Chapter 3

Documentation Techniques

Lack of documentation is becoming a problem for acceptance. –
Wietse Venema

We noted in Chapter 2.2.2 that one of the many ways in which the Unix operating system distinguished itself from other systems was that it included extensive documentation of high quality. Each tool provided by the OS came with a *manual page* describing its use, each library included a description of its interfaces, and configuration files were accompanied by documentation elaborating on its syntax and format.

Even simple systems cannot be operated, maintained, updated – or in a word, administered – without an abstract description of the various moving parts.

But while every System Administrator values good documentation, and will happily agree that, yes, all systems should in fact be properly documented, runbooks created, procedures described, and additional information referenced, it does take practice and conscious dedication to develop a habit of writing high quality system documentation.

For this reason, and before we dive into the fundamental technologies and concepts in Part II, let us briefly focus on a few principles that help us provide better documentation due to a better understanding of what purpose it serves.

3.1 System Documentation Writing 101

Writing is communication. You wish to provide information for your organization, your users, or colleagues. As a reader of system documentation, you are in search of information. This exchange of knowledge is direct communication between the author of the documents and the reader.

The motivation, purpose, and effective techniques used in any kind of writing are similar, so allow us to begin with some advice that, while it may sound like it was cribbed from a "Creative Writing 101" syllabus, still remains sound.

3.1.1 Know Your Audience

Understanding who may be reading our documentation is essential to providing the right information, just as a software engineer must understand who the product's users will be.

Knowing your audience means that you understand what information they are looking for, what background knowledge they may have, what additional documentation or resources they may have access to, and so on.

When we create system documentation, we usually target the following readers[1]:

We write documentation for ourselves. This is the simplest case, as we know our target audience well – or so we think! For the most part, we are writing things down so we can repeat them later on, so we don't forget, so we have a point of reference in the future. This means that we have a pretty good idea what the prospective reader might expect to find, what they might know about the infrastructure etc.

We write documentation for other system administrators. This case is still fairly straight forward. We expect our target audience to be very technical and have a very good understanding of our systems. For example, we write down the steps of a procedure that our colleagues may not normally perform, or we document the setup of a system component we are in charge of that is crucial to the overall operations of our organizations so that they may use this document as a reference when we are not available.

[1]You will note that this same advice for "knowing your audience" translates near verbatim to knowing your *users* when writing software. We will revisit this concept in chapter 9.

We write documentation for other technical people. Our services are used by a number of people in our organization. Some of these are very technical expert users, software developers or other engineers, perhaps. In order to allow these people to get information about our systems quickly, we provide end-user documentation which allows more efficient use of the resources we make available.

We write documentation for our users. The systems we maintain provide a certain service and thus has users, some of whom may be internal to our organization and others which may be outside customers. It is not rare that the people in charge of maintaining the systems provide documentation to these end-users in one way or another to allow them to utilize the service or help them find help when things break down.

We write documentation for other people everywhere. System administrators rely on the Internet at large to find answers to the rather odd questions they come up with in the course of their normal work day. Frequently we encounter problems and, more importantly, find solutions to such problems that are of interest to other people, and so we strive to share our findings, our analyses, and our answers.

Some of the most interesting technical blogs are written by system administrators or operational teams and describe how they scale their infrastructure to meet their demands. Likewise, system administrators present their findings and solutions at industry conferences, write technical papers or participate in Internet standards development, all of which require expert documentation and writing skills.

The differences in target audience have a certain impact on how you would structure your documentation as well as, in some cases, implications on access control of the information in question. Internal documentation targeted towards your peers may well include certain privileged information that you would rather not have made available to the entire organization (or outsiders via the Internet).

As much as the target audience differs, though, the best advice for writing good technical documentation is to avoid making any assumptions about the reader's prior knowledge and familiarity with the systems in questions. At the same time, we need to carefully identify and reference whatever assumptions, unavoidable and necessary as they may be, *are* being made.

Secondly, it is important to clearly distinguish *internal* from *external* documentation. The distinction here is not so much between your corporate or

legal identity and the public at large – an easy and obvious one to draw and enforce – but between distinct entities within a single organization. Documentation containing certain details regarding your infrastructure's network architecture, access control lists, location of credentials and the like may simply not be suitable to be made available outside your department. Unfortunately the boundaries can be less than obvious as people move in and out of departments as an organization evolves.

3.2 Online Documentation

One of the main differences from creative writing is that our documentation is in almost all cases intended to be read online. Even though certain documents such as a contact information registry of the support staff or the emergency escalation procedures may be useful to have in print, virtually all information we collect and make available are read in an online or computer context. This has a profound impact on the structure and style of our documents.

Studies ([5], [6], et al.) have shown that not all "reading" is equal: content consumed via a web browser, for example, is frequently only glanced through, skimmed over or otherwise not paid full attention to. When we read documents online, we tend to search for very specific information; we attempt to extract the "important" parts quickly and tend to ignore the rest.

As a result, online documentation needs to be much more succinct than information intended for actual print.[2] Sentences and paragraphs need to be short and simple; the subject matter covered concise. Additional information, further references, or more in-depth analyses can always be made available to the reader via appropriate linking. Section headings, bold or italic fonts and the use of itemized lists make it much easier for the reader to quickly absorb the content.

In addition, we are much less inclined to tolerate superfluous information when reading online. While Dale Carnegie's advice "Tell the audience what you're going to say, say it; then tell them what you've said." is spot-on for effective presentations or traditional writing, building up each document you create with an introduction and summary incurs too much repetition to be

[2]Imagine reading this book entirely online, perhaps as a series of blog posts. You would likely skip a lot of content and perhaps rely on summaries, bullet points and other visual emphases to find what might seem worth reading. I certainly would have structured the content differently.

useful. Getting straight to the heart of the matter is usually advisable in our context.

Another benefit of providing documentation online is that it becomes immediately searchable and can be linked to other sources of information. Helping your readers to find the content they are looking for becomes trivial if your documentation framework includes automated index generation and full text search. To make it easier for your readers, be sure to use the right keywords in the text or use "tags" – remember to include or spell out common acronyms or abbreviations depending on what you and your colleagues or users most commonly use!

tl;dr

Sometime in 2011, an acronym established itself in the "blogosphere": *tl;dr – too long; didn't read*. This acronym (in general use since at least 2003) was frequently accompanied by a short summary of the content in question, thus allowing online readers lacking the patience to actually read the document to nevertheless form opinions about it. While it is certainly useful to provide a succinct summary of the content in e.g. an email (where an appropriate 'Subject' line or an introductory sentence may suffice), ii you feel inclined to provide a *tl;dr* summary to your documentation, chances are that you're addressing the wrong audience.

3.3 Different Document Types

Knowing your audience and understanding what you wish to communicate to them are the most important aspects to understand before starting to write documentation. From these two main points derive a number of subsequent decisions that ultimately influence not only the process flow but also the structure of the document. Drawing parallels to software engineering once more, where we decide on the programming language or library to use based on its suitability to actually solve the given problem, we find that here, too, we need to use the right tool for the job. We cannot structure our end-user manual as a checklist, nor can we provide online help or reference documentation as a formal paper.

Each document type calls for a specific writing style, a unique structure

and flow of information. In this section, we will take a look at some of the most common distinct categories of system documentation.

3.3.1 Processes and Procedures

Purpose	describe in detail how to perform a specific task; document the steps to follow to achieve a specific goal; illustrate site-specific steps of a common procedure
Target audience	all system administrators in the organization
Structure	simple, consecutive steps; checklist; bullet points with terse additional information

This is probably the most common type of document you will have in your information repository. Without any further classification, pretty much anything you document will fall into this category; it is a broad definition of what most people think of when they think about system documentation. It is worth taking a closer look, as different types of process documentation exist and hence call for different formats or presentation.

Processes usually provide a description of what is done in a given situation, what the correct or suitable steps are to resolve a given problem or how a routine task is accomplished. These documents are generally focused on the practical side of the business or organization.

Procedures are focused slightly more on the operational side of things. They not so much describe what is to be done but list the actual commands to issue to yield a specific result. It is not uncommon to have well-documented procedures be turned into a "runbook", a list of ordered steps to follow and commands to issue under specific circumstances. Carefully and correctly written, a runbook can frequently be used to let even inexperienced support staff respond to an emergency situation. The procedures listed usually include simple if-this-then-that directions with exit points to escalate the problem under certain circumstances. Furthermore, a good runbook usually makes for an excellent candidate for automation – more on that in Chapter 8.

Another type of document can be considered a subcategory of a process document: the ubiquitous *HOWTO*. Instructional descriptions of how to

accomplish a certain task generally skip any reasoning or justification of the ultimate goal and focus on the individual steps to reach it instead. Using simple sentences, itemized, or enumerated lists and often times including sample invocations and command output, this document will outline the details of each task, noting in particular how the environment in question may differ from those covered by other such guides.

When writing process and procedure documents, it is usually a good idea to include actual command invocations together with their complete output rather than to fabricate seemingly illustrative invocations. By using the exact commands that are applicable in the given environment we make it easy for the reader to follow the process as well as to find this document if they only remember a specific command related to the task at hand.

3.3.2 Policies

Purpose	establish standards surrounding the use of available resources
Target audience	all users of the given systems
Structure	full text, including description and rationale

Most people think of *policies* as lengthy documents full of footnotes and fine print, frequently drafted by your organization's legal or human resources departments. But there are a large number of necessary policies that are – or should be – written by (or with the help of) System Administrators, as they relate to and impact their work directly. In particular, computer related protocols such as the "Terms of Service" or Acceptable Use Policies (of the systems, not the software product(s) possibly offered to outside customers on said systems), various Service Level Agreements (SLAs), and information security guidelines represent an abstract description of practical implementations on the system side under the given circumstances.

Unfortunately, it is not uncommon to have such guidelines be drafted completely outside the functional or possible environment in which they need to be turned into action. In such cases, these policies become meaningless, as they are adhered to only in the letter, but not in the spirit – the Payment Card Industry Data Security Standard (PCIDSS) compliance is, sadly, a common example.

The documents governing the interactions with the customers or consumers must be specific, clear, and avoid loopholes. Hence, they should be written with the direct feedback and input from the people in charge of implementing the rules and regulations prescribed.

For an excellent and much more in-depth discussion of Computing Policy Documents, please see [2].

3.3.3 Online Help and Reference

Purpose	list and index available resources; illustrate common tasks; describe common problems and provide solutions
Target audience	all users of the given systems; possibly restricted set of privileged users (depending on the resources indexed)
Structure	simple catalog or itemization; short question-and-answer layout; simple sentences with example invocations

In this category you will find all the various documents provided by System Administrators, IT Support, Help Desk, and similar groups that are intended to allow users to find solutions to their problems without having to engage personal assistance. A common solution includes increasingly complex documents entitled "FAQ" (for "Frequently Asked Questions").

When creating an FAQ compilation, it is important to focus on the questions your users *actually* ask, not the ones that you *think* they might ask. All too often these documents include answers to questions that are not, in fact, all that frequent, but that the author happened to have an answer to or would like the users to ask.

It should be noted that it may well be necessary to separate some of these documents for one audience from those provided to another. The references you provide to your peers may include privileged information, and the style in which you guide inexperienced users would be ill-suited to help your expert users find solutions to their problems. Once again, knowing your audience is key.

In addition, it is important to distinguish between *reference* material and

what might be called a "User Guide". The latter is conceptual in nature; it might explain the nature of the problem and how it is solved. The former is terse, simple, and strictly limited to the available options; it does not elaborate on why things are the way they are, it merely presents the interface.

3.3.4 Infrastructure Architecture and Design

Purpose	describe in great detail the design of the infrastructure; illustrate and document information flow; *document reality*
Target audience	other system administrators in the organization
Structure	descriptive sentences with detailed diagrams; references to rationale and decision making process behind the designs

Among the most important sets in a System Administrator's information repository are the infrastructure architecture and design documents. In these, the system architecture is described in great detail; they aim to provide an accurate representation of reality together with references to the rationale behind certain decisions.[3]

These records are meant to be authoritative and can be relied on to resolve disputes about what traffic may flow between security zones, to make decisions about where to place a new device, or to identify bottlenecks in throughput, to name just a few examples. Although the quality and accuracy of these documents ought to be a priority, all too often descriptions of the architecture or network are neglected to be updated, quietly drifting into obsolescence.

One of the reasons for this perpetual state of being outdated is the inherent complexity of the subject matter. Complex infrastructures are frequently described in complex, or worse, convoluted ways. Keep your content simple and to the point: you should be able to describe your network in a few straightforward sentences. If you feel yourself in need of numerous illustrations or lengthy explanations, break out larger components into less complex

[3]At times, this "reality" may, in fact, be a *desired* or *ideal* reality; it is important to explicitly note wherever this diverges from the actual architecture as implemented.

blocks, each documented individually.

3.3.5 Program Specification and Software Documentation

Purpose	describe a single software tool, its capabilities and limitations; illustrate common usage; provide pointers to additional information
Target audience	all users of the tool, inside or outside of the organization
Structure	short, simple, descriptive sentences; example invocations including sample output; possibly extended rationale and detailed description, command or syntax reference etc. via in-depth guide

One of the mistakes System Administrators tend to make is to not treat their tools as a complete software product. Software packages installed from third parties are often ridiculed or derided for having incomplete documentation or lacking examples, yet at the same time, our own little tools and shell scripts remain stashed away in our `~/bin` directory, possibly shared with colleagues. We will cover this topic in much more detail in Chapter 9, but let us consider how our expectations of other people's software differ from what we provide ourselves.

Every software package, every tool, every script, needs documentation. In addition to the generic information we might find on upstream vendors' websites such as where to download the software from, what users are in need of is practical examples. With the tools you have written yourself, you have the distinct advantage of knowing *precisely* the use cases most relevant to the users of the software, and so you should provide them.

I have made it a habit of creating a very simple "homepage" for every software tool I write. On it, I include a short summary of what the tool was designed to do, where in the revision control repository the sources can be found, where feature requests and bug reports can be filed, how the software can be installed, a pointer to the change log, and version history and last and certainly not least a number of example invocations with expected output. This information is also included in the project's repository in the form of a

`README`.

If the software in question becomes more complex, you may wish to reference additional documents, including pointers to the software architecture, the rationale for certain design decisions, a software specific FAQ etc.

3.4 Collaboration

Unlike other kinds of writing, creating and maintaining system documentation is not a solitary task. Instead, you collaborate with your colleagues to produce the most accurate information possible, and allowing end users to update at least some of the documents you provide has proven a good way to keep them engaged and improve the quality of your information repository.

Depending on the type of document, you may wish to choose a different method of enabling collaboration. Some documents should be treated as less flexible or mutable than others: for example, you probably do not want your customers be able to modify the SLAs you agreed to, but you *do* want to encourage your users to contribute use cases or corrections to your software documentation.

As Bertrand Meyer, creator of the Eiffel programming language, observed:

> *Incorrect documentation is often worse than no documentation.*

For this reason, it is critical that you lower the barrier for collaboration and make it easy for others to contribute with corrections or updates. Carefully marking obsolete documents as such is similarly of importance, but also beware overeagerly removing "stale" content: being able to read up on historical design and architecture decisions can be invaluable even if the documents do not reflect the *current* state.

Documentation should be kept under some kind of revision control, much like software code is. Many different solutions exist, ranging from storing simple text or markup files in your own repository, to using a database backed "Wiki", to hosting documents online using a cloud provider, to using e.g. Google Docs. Realistically, you will likely encounter a combination of all of the above. When choosing a solution, focus on these properties:

- Usability; how easy is it for you and your colleages to edit the documents, to keep them up to date?

- Collaboration; how easy is it for others to make corrections, to comment on the content, to suggest and provide improvements?
- Revision Control; how easy is it to review a document's history, see what has changed between edits, see who made what changes when?
- Access Control; how easy is it for you to grant or deny access to individuals, teams, the organization, the public?
- Searchability; how easy is it for you and your users to discover the documents?

Whichever collaborative documentation solution you choose remember that the easier you make it for yourself, your colleagues and your users to create content, to keep it up to date and to correct even minor errors, the better your documentation will be. And with good documentation come happy users, fewer alerts, and significantly less stress for you.

3.5 Formats

Finally, a word on the format of your documentation. Be aware of how your users might read the documents you provide. Frequently, it is necessary to be able to search documents offline, to forward them via email, to link to them, or to copy and paste portions.

With that in mind, be mindful of the format in which you generate and present your information. System Administrators, heavily influenced by the Unix philosophy, are often partial to plain ASCII text or simple markup formatting.

To increase the readability of your documents use a short line-length and the copious use of paragraphs. Viewing a single large block of run-away text with no line breaks immediately puts stress on the reader, as absorbing the information provided therein requires a high degree of concentration and eye movement.

Breaking up your text into smaller paragraphs helps the reader relax and facilitates reading comprehension, speed, and ease. Write the text as you would read it out aloud, with paragraphs allowing the reader to catch their breath for a second.

If you are writing a longer technical document, you can further structure it using headlines, numbered sections, subsections etc. using different ways to

underline or emphasize the section titles. Similarly, you can use itemization, bulleted or numbered lists, or indentation to make your text easy to read.

Use short sentences, even (and especially) if you're German. Just like one single block of text is hard to read, so are never ending sentences with multiple conditionals and subclauses. You are not writing Molly Bloom's Soliloquy.

Use proper punctuation. A period will almost always suffice; semicolons may be used as needed, but exclamation points are rarely called for!

Resist the temptation to use images instead of text. If you are unable to distill the concepts or thoughts into language, then you have likely not fully understood the problem. Use illustrations as *supplementary*, not *instead of* information. Text ought to be your primary content: it can can easily be skimmed, indexed and searched, glanced through in a matter of seconds, and parts of a paragraph be re-read with ease; a screencast or video, to cite an extreme example, must be watched one excruciating minute at a time (not to mention the challenges non-textual media pose to visually impaired users).

That's it - with the above advice in mind, you should find that you soon spend a lot more time on defining your thoughts and putting the important information forward rather than fiddling with font faces, sizes and colors.

And remember, if plain text is good enough for RFCs, the standards used to define the Internet and just about everything running on it, then it's quite likely going to be good enough for you.

Problems and Exercises

Problems

1. Identify a tool or utility you use on a regular basis which does not have an adequate manual page. Write the fine manual! Submit it to the tool's author/maintainer.

2. Identify existing systems documentation in your environment. What is the documents' intended purpose, and do they meet that goal? What format are they in? Are they up to date and accurate? How and when are changes made?

3. Many Open Source projects also are transparent in the ways that their infrastructure is maintained and operated – identify a major project and review their documentation. Compare to how different companies present their best practices (for example on a company blog, in presentations at conferences, in articles about their infrastructure, ...). Does documentation play a significant role?

Bibliography

[1] Thomas A. Limoncelli, *Time Management for System Administrators*, O'Reilly Media, 2005

[2] Barbara L. Dijker (Editor), *Short Topics in System Administration: A Guide To Developing Computing Policy Documents*, USENIX Association, Berkeley, CA, 1996

[3] William Strunk Jr., E. B. White, *The Elements of Style*, Longman, 4th Edition, 1999 (The original text is now in the public domain and available online, for example at `http://www.gutenberg.org/ebooks/37134` (visited January 23, 2017).)

[4] Gerald J. Alred, Charles T. Brusaw, Walter E. Oliu, *The Handbook of Technical Writing*, St. Martin's Press, 8th edition, 2006

[5] Mark Bauerlein, *Online Literacy Is a Lesser Kind*, The Chronicle of Higher Education, September 19, 2008; also available on the Internet at `http://greatlibrarynews.blogspot.com/2008/09/online-literacy-is-lesser-kind.html` (visited January 23, 2017)

[6] Jakob Nielsen, *Writing for the Web*, miscellaneous papers and links; on the Internet at `http://www.useit.com/papers/webwriting/` (visited April 06, 2012)

[7] Jan Schaumann, *The Art of Plain Text*; on the Internet at `https://www.netmeister.org/blog/the-art-of-plain-text.html` (visited January 28, 2016)