Preface to the preface

This “book” is not (yet) a book. It is incomplete, a works in progress. But given the lack of headway I’ve made, and considering the time and effort I have already invested, I have come to the conclusion that it’s preferable to publish what I have and worry about finishing the remainder another time.

Since I began working on this project back in 2011, a lot has changed in the industry as well as in the SysAdmin class on which this book is based. You may find references or topics to be outdated or seem quaint (although hopefully not obsolete). I remain hopeful that the lessons and examples continue to be applicable, even if details or specific technologies may continue to evolve.

As time goes by, I may come back to update chapters I’ve previously made available. As such, this text may change in content, style, available formats, and organization. It is even possible that updates are already available at https://www.netmeister.org/book/. In the mean time, I welcome suggestions, corrections, contributions, criticism, and any and all commentary at jschauma@netmeister.org.

Jan Schaumann
New York, January 14, 2019

This draft last updated:
January 3, 2021
Preface

The book you are holding is intended to cover the broad topic of “System Administration” from a conceptual point of view with a particular focus on the foundation of systems administration in a large scale context. While the book will pay attention to Operating System specific details – in particular the Unix family of operating systems – it does not provide step-by-step instructions, such as how to set up a mail server or how to run a monitoring system. Instead, the basic underlying principles and the factors that help decide how to best set up a system for a given purpose will be the topic of discussion.

Wherever possible, case studies and real world examples based on my own experiences in both small and large environments will be used to illustrate the lessons learned.

Why another book on System Administration?

Like a lot of Open Source software, this book was written to “scratch a specific itch”. In particular, I was in need of a suitable course book as a companion to my class “Aspects of System Administration”, which I have been teaching at Stevens Institute of Technology since 2005. The class was developed to give students a broad overview of the profession and expose them to the many different practical aspects, but not to teach specific individual technologies that would soon be obsolete or replaced by the next generation. The breadth of the topic required that the class focused on fundamental underlying principles rather than implementation details.

A significant problem with teaching System Administration in an academic setting derives from the fast-paced nature of the Information Technology (IT) world. Curricula and degree programs take a long time to be developed, reviewed and approved. By the time a new textbook has been
published, the technology described may have already become obsolete.

Most existing books relating to System Administration, thorough and useful as they are, have a practical ”howto” approach or are targeted towards people already in the profession. They provide detailed instructions for various specific problems or discuss different implementations of a given software service across different operating systems.

This book is different. It focuses on concepts and the understanding of the given topics on a fundamental level, so as to facilitate the learning of how to build scalable solutions that are flexible enough to adapt to different system requirements.

Who should read this book

This book was primarily written to address the lack of a general purpose course book on the broad topic of System Administration for my own class. As such, the target audience consists of both instructors, teachers, professors as well as, most importantly, students. Because of the wide range of material covered, some prior knowledge of common computer-related topics is required, including a basic understanding of TCP/IP, operating system and file system concepts, user-level experience using a Unix like operating system and proficiency in at least one programming language.

As a graduate level course book, the target audience may of course also include people with some prior experience in System Administration. Their background will help them take the concepts and topics presented to the next level and apply the lessons learned in their own environment.

Organization of the book

The outline of this book follows the syllabus of my class. The class, in turn, was developed as a series of lectures covering complex computer systems from the bottom up. That is, we begin with more low level concepts such as storage devices and file systems and work our way up a stack of layers (including OS and software installation, multi-user basics and networking) until we have well understood, networked and easy to maintain general purpose system, before we dive into the maintenance of special purpose services.

This book is therefore divided into three major parts. We begin with an
introduction to the profession of System Administration and the approach taken in this book. Next, we cover a number of fundamental technologies and concepts; this part makes up the bulk of the content, and each chapter is intended to accompany or complete a single lecture. Suggested exercises, problems or topics of discussions are included wherever suitable. Finally, we conclude in the third part with a brief review and a look ahead into the future direction of System Administration, both the teaching as well as the professional practice thereof.

Throughout each chapter we will pay special attention to three areas: Scalability, Security and Simplicity. These three related and at times overlapping features can be seen as pillars of professional system design. Without them, no solution is complete. Neither can be added after the fact; all three have to be integral components of a system’s architecture.

While in general there is some order to the chapters, you may find yourself flipping back and forth once you arrive in Part II as it builds on the previous chapters. Part IV concludes this book with a look at everything we’ve not covered, a discussion of the ethical and legal implications, and a brief look into the future of the profession.

While I have attempted to provide an overall structure and flow to the topics covered in this book, please feel free to pick and choose topics based on your interest or course needs. Each chapter is, as much as possible, self-contained to allow you to assign reading or exercises as you see fit.

Notes, Digressions, Tangents
At times we will include “supplementary” or rather loosely related information that is too large for footnotes but not important enough to be included in the actual text. Such content will be displayed in boxes like this close to the most relevant paragraph. In order to trick the casual user into actually reading the box, it is accompanied by the entirely unrelated graphic of a surfboard, Channel Islands’ “The Tangent by Kelly Slater”.

Advice for instructors
As a course book, we sometimes include advice for instructors. Such advice might be a suggestion how to integrate the subject matter in the class room, an experience report about something
that helped students in the past, or a pointer to supplementary information. Such advice will be presented in a box like this. The rather adorable dog accompanying this box is, of course, a pointer.

**Anecdotes from experience**

Sometimes we will illustrate important lessons with practical examples, experience reports or anecdotes. These short stories serve to relate the theoretical material covered to the so-called “real world”.

Such anecdotes will be presented using a box like this. The graphic intended to lure you into reading the content is a sock puppet, much like the ones commonly used to entertain children with an educational story of sorts. This particular sock puppet happens to be a donkey because I like to remind myself that despite initial subjective impressions to the contrary it is usually I who turns out to be the ass. Some of these anecdotes will prove this point.

**Conventions**

A lot of ink and many more innocent bits have been wasted on the difference of the terms “UNIX®”, UNIX, “Unix”, and “unix-like”. For the purpose of this book it hardly warrants further distinction anymore: all such systems discussed in this text and used for examples are “a kind of Unix” (though few will actually be certified by The Open Group and thus be allowed to call themselves UNIX®). These include the various Open Source implementations and derivatives such as the BSD family or Linux (itself trademarked, by the way). We will use the term “Unix” to refer to these different systems.

If we have to distinguish between the “Linux” operating system and other “Unices” (the plural we shall avoid), we will follow common convention and simply call the operating system technically consisting of the Linux kernel and all additional software – much of it provided by or derived from the GNU Project – as “Linux” rather than “GNU/Linux”.
We shall also adopt the custom of referring to specific Unix commands by way of writing the name followed in parenthesis by the section of the manual pages in which the command is described. That is, a reference to the `stat(1)` command differs from a reference to the `stat(2)` system call; the documentation for either can be found by providing the section number to the `man(1)` command:

If you see “`stat(2)`” referenced, you’d type `man 2 stat` to get the manual page for the system call; if you see “`stat(1)`”, you’d type `man 1 stat` to get the manual page for the command. Leaving out the section number will get you the first matching manual page, so `man stat` and `man 1 stat` would be equivalent.

Next, this book contains a number of code examples and listings of commands to be executed at the shell prompt. To distinguish between commands entered as a regular user and commands entered as the superuser, we use a different prompt following general Unix convention: a `$` denotes a user prompt, a `#` a superuser prompt. Listing 1 illustrates this.

The English language does not require a gender to be assigned to the “System Administrator”; however, we need to use a personal pronoun if we wish to refer to a person holding such a job title. Consistent use of only a single gender pronoun might raise the impression that the duties can only be performed by a member of that gender, which of course is entirely absurd.

Since any attempt at using “he/she”, “(s)he” or any variation is doomed to become nothing but distracting, we use either “he” or “she” in examples in this book. No suggestion of gender inequality is intended in any of the examples.
Operating systems are distributed by different kinds of organizations. Some have commercial backing; others are provided by a team of volunteers. Some software is provided free of charge, other software requires payment of licensing fees. Some includes the source code, some does not – and either may be the case for both commercially licensed or freely available software. For simplicity, we will use the terms “vendor” or “provider” in either case when referring to the entity distributing the software.

**Examples and Exercises**

As a course book for a graduate level class, the recommended way to follow the material presented is by completing (some of) the examples and exercises included in most chapters.

I have found it invaluable to provide students access to an environment in which they can create different server instances running different operating systems on demand. At the same time, the nature of many of the exercises demand unfettered access to the systems themselves as well as to the network. This is something that is nearly impossible to achieve in a traditional university environment – had a faculty member approached me requesting such access for her students while I was a System Administrator at Stevens Institute of Technology, I would have balked and dismissed the request as hardly feasible.

However, a flexible Infrastructure as a Service (IaaS) environment such as Amazon’s [Elastic Compute Cloud (EC2)](https://aws.amazon.com/ec2/) or other VPS hosting companies overcomes these hurdles; non-commercial solutions to this problem, such as the “Virtual Unix Lab”[^1] for example, exist as well. My own teaching experience has been supported by Amazon’s “AWS in Education” research grants[^2] ; as a result, a number of exercises will reference the EC2 service, but it is entirely reasonable to perform them in another environment that provides similar services. In many cases, virtualization software installed on a regular workstation or laptop allowing students to create virtual instances of different OS is fully sufficient for many exercises as well.

The job of a System Administrator routinely requires self-guided research, analysis and meticulous documentation of one’s practices and procedures. To enforce these practices, many of the assignments, problems and exercises in this book do not have a simple right-or-wrong answer. Just as in real life,
we frequently need to find an answer and then determine if the solution we derived is suitable, even though other equally correct solutions may exist.

To this end, I have made it a habit of requiring students to search any and all information materials available to them – including, but not limited to their course book(s), library materials, online newsgroups and forums, websites and the Internet at large – and to allow them to use such materials so long as they properly cite their resources.

Systems

Practical exercises usually target the Unix family of operating systems, where they focus on exposing students to multiple variations of any given system. I usually assign exercises to be done on the following operating systems, where possible:

- **Linux** – different distributions allow for valuable insights into different solutions to the same problem and frequently illustrate the point that one system running “Linux” may behave rather differently from another. To this end, I usually pick a Red Hat related distribution (Fedora, CentOS) and at least a Debian based distribution (Debian or Ubuntu).

- one of the **BSDs** – as a developer of the NetBSD operating system, I’m of course partial to this particular variant, but any one from this lineage will suffice to illustrate the genetic Unix heritage.

- **Solaris** – now freely available as derivatives of the unfortunately short-lived OpenSolaris project or available as the commercial original version, this system is “different enough” from the more common Linux distributions to provide a good comparison.

Even though the general concepts of System Administration apply across all operating systems and even across administrative domains such as support for a large number of desktop systems versus large deployments of servers in datacenters, we will focus primarily on the large scale installations and infrastructure components. As a result, and due to the significantly different philosophies and practical means by which these systems are maintained, we explicitly exclude Mac OS X and the Windows family of operating systems as a target platform for assignments. Doing so allows us to not get distracted by
implementation details and fundamental platform differences and to instead focus on internalizing the principles and lessons explained.

**Programming Assignments**

Programming assignments are normally checked on the university’s systems. I seldom specify the language in which students have to write their programs. This reflects the real-world scenario, where languages are not usually dictated and the objective is first and foremost to “get the job done”. A few select exercises may specify the programming language to use; there is usually an explicit or implicit requirement, or the implementation in one language lends itself to better illustrate the lessons learned.

Programming assignments are usually graded not only by functionality, but also by code quality, user interface and other factors. A demonstrated understanding of the three core pillars – Scalability, Security, Simplicity – is essential. Frequently the assignments will target these criteria, if at times as hidden requirements.

The practice of giving students maximum liberty in their research also extends to programming assignments. Students are allowed – encouraged, in fact – to search for solutions and reuse existing code, to collaborate and discuss possible approaches amongst each other. It should be noted, though, that as in their workplace, students need to make sure to have permission to (re)use the code in question and to properly credit the origin. I have found that this distinction between lifting code found on a website without acknowledgment and deriving your own solution from an attributed reference has become harder for students to make, which is all the more reason to encourage responsible research.

I believe that this freedom to work as one would under “normal” circumstances teaches awareness of code licensing and improves the student’s research skills. Frequently, the most important part of a successful homework submission is not the functionality of the program itself, but the required accompanying documentation, which illustrates the student’s problem solving progress.
Acknowledgements

My biggest thanks go out to the late Prof. Lawrence Bernstein, formerly of Bell Laboratories, my former Computer Science and Software Engineering professor, an IEEE and ACM fellow, and an industry expert on Trustworthy Computing. Throughout my career, I have found myself going back to the lessons he tried to teach me, the examples he gave, the direction he provided. Larry is the reason all of these words have eventually come into existence and been put into a semi-coherent form:

How about writing a book? If you follow your class notes it is not a burden.

This message came out of the blue back in February of 2011. After a few emails going back and forth, life intervening, my second daughter being born, and me generally trying to figure out how to approach such a project, I eventually signed a contract with Wiley & Sons to actually write this book.

The fact that I never actually completed the work and eventually withdrew my agreement with Wiley & Sons notwithstanding, I remain humbled by the confidence and encouragement I received from Prof. Bernstein. I deeply regret not having been able to complete this work before he died on November 2nd, 2012. Thanks for being a great teacher, Larry!

I have received a lot of feedback on the various chapters making up this book from many people in the industry over the years. I appreciated all the help given by friends and colleagues, with particular thanks to those having reviewed drafts of the different chapters: Subbu Allamaraju, Mohit Chawla, Jennifer Davis, Peter Ellehauge, Patrick Debois, Hubert Feyrer, Al Hoang, Kevin Kempton, Tom Limoncelli James K. Lowden, Jan Lehnardt, Daria Mehra, Marc Farnum Rendino, Kimo Rosenbaum.

Many thanks to Amazon for granting my students ample AWS usage credits. Without access to EC2, many of the exercises in this book would not be possible.

Finally, and as expected, my deepest thanks to my wife Paula and my daughters Ana and Sofie, without whom, as many a famous dedication goes, this book would have been completed much, much earlier. I may still get there.
Bibliography
