwar noch den letzten Bildschirm zeigt, aber weder auf Tastatur, Maus noch Netzwerkzugriffe reagiert. Ein anderes Mal kommt die Meldung, dass das Internet nicht geht (Totalausfall) und auf die Bitte, ein oder zwei andere Websites zu besuchen, stellt sich heraus, dass nur die Startseite des Browsers betroffen ist. Darum versuche ich mit den ersten Fragen herauszufinden, ob es sich um einen Totalausfall handelt, den ich anders behandeln muß als einen Teilausfall.

Liegt kein Totalausfall vor, frage ich als nächstes, ob alle für das Problem relevanten Dienste funktionieren. Es erfordert schon einige Detailkenntnisse zur Problemzone, um zu entscheiden, ob ein Dienst für das Problem relevant ist oder nicht. Im Zweifelsfall kontrolliere ich lieber einen Dienst mehr. Hier geht es vor allem darum, einen Überblick zu bekommen, was funktioniert und was nicht und sich dann über Abhängigkeiten der Teilsysteme an den oder die Urheber des Problems heranzutasten. Dabei gilt es immer im Hinterkopf zu behalten, dass es zwar meist einen konkreten Auslöser für ein Problem gibt, aber oft mehrere Ursachen. Eine Möglichkeit, diese Frage zu beantworten, ist, verschiedene Funktionen einer Software auszuprobieren, verschiedene Netzdienste und Netzziele zu testen. Hierbei kann ein Monitoringsystem wie Nagios gute Dienste leisten, wenn es entsprechend aufgesetzt ist.

Funktionieren alle notwendigen Dienste prinzipiell, kann ich die nächste Frage stellen: ob es schnell genug ist. Diese Frage ist nicht leicht zu beantworten, da jeder seine eigene Vorstellung von schnell genug hat. Gibt es SLA, können diese vielleicht bei der Beantwortung der Frage helfen. Bei nicht interaktiven Aufgaben, wie Datensicherungen oder Batchjobs kann ich die

Gesamtlaufzeit betrachten und an Hand dieser entscheiden, ob es schnell genug ist, oder nicht. Bei Dateiübertragungen kann ich an Hand von Datenübertragungsrate, Netzauslastung und Übertragungsdauer überschlagen, ob es Performanceprobleme gibt oder nicht. Bei interaktiven Programmen oder Netzwerkdiensten zählt meist nur die Antwortzeit des Systems, die im Bereich von Sekundenbruchteilen liegen sollte. Komme ich zu dem Schluss, dass es sich um ein Performanceproblem handelt, gehe ich dieses an. Anderenfalls begründe ich, warum es sich meiner Meinung nach um kein Performanceproblem handelt. Dabei kann mir eine Baseline helfen.

Alles in allem habe ich mit den drei Fragen dieses Entscheidungsbaumes eine Richtschnur, die mir hilft, ein Problem herunterzubrechen und mich dem wichtigsten Bereich zu widmen, bevor ich mich in den Details verliere.

Dabei muss ich den Entscheidungsbaum nicht zwangsläufig von oben nach unten verwenden. Wenn mir ein Netzwerk-Performanceproblem gemeldet wird, überzeuge ich mich zunächst davon, dass alle für das Problem relevanten Dienste auch funktionieren. Ich gehe in diesem Fall von unten - dem gemeldeten Performanceproblem - einen Schritt nach oben um sicher zu sein, dass meine folgenden Überlegungen auf einer gesicherten Basis stehen. Zum Beispiel kann ein ausgefallener DNS-Server durch Redundanz zwar kompensiert werden, aber trotzdem zu Verzögerungen durch Timeouts führen, die dann als Performanceproblem wahrgenommen werden.

Außer den drei Hauptfragen gibt es eine vierte Frage, die ich ständig im Hinterkopf behalten muss. Das ist die Frage nach intermittierenden Fehlern und nach der Reproduzierbarkeit des Problems, beziehungsweise meiner Beobachtungen. Habe ich es mit intermittierenden Fehlern zu tun, kann ich nicht

mit Sicherheit sagen, ob das Problem wirklich gelöst ist. Nicht einmal, ob meine bisherigen Überlegungen überhaupt zutreffen. Stattdessen hatte ich es vielleicht gerade mit einer komplikationsfreien Zeit zu tun und kurz danach kommt das Problem wieder. Bei intermittierenden Problemen bleibt mir nur, Daten zu sammeln und über Korrelation eine Idee zu bekommen, was das Problem auslösen könnte. Jede Idee, die mir dazu einfällt, muss ich dahingehend prüfen, ob sie das Problem stabil vorhersagbar macht oder nicht.

Bisektion

Die Bisektion, auch Intervallhalbierungsverfahren genannt, verwende ich um eine Fehlerstelle, die in einem Intervall auftritt, schneller zu finden. Das Intervall kann örtlich sein - zum Beispiel bei einer Datenübertragung über mehrere Hops - oder zeitlich - zum Beispiel, wenn sich der Fehler in Logfiles zeigt und ich den Zeitpunkt des ersten Auftretens über viele Logdateien hinweg suche.

Das Verfahren selbst ist sehr einfach. Anstatt das Intervall sequentiell in einer Richtung abzusuchen, halbiere ich es und untersuche von den entstehenden Teilintervallen dasjenige, dessen Grenzen sich unterscheiden. Bei diesem fahre ich mit der Bisektion fort.

Stellen wir uns einen Stapel Papier vor, der auf dem Tisch liegt und auf den ein Tropfen einer sehr aggressiven Flüssigkeit gefallen ist. Um die guten von den beschädigten Blättern zu trennen, kann ich die beschädigten Blätte oben entfernen. Alternativ teile ich den Stapel in der Mitte auf und schaue nach, ob die Blätter unten beschädigt sind. Sind sie beschädigt, werfe ich alle abgehobenen Blätter weg und teile den verbliebenen Stapel

wieder bei der Hälfte. Sind sie nicht beschädigt, sichere ich die untere Hälfte und teile die oberen Blätter wieder bei der Hälfte. So fahre ich fort, bis ich zum ersten unbeschädigten Blatt komme. In den meisten Fällen ist dieses Verfahren schneller, als wenn ich von oben nach unten durchgeblättert hätte.

Wenn ich zum Beispiel einen Rechner in einem weit entfernten Netz zwar via Ping erreichen kann, aber nicht mit Port 25, dann kann ich zunächst untersuchen, ob Datenpakete zu Port 25 beim Sender abgehen und ob sie beim Empfänger ankommen. Kann ich die Datenpakete beim Sender nachweisen, aber beim Empfänger nicht, dann suche ich etwa auf der Hälfte der Strecke, ob diese Datenpakete nachzuweisen sind. Je nachdem, ob sie dort auftreten oder nicht, halbiere ich danach die Strecke zum Empfänger oder zum Sender.

Die Bisektion ist nur dann langsamer als die sequentielle Suche, wenn die gesuchte Stelle sich gleich am Anfang des Intervalls befindet.

Statistische Verfahren, Korrelation

Statistische Verfahren liefern keine Belege für Ursache-Wirkung-Beziehungen. Sie können mir nur helfen, Ansatzpunkte für Hypothesen zu finden, die ich mit gezielten Tests bestätigen oder widerlegen kann. Dabei muss ich mir vorher im klaren sein, ob ein bestimmter Test darauf angelegt ist, eine Hypothese zu widerlegen und damit von der weiteren Betrachtung auszuschließen oder die Hypothese zu bestätigen, so dass ich davon meine weiteren Handlungen leiten lassen kann.

Ein statistisches Verfahren ist die Korrelation. Mit dieser kann ich eine Beziehung zwischen zwei oder mehreren Merkmalen, Ereignissen oder Zuständen beschreiben. Dabei muss ich im-

mer beachten, dass diese Beziehung kausal sein kann aber nicht muss. Im mathematischen Sinne beschreibt die Korrelation einen statistischen Zusammenhang im Gegensatz zur Proportionalität, die einen festes Verhältnis beschreibt.

Will ich den Vierfelder-Korrelationskoeffizient ϕ für ein Merkmal und den Fehler ermitteln, stelle ich eine Kontingenztafel auf, in der ich die gemeinsame Häufigkeit der Merkmale eintrage.

	kein Fehler	**Fehler**
Merkmal trifft zu	A	B
Merkmal trifft nicht zu	C	D

A, B, C, D stehen für die Anzahl der Ereignisse, bei den der Fehler auftrat beziehungsweise nicht auftrat und das Merkmal zutraf oder nicht. Als Werte für A, B, C und D kann ich Zeiträume summieren, oder zu regelmäßigen Zeitpunkten ermitteln, welcher Fall zutrifft und die entsprechende Variable hochzählen.

Der Phi-Koeffizient ermittelt sich dann aus:

$$\phi = \frac{AD - BC}{\sqrt{(A+B)(C+D)(A+C)(B+D)}}$$

Im einfachsten Fall kann ich eine Korrelation selbst sehen, zum Beispiel, wenn immer die Verbindung zum Netz A wegfällt, sobald ich die Verbindung zum Netz B umschalte. Das sagt zwar nichts aus über die Ursache, es gibt mir aber die Möglichkeit, das Problem kurzfristig temporär aus dem Weg zu schaffen und, wenn die Zeit geeignet ist, das Problem hervorzurufen, um es gründlich zu studieren.

Intermittierende Probleme, die sich dem Zugang entziehen, sind eine Klasse von Problemen, bei welchen Korrelationen

hilfreich sind. Bei diesen Problemen hilft mir am Anfang nur, so viele Daten wie möglich über das betroffene Gesamtsystem zu sammeln. Bei der Analyse dieser Daten hilft mir dann die Kovarianzanalyse, das heisst, ich vergleiche jeweils paarweise verschiedene beobachtete Systemvariablen und ermittle den Korrelationskoeffizienten.

Wichtig dabei ist, dass ich immer die zusammengehörigen Daten in Beziehung setze. Da ich den Zusammenhang meist über die Uhrzeit herstelle, sorge ich also entweder im Vorfeld für gleichlaufende Uhren, oder ich muss den Zeitoffset bestimmen und heraus rechnen.

Ein Programm, das sehr hilfreich für die Kovarianzanalyse ist, wurde in [WSA2011] beschrieben.

Wichtig bei Korrelationsanalysen ist, sich vor Augen zu halten, dass unabhängige Merkmale eine Kovarianz nahe 0 haben.

Abkürzungen und Umwege

Im Laufe der Zeit, wenn ich Erfahrungen mit den Systemen gesammelt habe, kann ich bei etlichen Fehlern intuitiv sagen, woran es liegt. Das erspart mir erhebliche Zeit bei der Fehlersuche. Ich muss jedoch immer im Auge behalten, ob meine Annahmen auf einer realen Grundlage beruhen, oder ob mich meine Intuition hier in die Irre führt.

Das heißt, dass ich jedes Mal, wenn ich eine Abkürzung nehme, mich vergewissern muss, dass die Voraussetzungen dafür stimmen.

Abkürzungen

Eine Möglichkeit, die Fehlersuche abzukürzen, ist durch simple Korrelation. Bei der Aufnahme des Fehlers frage ich, wann der Fehler das erste Mal bemerkt wurde und wann es das letzte Mal funktioniert hatte. Im Rahmen der Fehlersuche, schaue ich nach, was am System in diesem Zeitraum geändert wurde und überlege für jede Änderung, ob diese den beschriebenen Fehler hervorrufen könnte. Dazu benötige ich natürlich eine aussagefähige Protokollierung der Änderungen am System. Und der Fehler sollte so schnell wie möglich gemeldet werden, damit der Zeitraum, den ich in Betracht ziehe, nicht zu groß wird.

Eine weitere Abkürzung passiert implizit, wenn ich ähnliche Probleme bereits hatte. Dann konzentriere ich mich automatisch auf die Punkte, die beim letzten Mal zur Lösung geführt hatten. Natürlich muss ich mich fragen, ob die gleichen Voraussetzungen zutreffen.

Umwege

Und damit bin ich schon bei den Umwegen, weil ich, anstatt sofort auf die Lösung zuzustürmen, erst einmal die genauen Umstände prüfe.

Man sagt, Umwege verbessern die Ortskenntnis. Ich interpretiere das so, dass ein Umweg eine Investition in die Zukunft ist, wenn ich damit, neben dem eigentlichen Ergebnis, gleichzeitig eine Annahme über das Gesamtsystem überprüfen kann. Dann ist der Zeitverlust durch den Umweg der Preis für eine Erkenntnis über das System.

Probe und Gegenprobe

Jeder Test, der erfolgreich ist, sollte durch eine geeignete Gegenprobe evaluiert werden, die bestätigt, dass der Test auch versagen kann. Das gleiche gilt für fehlgeschlagene Tests. Diese müssen evaluiert werden, ob sie funktionieren können.

Ist ein Verbindungsversuch auf einem Rechner fehlgeschlagen, versuche ich den gleichen Versuch auf einem anderen Rechner, um nachzuweisen, dass er hätte funktionieren können. Das hat mich schon oft davor bewahrt, einen Netzwerkfehler zu vermuten, wenn lediglich ein Paketfilter die Verbindung unterbunden hatte.

2. Herangehen

Neben dem geistigen Rüstzeug, das ich im vorigen Kapitel besprochen habe, bestimmt die Art und Weise, wie ich an ein Problem herangehe, den Erfolg.

Natürlich gibt es kein Patentrezept, das ich unverändert auf alle Probleme anwenden kann. Je nach Wichtigkeit und Dringlichkeit werde ich vielleicht in einem Fall auf eine schnelle, dafür nicht sehr gründliche Lösung hinarbeiten und mir in einem anderen Fall die Zeit nehmen, ein Problem ein für alle Mal aus der Welt zu schaffen.

3. Nachbereitung

Manchen Leuten scheint es gegeben, häufiger als andere richtige Voraussagen zu machen und damit Probleme schneller zu lösen. Beobachtet man diese Leute genauer, stellt man fest, das sie keine Voraussagen über die Zukunft machen, sondern aus der Vergangenheit. Der Volksmund nennt dieses Phänomen auch Erfahrung.

Nach dem letzten Fehler wird es einen nächsten geben. Darum ist die Nachbereitung immer auch eine Vorbereitung auf kommende Fehlersuchen.

Manchmal folgt der nächste Fehler so schnell, dass kaum Zeit für die Rückschau bleibt. Tritt dieser Umstand häufiger auf, ist das ein Zeichen dafür, dass eine Rückschau umso nötiger ist. Dann kann ich mir wenigstens eine Liste oder Tabelle anlegen, in der ich die aufgetretenen Fehler zusammen mit ihrem Schweregrad festhalte. So habe ich später eine Entscheidungsgrundlage, wenn Zeit da ist, sich einem Thema intensiver zu widmen.

II Lokale Probleme

4. Grundlagen Linux

Ein paar Grundlagen benötige ich immer wieder, wenn ich die Vorgänge in einem Linux-System verstehen will.

Von diesen halte ich im Zusammenhang mit Fehlersuche für besonders wichtig:

- die Bedeutung von Dateien und Verzeichnissen bei UNIX und Linux
- das UNIX-Prozessmodell
- die Schnittstellen über die ein Prozess mit seiner Umgebung interagiert

Darauf gehe ich in den folgenden Abschnitten ein.

5. Totalausfall

Ein Totalausfall bedeutet für mich zuallererst: der Rechner selbst ist keine Hilfe bei der Fehlersuche. Das muss nicht heißen, dass der Rechner überhaupt nicht mehr reagiert, aber auf jeden Fall stehen mir nicht alle Werkzeuge zur Verfügung, die ich sonst zur Analyse einsetzen könnte.

Ich betrachte die folgenden Probleme als Totalausfall eines Rechners.

- Hardwarefehler, der Rechner reagiert überhaupt nicht beim Einschalten. Abgesehen von dem Tipp, den Stromanschluss zu überprüfen, gehe ich darauf hier nicht weiter ein.
- Boot-Probleme, zwar passiert etwas, die Hardware scheint größtenteils in Ordnung, aber das Betriebssystem wird nicht gestartet. Ich analysiere den Bootprozess Schritt für Schritt, beobachte, wie weit das System kommt und versuche die Ursache zu ermitteln.
- Einfrieren, der Rechner war normal gestartet, hat eine Zeit lang gearbeitet, aber jetzt reagiert er nicht mehr. Ich versuche zunächst den Rechner mit Magic SysRequest geordnet neu zu starten und nachträglich zu analysieren. Falls möglich, werte ich die mit SysRequest gewonnenen Erkenntnisse aus.
- Überlast, Swapping, Thrashing. Ich sehe oder höre, dass der Rechner intensiv arbeitet. Trotzdem reagiert er sehr langsam. Auf Grund von Timeouts kann ich mich nicht einmal mehr anmelden, in einer Shell bekomme ich kein

Programm gestartet. Wenn ich die Last nicht auf anderem Weg vom Rechner nehmen kann, bleibt mir nur, den Rechner neu zu starten. Anschließend versuche ich zu ermitteln, was die Überlast ausgelöst hatte.

- Ausfälle bei virtuellen Maschinen. Das sind die gleichen Fehler wie bei "echten" Maschinen aus Blech. Hinzu kommen Probleme mit der Virtualisierungsschicht. Im Gegenzug bekomme ich mehr Diagnosemöglichkeiten an die Hand.

  Neben dem Totalausfall des Gastsystems gibt es noch die Möglichkeit, das das Wirtssystem ausgefallen ist. Diesen Fehler behandle ich, wie oben beschrieben.

  Bei den Gastsystemen sind meine Optionen abhängig von der verwendeten Virtualisierungslösung und deren Werkzeugen zur Diagnose. Manchmal kann ich einfach die Festplattenpartitionen des Gastsystems im Hostsystem oder einem anderen Gastsystem einhängen und dort untersuchen. Damit stehen mir zusätzliche Werkzeuge zur Verfügung.

6. Partieller Ausfall

Bei einem teilweisen Ausfall auf einem Server sind nur einzelne Dienste des Systems betroffen. Dann steht mir, im Gegensatz zum Totalausfall, meist die gesamte Vielfalt an Werkzeugen zur Diagnose zur Verfügung.

Bevor ich mich wegen eines Problems an einem Server anmelde, habe ich bereits einige Informationen gesammelt.

- Wie äußert sich das Problem?
- Wann wurde es zuerst bemerkt?
- Ist es reproduzierbar?
- Gibt es zeitliche Muster?
- Sind alle betroffen oder nur ein Teil der Nutzer?

Damit habe ich schon eine gewisse Vorstellung, wonach ich schauen werde, wenn ich mich anmelde.

7. Performance des Rechners

Der Rechner funktioniert zwar, aber alles erscheint zäh und in Zeitlupe abzulaufen. Jobs brauchen eine gefühlte Ewigkeit, die Tastatur scheint mit 8 Baud angeschlossen. Irgendwo sind Engpässe, aber wo?

Oft ist es einfach nur eine Überlast, das heißt, das System erhält temporär mehr Aufgaben, als es bewältigen kann. Dann werde ich zunächst alles tun, um die Last zu reduzieren. Anschließend mache ich mir Gedanken, wie ich die zu erwartende Last bewältigen kann.

Die drei wesentlichen Engpässe beim Betrieb eines Rechners sind CPU, Hauptspeicher und I/O. Welcher von diesen drei im Moment gerade den Betrieb hemmt, bekomme ich sehr schnell mit dem Befehl `vmstat` heraus. Damit weiß ich allerdings erst, wo ich suchen will, die eigentliche Ursache muss ich noch ermitteln.

Mit dem Programm `top` kann ich etwas genauer hinschauen. Je nach dem, was ich mit `vmstat` herausbekommen habe, kann ich interaktiv mit `P` die Prozesse nach prozentualer Nutzung der CPU und mit `M` nach der prozentualen Nutzung des Speichers sortieren lassen. Mit `1` zeigt das Programm in den Kopfzeilen für jede einzelne CPU an, wie sie ihre Zeit verbringt. Mit `f`, gefolgt von `j` und `<Enter>` zeigt das Programm in einer zusätzlichen Spalte, welcher Prozess auf welcher CPU läuft.

Wenn ich es ganz genau wissen will, erstelle ich ein Lastprofil meines Systems mit Accounting-Software wie sysstat. Zwar be-

lastet diese das System zusätzlich, doch ich kann die Ergebnisse meiner Analyse dann nutzen, um gezielt in neue Hardware zu investieren, ein oder mehrere leistungsstärkere Prozessoren, mehr RAM, schnellere Festplatten und Netzwerkkarten. Oder ich verteile die Last auf mehrere Systeme.

Als Einführung in das Gebiet empfehle ich Loukides1996, das mittlerweile in neuer Auflage erschien.

8. Werkzeuge zur lokalen Fehlersuche

Linux stellt mir eine Unmenge an Werkzeugen für die Fehlersuche zur Verfügung. Etliche davon kommen mir sowohl bei Total- als auch bei Partialausfällen zu gute. Andere bei Performanceproblemen. Einige sind so nützlich, dass ich sie immer wieder bei den unterschiedlichsten Problemen einsetze.

Da es mir schwerfällt, die einzelnen Werkzeuge bestimmten Kategorien zuzuordnen, stelle ich sie nachfolgend in alphabetischer Reihenfolge vor. Das erleichtert zumindest das Wiederfinden, wenn ich nur etwas nachschlagen möchte.

III Netzwerkprobleme

9. Grundlagen IP-Netzwerke

Ein paar Grundlagen benötige ich immer wieder, wenn ich die Vorgänge in einem IP-Netzwerk verstehen will.

Insbesondere halte ich bei der Fehlersuche die Kenntnis folgender Themen für wichtig:

- Das OSI-Referenzmodell, mit dessen Hilfe ich die Position eines Protokolls im Protokoll-Stack bestimmen und seine Bedeutung einschätzen kann.
- ARP, weil es die Lücke schließt zwischen dem Internet-Protokoll und der Netzzugangsschicht, wie zum Beispiel Ethernet.
- Die Bestimmung von IPv4-Netzmasken.
- Die Zustände von TCP-Verbindungen.
- Zwei wesentliche Algorithmen bei Routing-Protokollen: Distanz-Vektor und Link-State.

Auf diese Themen gehe ich in den folgenden Abschnitten ein, damit wir sie im positiven Sinne vergessen können.

10. Totalausfall des Netzwerks

Habe ich an einem Rechner überhaupt keine Verbindung zum Netz, oder kann ich ein ganzes Netzsegment nicht erreichen, spreche ich von einem Totalausfall.

Dabei unterscheide ich zwei Arten, je nachdem, von wo ich das Problem betrachte. Untersuche ich es von einem Rechner, der überhaupt keine Verbindung hat, dann sehe ich das Problem von innen, weil ich versuche, mit dem Rechner innerhalb eines Segments eine Verbindung zu bekommen. Kann ich ein ganzes Netzsegment nicht mehr erreichen, dann betrachte ich das Problem von außen, da ich mich in einem anderen Segment befinde und andere Rechner erreichen kann, nur dieses Netzsegment und die Rechner darin nicht.

Bei der Betrachtung von innen, überprüfe ich die Schichten des OSI-Modells von unten nach oben. Das heißt, ich fange bei der Kabelverbindung zum Switch an und arbeite mich zur IP-Konfiguration hoch, bis ich Kontakt zu mindestens einem anderen Rechner bekomme.

Bei der Betrachtung von außen muss ich die Topologie des Netzwerks kennen. Dann überprüfe ich den Weg zum nicht erreichbaren Segment und kontrolliere auf dem letzten erreichbaren Gateway dorthin die physikalische Verbindung und die Routen. Auf diese Weise arbeite ich mich bis zum ausgefallenen Netzsegment vor.

11. Partieller Ausfall des Netzes

Ich spreche von einem Teilausfall im Netz, wenn wenigstens die Grundfunktionen bis OSI-Schicht 4 funktionieren und ich alle Netzsegmente erreichen kann. In diesem Fall kann ich das Netz selbst zur Fehlersuche einsetzen und brauche mir nicht die Schuhsohlen ablaufen, um vor Ort nach dem Rechten zu sehen oder via Telefon zu debuggen.

Was also kennzeichnet einen Teilausfall im Netz?

Da ist zunächst der Ausfall essentieller Dienste im Netz, wie DHCP, DNS, NTP, Kerberos und weiterer. Der Ausfall einiger dieser Dienste kann Auswirkungen haben, die die Kunden als Totalausfall wahrnehmen. Das muss ich bereits bei der Aufnahme des Problems berücksichtigen. Für den Kunden macht es keinen Unterschied, ob er das Netz nicht nutzen kann, weil ein Kabel unterbrochen ist, oder weil sein Rechner keine IP-Adresse zugewiesen bekam. Für mich, der ich den Fehler beseitigen will, ist der Unterschied durchaus von Belang.

Weiterhin betrachte ich als Teilausfall, wenn ich einzelne Rechner oder einzelne Dienste auf bestimmten Rechnern nicht erreichen kann. Im Laufe der Untersuchung wird daraus vielleicht ein Total- oder Teilausfall des Rechners; dann behandle ich das, wie in Teil 2 dieses Buches beschrieben.

Als drittes zähle ich partielle Einschränkungen von funktionierenden Diensten dazu. Darunter verstehe ich zum Beispiel Mengenbeschränkungen beim Upload auf Webserver oder beim

Versand von E-Mail, Zugriffsbeschränkungen für einzelne Netzsegmente und ähnliches. Diese lassen sich in vielen Fällen auf die Konfiguration des betreffenden Dienstes zurückführen.

Schließlich besteht die Möglichkeit, dass ein Dienst absichtlich gestört wird. Angriffe von Dritten und deren Abwehr sind jedoch nicht Thema dieses Buches und werden höchstens am Rande behandelt.

Bei der Behebung des Teilausfalls lasse ich mich wieder vom grundlegenden Entscheidungsbaum leiten. Das heißt, ich versuche zuerst festzustellen, ob überhaupt etwas funktioniert, also ob ich den Server, auf dem der Dienst läuft, erreichen kann. Eventuell muss ich hier noch einen Schritt zurück und sehen, ob ich andere Rechner im gleichen Netzsegment erreichen kann. Kann ich den Server erreichen, untersuche ich, ob alles funktioniert. Bekomme ich Kontakt zum Dienst, antwortet er korrekt laut Protokoll? Schließlich untersuche ich, ob der Dienst schnell genug antwortet.

12. Netzwerkperformance

Manchmal scheint alles in Ordnung zu sein und trotzdem sind die Kunden nicht zufrieden. Performance ist ein heikles Thema, weil jeder seine eigene Vorstellung davon hat, was ausreichende Performance ist. So heikel, dass mancher sein Heil in immer "dickeren" Leitungen mit immer höherer Datenübertragungsrate sucht, um sich nicht näher mit dem eigentlichen Problem beschäftigen zu müssen.

Was die Sache so schwierig macht, ist, dass ein Performance Problem sehr viele Aspekte haben kann.

Da gibt es mitunter eine Diskrepanz zwischen wahrgenommener Performance und realer Performance. Ein Kunde ruft an, weil "sein E-Mail-Programm nicht funktioniert". Er schreibt die E-Mail, drückt auf Senden und dann friert das Programm ein. Bei genauer Betrachtung stellt sich heraus, dass da ein Anhang von mehreren MB an der E-Mail hängt und die Leitung beim Upload nur eine maximale Datenübertragungsrate von ein paar hundert kBit/s hat. Ein kurzes Nachrechnen von verfügbarer Datenübertragungsrate und zu sendenden Daten ergibt bereits eine Dauer von mehreren Minuten bei alleiniger Nutzung der Leitung durch diesen Upload.

Aber auch reale Performance-Probleme können vielfältige Ursachen haben, die es einzugrenzen gilt. Oft genug sind diese Probleme intermittierend, dass heißt, in einem Moment da und im nächsten weg. Darum ist die Fehlersuche bei Problemen mit der Performance fast immer mit dem Sammeln von Daten verbunden.

Wenn ich ein Performance-Problem untersuche, beginne ich zunächst die Charakteristiken des Problems so genau wie möglich zu bestimmen.

Dann versuche ich das Problem grob zu klassifizieren:

- Ist es ein normales Problem, weil zu viele Daten gesendet werden sollen?
- Ist es vielleicht ein DNS-Problem mit Verzögerung am Anfang?
- Gibt es Probleme mit dem Durchsatz bei längeren Downloads?
- Liegt das Problem eher an der Latenz bei interaktiven Programmen?

Auch die Art des verwendeten Protokolls kann Hinweise auf mögliche Probleme geben. So gehen bei datagrammorientierten Protokollen wie UDP Daten einfach verloren, während bei datenflussorientierten Protokollen wie TCP Zeit verloren geht, während das Protokoll die Daten nochmals sendet.

Bei der Klassifizierung hilft es mir, wenn ich bereits bei der Aufnahme des Problems nach Anzeichen für mögliche Ursachen Ausschau halte, um in der betreffenden Richtung detaillierter nachzufragen.

13. Werkzeuge zur Fehlersuche im Netzwerk

Verschiedene Werkzeuge eignen sich zur Fehlersuche im Netz. Einige verwende ich immer wieder bei den unterschiedlichsten Problemen, andere sind spezialisiert auf bestimmte Fragestellungen.

Hier stelle ich diejenigen, die ich am häufigsten einsetze, in alphabetischer Reihenfolge vor.

14. Literatur

Bücher

Csikszentmihalyi2014

Csikszentmihalyi, Mihaly; FLOW und Kreativität; Klett-Cotta Verlag, Stuttgart 2014
ISBN: 978-3-608-94882-6

Bereits 1975 beschrieb M. Csikszentmihalyi das Konzept des Flow, das bei einigen Menschen auftritt, die offensichtlich Freude an bestimmten Tätigkeiten haben. In diesem Buch fasst er dieses Konzept mit Erkenntnissen aus Interviews kreativer Menschen und Hinweisen zur Förderung der persönlichen Kreativität zusammen.

CT2000

Christiansen, Tom and Torkington, Nathan; Perl Kochbuch; O'Reilly, Köln, 2000;
ISBN: 3-89721-140-8

Neben dem Kamel-Buch ist dieses für mich das wichtigste Buch bei der Arbeit mit Perl. Der Vorteil dieses Buches ist die Darbietung als anwendungsbereite Rezepte, so dass ich mich auf das zu lösende Problem konzentrieren kann und nicht auf die subtilen Feinheiten der Programmiersprache umschalten muss.

Edwards2012

Edwards, Betty; Das neue GARANTIERT ZEICHNEN LERNEN: Die Befreiung unserer schöpferischen Gestaltungskräfte; Ro-

wohlt, Reinbek bei Hamburg, 2012;
ISBN: 978-3-498-01669-2

Dieses Buch hat vordergründig nichts mit Fehlersuche zu tun. Betty Edwards gibt in diesem Buch einige Tipps zum Umschalten der verschiedenen Gehirn-Modi, konkret zwischen dem logisch-analytischen und dem bildlich-intuitiven Denken. Für die erfolgreiche Fehlersuche insbesondere bei schwierigen Problemen ist es vorteilhaft mit beiden Modi souverän umzugehen. Das zu lernen, kann dieses Buch helfen.

Foessmeier1991

Fößmeier, Reinhard; Die Schnittstellen von UNIX-Programmen: Tips zur Programm-Organisation unter UNIX; Springer, Berlin Heidelberg New York London Paris Tokyo Hong Kong Barcelona, 1991;
ISBN: 3-540-53521-7

Eigentlich eher für Programmierer, die Software für UNIX schreiben, gedacht, war dieses Buch eine Offenbarung für mich um zu verstehen, wie Programme unter UNIX und Linux kommunizieren.

Kahneman2011

Kahneman, Daniel; Thinking, fast and slow; Anchor Canada, 2013;
ISBN: 978-0-385-67653-3

Noch ein Buch, das sich mit den Gegebenheiten und Wechselwirkungen der verschiedenen Gehirn-Modi befasst. Nicht direkt zum Thema Fehlersuche, aber sehr erhellend darüber, warum und wie man sich immer wieder selbst an der Nase herumführt.

Loukides1996

Loukides, Mike Kosta; System Performance Tuning; O'Reilly, Sebastopol, 1996;
ISBN 0-937175-60-9

Dieses Buch führt grundlegend in das Gebiet des Performance-Tuning von UNIX-Systemen ein. 2002 gab es eine zweite Auflage mit Gian-Paolo D. Musumeci als Koautor.

Malhotra2002

Malhotra, Ravi; IP Routing O'Reilly, Sebastopol, 2002 ISBN: 0-596-00275-0

Dieses Buch führt in die Grundlagen und Feinheiten des IP Routing ein, wie es in Cisco-Routern implementiert ist. Da es sich nicht auf die Konfiguration beschränkt, sondern auch die Mechanismen und die gesendeten Datagramme erläutert, ist es insbesondere zur Analyse von Routingprotokollen eine große Hilfe. Mit der Routingsoftware Quagga lassen sich die Konfigurationsbeispiele auch unter Linux nachvollziehen, da sich deren Benutzerschnittstelle stark an IOS von Cisco orientiert.

Polya2010

Pólya, George; Schule des Denkens: Vom Lösen mathematischer Probleme; Narr Francke Attempto Verlag, 2010; ISBN 978-3-7720-0608-1

Dieses Buch ist schon sehr alt, das merkt man beim Lesen am heute teilweise unüblichen Sprachgebrauch. Außerdem geht es um das Lösen mathematischer Probleme. Wer sich davon nicht abschrecken läßt, kann daraus wunderbare Inspirationen für das Lösen von Problemen ziehen. Pólya geht von vier Phasen der Problemlösung aus, die er in seinem Buch dann ausführlich

erläutert. Die vier Phasen sind: 1. verstehe die Aufgabe, 2. mache einen Plan, 3. führe den Plan aus, 4. prüfe die erhaltene Lösung.

Roam2009

Roam, Dan; Auf der Serviette erklärt, Mit ein paar Strichen schnell überzeugen statt lange präsentieren; Redline Verlag, 2009; ISBN 978-3-86881-016-5

Der Autor beschreibt in diesem Buch, wie man mit wenig Aufwand Probleme visualisieren kann und damit seine eigenen Gedanken besser ordnen sowie das Problem anderen verständlich erklären kann. Obwohl das Buch, laut Klappentext, Hilfe für das Visualisieren von Geschäftsideen geben soll, sind die Prinzipien und Ideen bei der Fehlersuche mindestens ebenso hilfreich.

Sloan2001

Sloan, Joseph D.; Network troubleshooting tools; O'Reilly, Sebastopol, 2001;
ISBN 978-0-596-00186-5

Dieses Buch stellt etliche Werkzeuge für die Fehlersuche vor und zeigt wie man mit relativ einfachen Werkzeugen komplexe Fragen beantworten kann.

Taleb2012

Taleb, Nassim Nicholas; Antifragile, Things that Gain from Disorder Penguin, London, 2012; ISBN 978-0-141-03822-3

In diesem Buch geht es nicht um Fehlersuche, sondern um Systeme, die durch Störungen stärker werden. Es ist nicht für den Anfänger, der so schnell wie möglich einen Fehler beseitigen will, obwohl auch dieser davon profitieren könnte. Vielmehr

geht es darum unter welchen Umständen Systeme von kleinen Störungen profitieren können und wie man den Zufall zu einem Verbündeten statt zu einem Gegner machen kann.

WCS1997

Larry Wall, Tom Christiansen and Randal L. Schwartz; Programmieren mit Perl; O'Reilly, Köln 1997;
ISBN 3-930673-48-7

Die Einführung in die Programmiersprache Perl. In dem Buch wird nicht nur die Sprache selbst kurzweilig vorgestellt, sondern auch ein Einblick in Hintergründe und die mit dieser Programmiersprache verbundene Kultur gegeben. Die zweite deutsche Auflage führt Jon Orwant statt Randal Schwartz als Autoren auf.

Weidner2012

Weidner, Mathias; Linux kopflos mit PC Engines ALIX; Lulu Press, Raleigh, 2012;
ISBN 978-1-4717-2849-5

Ein Buch über kleine lüfterlose Rechner, auf denen man mit Linux verschiedene Projekte realisieren kann. Das Buch geht unter anderem auf verschiedene Probleme unter dem Aspekt begrenzter Ressourcen ein und beschreibt einige Protokolle und Mechanismen ausführlicher.

Artikel in Zeitschriften, Proceedings, ...

acmqCM2010

Millsap, Cary; Thinking Clearly about Performance; ACM Queue, Volume 8 Issue 9, September 2010,

http://queue.acm.org/detail.cfm?id=1854041

Der Autor geht in diesem Artikel auf die praktische Performanceanalyse von Programmen ein und erläutert den Zusammenhang von Antwortzeit, Durchsatz sowie den Einfluss der Auslastung bei der Nutzung paralleler Ressourcen.

acmqFM2005

Fall, Kevin, McCanne, Steve; You Don't Know Jack About Network Performance; ACM Queue, Volume 3 Issue 4, May 2005, Pages 54-59
http://queue.acm.org/detail.cfm?id=1066069

Die Autoren gehen in diesem Artikel vor allem auf die Performance von TCP ein, insbesondere auf den Einfluss von Bandbreite, Latenz und Window Size auf die Übertragungsgeschwindigkeit.

Bartsch2013

Bartsch, Michael; Service Level Agreements - rechtliche Aspekte; Informatik-Spektrum, Volume 36, Ausgabe 5, Oktober 2013, S. 449ff;
DOI 10.1007/s00287-013-0712-1

In diesem Beitrag für Nichtjuristen geht der Autor in verständlicher Form auf die rechtlichen Aspekte von Service Level Agreements als Bestandteil von IT-Verträgen ein.

Bonwick1994

Bonwick, Jeff; The slab allocator: an object-caching kernel memory allocator; Proceedings of the USENIX Summer 1994 Technical Conference on USENIX Summer 1994 Technical Conference - Volume 1, S. 6

In diesem Artikel stellte der Autor ein effizientes Verfahren zur Speicherverwaltung für SunOS vor. Das Verfahren, Slab Allocator, gelangte 1999 in den Linux-Kernel.

ctDiedrich2012

Diedrich, Dr. Oliver; Zugriffschutz - Dateiüberwachung mit Fanotify; c't Magazin für Computertechnik 2012 Heft 9, S. 174ff; ISSN 0724-8679

Der Autor stellt in diesem Artikel die Möglichkeiten der Fanotify Systemschnittstelle vor.

ctZivadinovic2012

Dušan Zivadinović; Blindflug mit Full Speed - Wie die TCP/IP-Flusskontrolle funktioniert; c't Magazin für Computertechnik 2012 Heft 10, S. 188ff; ISSN 0724-8679

Der Autor erläutert in diesem Artikel leicht verständlich, wie die in TCP enthaltene Flusskontrolle dazu beiträgt, die Datenübertragungsrate einer Verbindung möglichst optimal auszunutzen.

ctHM2013

Hillier, Gernot, Mauerer, Dr. Wolfgang; Linux durchleuchtet - Perf als Universalschnittstelle für Performance-Analysen unter Linux; c't Magazin für Computertechnik 2013, Heft 3, S. 166ff; ISSN 0724-8679

Dieser Artikel geht auf die Möglichkeiten der Performance-Analyse mit Hilfe der Performance-Counter der CPU ein und stellt das Programm `perf` vor, dass eine einheitliche Schnittstelle dafür unter Linux bereitstellt.

ctTZ2013a

Michael Tremer, Dušan Zivadinović; Kürzungsmaßnahme - Internet-Tuning: Puffercontrolle mit CoDel; c't Magazin für Computertechnik 2013, Heft 13, S. 184ff; ISSN 0724-8679

In diesem Artikel erläutern die Autoren die Wirkungsweise des CoDel-Verfahrens zum Router-Queue-Management bei Bufferbloat.

ctTZ2013b

Michael Tremer, Dušan Zivadinović; Paketbeschleuniger - Internet-Tuning mit IPFire und CoDel; c't Magazin für Computertechnik 2013, Heft 13, S. 188f; ISSN 0724-8679

Dieser Praxis-Artikel zeigt, wie man mit der Firewall-Appliance IPFire erste Erfahrungen mit dem CoDel-Verfahren zum Router-Queue-Management bei Bufferbloat machen kann.

Gregg2013

Gregg, Brendan; Thinking Methodically about Performance; CACM Vol. 56, No. 2, February 2013, pp 45 ff.; ISSN 0001-0782

Koenig2012

König, Harald; strace für Linux-Versteher; UpTimes - Mitgliederzeitschrift der German Unix User Group; Ausgabe 2012/1, S. 61ff

Harald König zeigt in seinem Artikel welche Informationen über ein Linux-Programm er mit *strace* gewinnen kann.

Shepard1991

Shepard, Timothy Jason; TCP Packet Trace Analysis; MIT/LCS/TR-494

Die Masterarbeit von Tim Shepard geht ausführlich auf die Analyse des Verhaltens von TCP-Verbindungen ein. Er zeigt darin die Vorzüge von Sequenz-Zeit-Diagrammen für die Analyse und stellt das Programm `xplot` vor, welches die interaktive Analyse dieser Diagramme erleichtert.

WSA2011

Weigend, Johannes and Siedersleben, Johannes and Adersberger, Josef; Dynamische Analyse mit dem Software-EKG; Informatik-Spektrum, Volume 34, Ausgabe 5, Oktober 2011, S. 484ff;
DOI 10.1007/s00287-011-0541-z

In diesem Papier zeigen die Autoren, wie man komplexe heterogene Systeme analysieren kann, wenn man mit einfachen Methoden nicht weiter kommt.

Online-Quellen

AppArmor

Die Seiten des AppArmor-Projekts.
http://apparmor.net/
Aufgerufen: 2014-05-21

Bei der Dokumentation finden sich Handbuchseiten, HowTos, Tutorials und eine Anleitung, wie man bestimmt, ob AppArmor ein Problem verursacht.

Bad block HOWTO for smartmontools

This article describes what actions might be taken when smartmontools detects a bad block on a disk.
http://smartmontools.sourceforge.net/badblockhowto.html
Aufgerufen: 2014-06-02

Der Artikel erläutert, wie man eine Festplatte "reparieren" kann, für die die *smartmontools* fehlerhafte Blöcke gemeldet haben. Obwohl von 2007, ist der Artikel immer noch brauchbar, wenn man gezwungen ist, eine Festplatte mit fehlerhaften Blöcken weiter zu betreiben.

Bullet Journal

For the list-makers, the note-takers, the Post-It note pilots, the track-keepers, and the dabbling doodlers.
http://bulletjournal.com/
Aufgerufen: 2014-05-04

Ich hatte schon jahrelang Laborbücher in A5-Kladden geführt. Mit diesem System habe ich es auf eine höhere Stufe gehoben.

IETF Tools

IETF-related tools, standalone or hosted on tools.ietf.org
http://tools.ietf.org/
Aufgerufen: 2014-05-03

Die Quelle für RFCs und Internet Standards von der Internet Engineering Task Force.

Leanpub

Leanpub - Publish Early, Publish Often
https://leanpub.com/
Aufgerufen: 2014-05-03

Autoren und Herausgeben können Leanpub nutzen, um fertige Bücher und Bücher in Bearbeitung zu veröffentlichen. Dieses Buch selbst ist bei und mit Leanpub entstanden.

Performance Analysis Methodology

Verschiedene Methoden zur Performanceanalyse
http://www.brendangregg.com/methodology.html
Aufgerufen: 2014-08-24

Brendan Gregg stellt hier verschiedene Methoden und Anti-Methoden für die Performanceanalyse vor. Neben diesen finden sich auf seiner Website ausführliche Vorstellungen einiger Visualisierungstechniken nebst ihrer Anwendung bei der Performanceanalyse. Für die Performanceanalyse von Linuxsystemen sind vor allem die Grundlagenartikel unter *Linux Perf* sowie die Checkliste für die USE-Methode bei Linux hilfreich.

Perf Wiki

Perf Wiki
https://perf.wiki.kernel.org/
Aufgerufen: 2014-05-22

In diesem Wiki finden sich viele nützliche Informationen für die Performanceanalyse mittels CPU Performance Counter und `perf`.

POSIX.1-2008

IEEE and The Open Group; The Open Group Technical Standard Base Specifications, Issue 7; POSIX.1-2008 / IEEE Std. 1003.1™-2008
http://pubs.opengroup.org/onlinepubs/9699919799/
Aufgerufen: 2013-11-29

POSIX.1-2008 definiert eine Standardschnittstelle und -umgebung für Betriebssysteme, einschließlich eines Befehlsinterpreters (Shell) und allgemeiner Hilfsprogramme um die Portabilität von Anwendungen auf der Ebene des Quellcodes zu unterstützen.

POSIX.FileCap

Chris Friedhoff; POSIX Capabilities & File POSIX Capabilities
http://www.friedhoff.org/posixfilecaps.html
Aufgerufen: 2013-12-20

Dieser Artikel, 2008 geschrieben, führt mit praktischen Beispielen in den Umgang mit POSIX Capabilities ein.

SELinux

Die Projektseite des SELinux-Projektes.
http://selinuxproject.org/
Aufgerufen: 2014-05-21

Von grundlegenden Konzepten bis anwendbaren Rezepten findet sich hier eine Menge Information zu SELinux

Wayback Machine

Internet Archive Wayback Machine
https://archive.org/web/
Aufgerufen: 2014-05-04

Eine der besten Erfindungen gegen die Flüchtigkeit des Internet. Mit der Wayback Machine lassen sich ältere Versionen von bestehenden und bereits vergangenen WWW-Seiten betrachten. Natürlich sind nicht alle Seiten und erst recht nicht alle Versionen der einzelnen Seiten enthalten. Trotzdem ist es den Versuch wert, wenn eine irgendwo beworbene URL nicht mehr erreichbar ist.

15. Kolophon

Dieses Buch ist im Markdown-Format mit Erweiterungen von Leanpub geschrieben. Anschließend wurden aus den Markdown-Quelldateien die PDF-, EPUB- and MOBI-Versionen erzeugt.

Den Entscheidungsbaum im ersten Kapitel habe ich mit yEd[1] gezeichnet und für die Bucherstellung im PNG-Format exportiert.

Die TCP-Sequence-Diagramme im neunten Kapitel habe ich mit UMLet[2] gezeichnet, in das SVG-Format exportiert und anschließend mit Inkscape[3] in das PNG-Format für das Buch umgewandelt.

Der Router-Graph im neunten Kapitel ist mit dem Programm `dot` von der GraphViz[4] Software erzeugt.

Revision

Die Markdown-Quellen dieses Buches werden im verteilten Revisionsverwaltungssystem *monotone* gespeichert.

Diese Ausgabe basiert auf Revision dae6dbf2676ca6fcb5e2b72c0ac1c1d509a3132a von 2014-10-24.

[1] http://www.yworks.com/de/products_yed_about.html
[2] http://www.umlet.com/
[3] http://www.inkscape.org/
[4] http://www.graphviz.org/