

Doc Like an Egyptian

Dru Lavigne

Documentation Lead, iXsystems

SCALE, January 23, 2016

All the old paintings on the tombs,
They do the sand dance, don't you know?

If they move too quick (oh whey oh)
They're falling down like a domino.

Walk Like an Egyptian, The Bangles

Outline

Overview of documentation goals and inherent difficulties

Introduction to Sphinx as part of an open source documentation solution

Disclaimer: based on my experiences. While from an open source perspective, discussion also applies to proprietary documentation.

In Theory....Docs:

are published with each software release...

in multiple formats which suit the needs/devices of that software's user base...

in multiple translations which match that software's global audience...

and are grammatically and technically correct.

The Reality

Docs are incomplete or outdated (assuming they even exist)

Docs are given lower priority than code

Noone wants to write docs / Noone reads docs anyways

No perfect doc management solution exists

Incomplete/Outdated Docs

Software is a *MOVING* target with varying release schedules

Outdated, unversioned docs are actually worse than no docs

Need process for versioning docs, providing revision control, archiving older docs, and helping people find the correct docs for their software version

I Get No Respect Around Here...

Cultural issue: docs are equally important to code and both are equally important to Q/A

Know your audience and give them credit
(users actually want usable docs)

Train projects' question answerers to gently point users to the pertinent section of the docs and to create bug reports (or update docs)

Finding the “Perfect” Tool

Is its formatting language easy to learn or a significant barrier to entry?

Are editors available which understand the syntax?

Does it support the required layout? (code excerpts, table of contents, glossary, searchability, internal and external links, images, charts, localization, warnings, etc.)

Finding the “Perfect” Tool

Dru's 1st Law of Doc Tools: the number and quality of the outputs is inversely proportional to the ease of the markup language

Dru's 2nd Law of Doc Tools: the fewer the number of doc maintainers, the higher the number of desired outputs

Understand your audience and your project's limitations

doc/odt

Good: WYSIWYG editor available to any author

Bad: templates are painful

Collaborative editing and tracking changes is often a tangled mess of emails, editing conflicts, or waiting for someone else to make their changes

Outputs are limited and require various degrees of cleanup

wiki

Good: entry barrier fairly low and syntax quick to learn

Bad: no concept of ToC, content flow, or versioning

If you build it, they don't necessarily come (except for the spambots...)

Outputs are limited and typically require a lot of cleanup; translation tools are clunky

Latex

Good: multiple outputs integrate well into build and translation systems; can format into any desired layout

Bad: very high barrier to entry as it takes a dedicated time (and interest) commitment to learn (and teach) the formatting language

Sphinx Features

Relatively easy-to-learn formatting language

Supports common outputs: HTML, PDF, ePUB, LaTeX, Texinfo, man pages, txt, API docs

Provides a number of extensions such as auto-numbered figures

Automatic generation of ToC, index, glossary

Sphinx Features

Source files are text and easily integrate into existing revision control, translation, build, and CI infrastructures

Fully customizable `conf.py` for controlling doc layout

Several useful built-in builders (eg link checker)

Sphinx Features

Several built-in themes and support for custom themes; layout fully customizable using CSS

Anything can be linked (figures, keywords, headings, etc.)

Writers can use any text editor on any system with Python installed (or issue git pull requests)

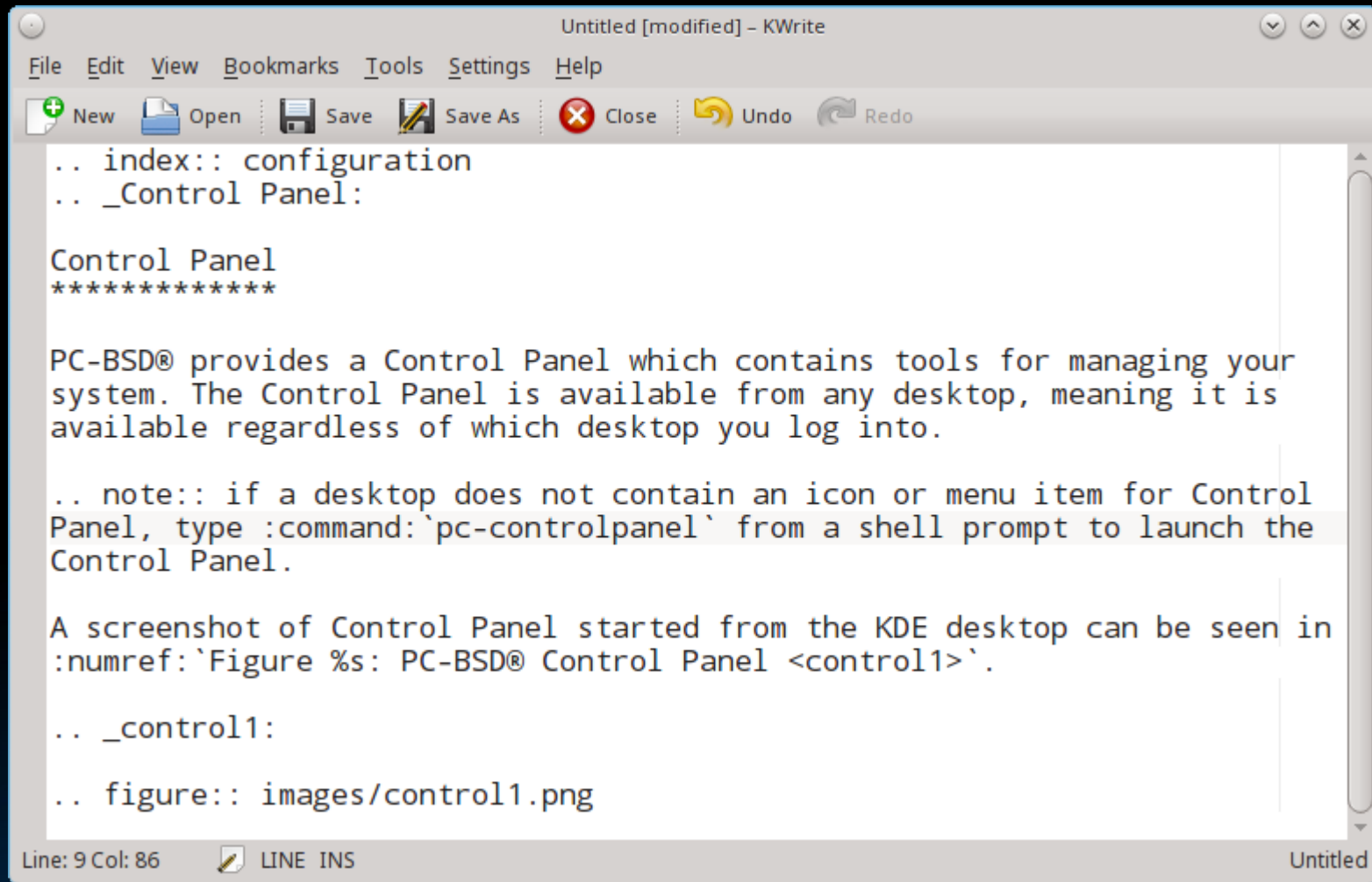
Sphinx Limitations

Some odd formatting limitations require CSS workarounds (eg bold italic)

Search SUCKS (and ironically there is a search engine of the same name)

Documentation is limited and needs more usage examples

Sample Text



The image shows a screenshot of a KWrite text editor window. The window title is "Untitled [modified] - KWrite". The menu bar includes "File", "Edit", "View", "Bookmarks", "Tools", "Settings", and "Help". The toolbar contains icons for "New", "Open", "Save", "Save As", "Close", "Undo", and "Redo". The text area contains the following content:

```
.. index:: configuration
.. _Control Panel:

Control Panel
*****

PC-BSD® provides a Control Panel which contains tools for managing your
system. The Control Panel is available from any desktop, meaning it is
available regardless of which desktop you log into.

.. note:: if a desktop does not contain an icon or menu item for Control
Panel, type :command:`pc-controlpanel` from a shell prompt to launch the
Control Panel.

A screenshot of Control Panel started from the KDE desktop can be seen in
:numref:`Figure %s: PC-BSD® Control Panel <control1>`.

.. _control1:

.. figure:: images/control1.png
```

At the bottom left, the status bar shows "Line: 9 Col: 86" and "LINE INS". At the bottom right, it says "Untitled".

Sample HTML Output

PC-BSD User Guide 10.2 »

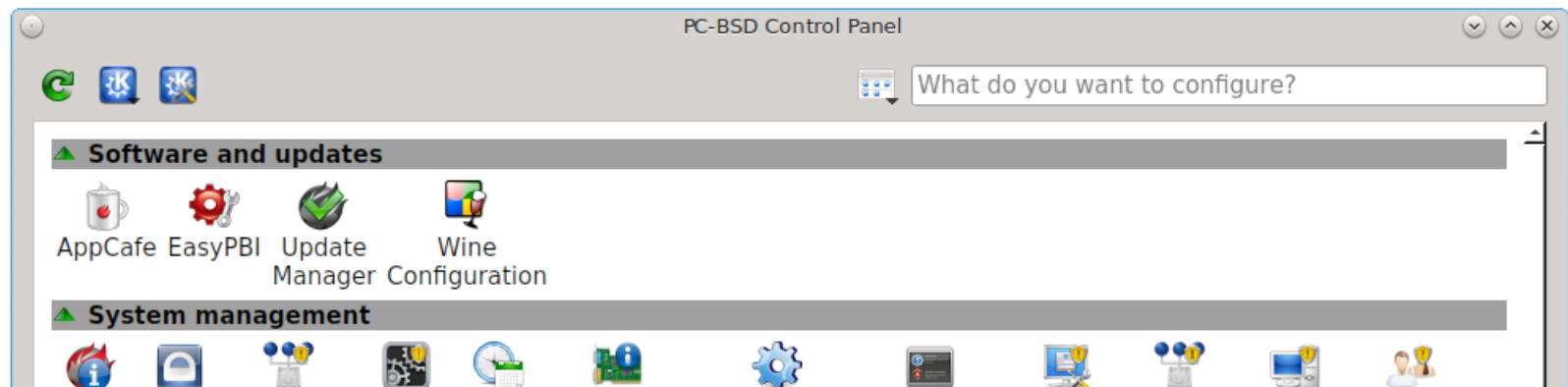
- 8. Control Panel
 - 8.1. EasyPBI
 - 8.1.1. Creating a PBI Module
 - 8.1.2. Advanced Module Configuration
 - 8.1.3. Bulk Module Creator
 - 8.1.4. EasyPBI Settings
 - 8.1.5. PBI Module Components
 - 8.2. About
 - 8.3. Active Directory & LDAP
 - 8.4. Boot Manager
 - 8.4.1. Managing Boot Environments from the Command Line
 - 8.5. Hardware Compatibility
 - 8.6. Login Manager
 - 8.7. Service Manager
 - 8.8. System Manager
 - 8.9. User Manager
 - 8.9.1. Managing

8. Control Panel

PC-BSD® provides a Control Panel which contains tools for managing your system. The Control Panel is available from any desktop, meaning it is available regardless of which desktop you log into.

Note: if a desktop does not contain an icon or menu item for Control Panel, type **pc-controlpanel** from a shell prompt to launch the Control Panel.

A screenshot of Control Panel started from the KDE desktop can be seen in Figure 8.1: PC-BSD® Control Panel.



Sample Pootle Interface

PC-BSD Translations Register Log In - T +

German x ▾ / PC-BSD Handbook x ▾

Overview News **Translate**

[More](#) | [Hide](#) | [Less](#)

To begin the PC-BSD® installation, insert the prepared boot media and boot the system. If the computer boots into an existing operating system instead of the installer, reboot and check your computer's BIOS program to ensure that the drive containing the installation media is listed first in the boot order. Save your BIOS changes and reboot.

Um die Installation von PC-BSD® zu beginnen, fügen Sie das vorbereitete Medium für das Booten ein und booten Sie das System. Wenn der Computer in ein bestehendes Betriebssystem anstatt in das Installationsprogramm bootet, booten Sie erneut und prüfen Sie das Programm für das BIOS Ihres Computers, um sicherzustellen, dass das Laufwerk, welches das Medium zur Installation beinhaltet, als erstes bei der Rangfolge zum Booten gehört wird. Speichern Sie die Änderungen für das BIOS und booten Sie erneut.

German / PC-BSD Handbook / install.po String 473416

Locations: ../install.rst:15

English W

The initial boot screen, shown in Figure 3a, offers a choice of using either the graphical or the text based installer. Unless you select otherwise, the graphical installer will load. To instead use the text based installer, either from the console or over a serial connection, use the arrow keys to select an option. If the graphical installer hangs when loading the graphics driver, try selecting the VESA mode option of the graphical installer.

Comment: 2b02a1475f434fd1a6fa3172c46e387c 57b9bc4fbeee4b44a714c9b7b0973210

Search Filter by: [Export View](#)

/ 112

Helpful Tips

Do you plan to convert existing docs or archive those in current format and start with new docs?

Open source converter utilities exist for most formats, often multiple utilities per format

Experiment by converting a small doc that contains most of your formatting requirements

Helpful Tips

Expect to spend time cleaning up the conversion

Is your goal to have the docs match a specific software version? to have only one version that is the latest and greatest?

What output formats are required? Versioned PDFs? HTML on your project page? Built into the software itself as a help system?

Helpful Tips

Review your code repository (you do have one, right? if not, you need one!)

Determine the required doc versioning system and where to store your .rst files

Update the README.md (or equivalent) with instructions for authors to create their own doc build environment

Helpful Tips

Review your converted docs and create a cheat sheet that includes a list of the formatting tags and conventions used by your project

Include any gotchas to help new authors quickly get up-to-speed

Publish the cheat sheet (eg README.md in doc directory of repo)

Helpful Tips

Spend some time playing with `conf.py` and experiment with existing themes BEFORE writing new docs as themes change the layout

Research, experiment with, and install extensions that are useful for your docs

Determine how much layout customization is required and who has the CSS skills for extra customizations

Resources & Examples

<http://sphinx-doc.org/>

<http://pootle.translatehouse.org/>

github.com/pcbsd/pcbsd/tree/master/src-qt5/docs/

github.com/pcbsd/sysadm/tree/master/api/

Questions

Contact:

dru@freebsd.org

URL to Slides:

<http://slideshare.net/dlavigne/scale2016>