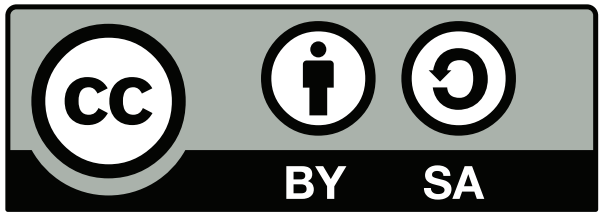
Configuration Management

Markus Raab

Institute of Information Systems Engineering, TU Wien

9.3.2018



Language of the Talk?

Task

Hands up if you prefer German.

Unanimous preference of German required, otherwise English.

Organization

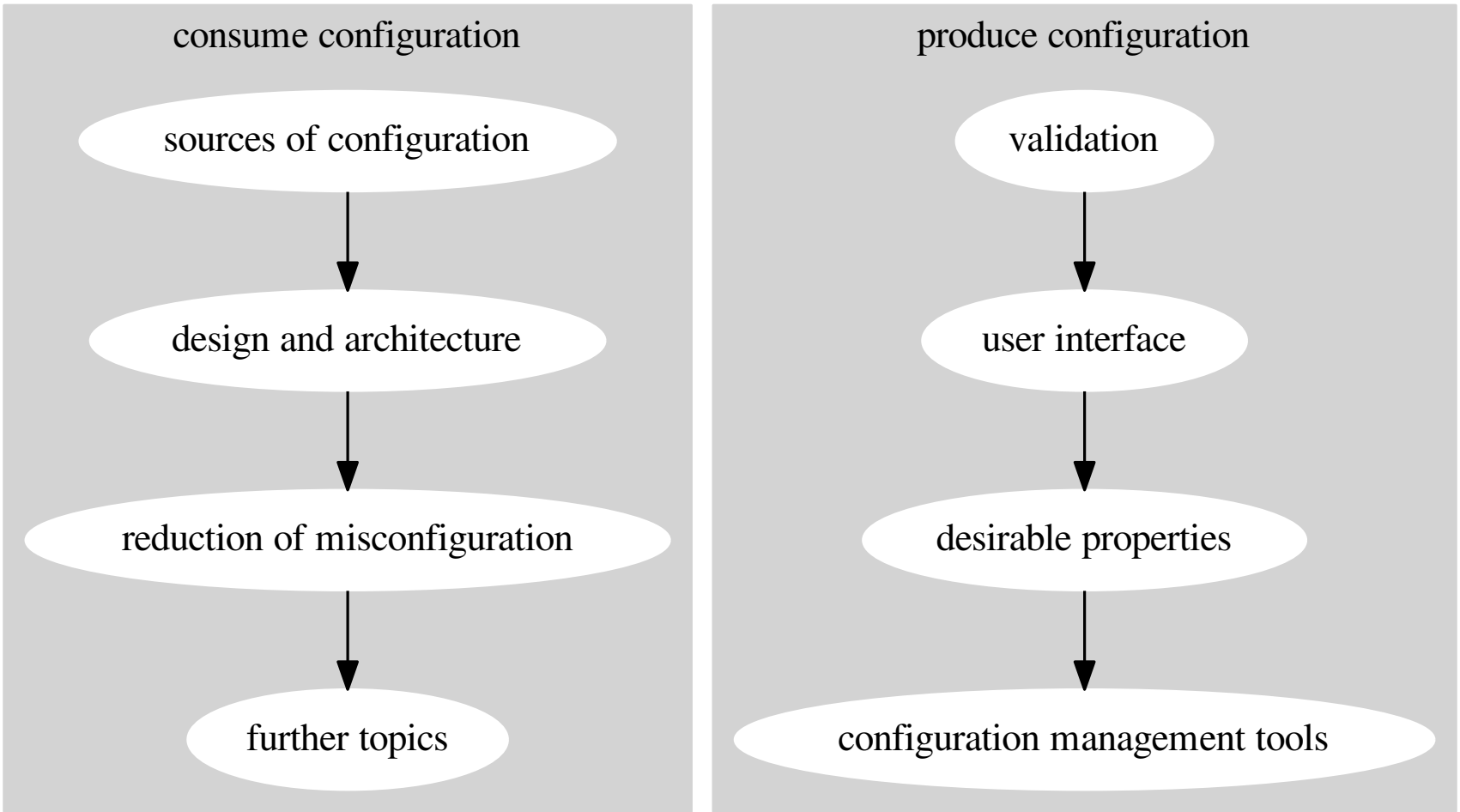
Slides now available at
`https://www.libelektra.org/ftp/elektra/slides/cm/`

Next lectures:

- 9.3.2018: **TISS registration**
- 16.3.2018: **topic homework and talk** (GitHub account!)
- 23.3.2018: teams found together
- 13.4.2018: homework submitted, topics of team exercise
- 20.4.2018: **no lecture**
- 18.5.2018: guest lecture
- 25.5.2018: team exercise submitted
- 22.6.2018: last corrections of team exercise

Popular Topics

- 4 validation
- 4 user interface
- 3 tools (benefits?)
- 3 testability
- 3 complexity reduction (when conf. needed?)
- 3 architectural decisions
- 2 Puppet
- 2 modularity
- 2 environment variables
- 2 documentation
- 2 configuration specification
- 2 command-line args
- 2 code generation
- 1 variability
- 1 self-description
- 1 round-tripping
- 1 introspection
- 1 early
- 1 dependencies
- 1 context-awareness
- 1 auto-detection
- 1 administrators


consume configuration
sources of configuration
design and architecture
reduction of misconfiguration
further topics
produce configuration
validation
user interface
desirable properties
configuration management tools

Configuration File Formats

1. Configuration File Formats
   - Definitions
   - Formats
   - Abstractions
2. Command-line Arguments
   - Usage and Popularity
   - Semantics
3. Environment Variables
   - Trends
   - Requirements
   - Conclusion

Basic Definitions

The ***execution environment*** is information outside the boundaries of each currently running process [6].
Controlling the execution environment is essential for configuration management [5, 11], testing [24, 28], and security [9, 14, 17, 22].

Configuration Setting

Definition

A ***configuration setting***, or ***setting*** in short, fulfills these properties:

1. It is provided by the execution environment.
2. It is *consumed* by an application.
3. It consists of a key, a configuration value, and potentially *metadata*. The ***configuration value***, or ***value*** in short, influences the application's behavior.
4. It can be *produced* by the maintainer, user, or system administrator of the software.

Synonyms for Configuration Settings

User preferences [12] and ***customization*** [2] stress that users make the change although that might not always be the case. ***Variability points*** [10, 15, 16, 25–27] aim at describing the capability of software to adapt its behavior. ***Derivation decision*** [7, 8] puts the decisions to make and not the result in focus. ***Configuration parameter*** [3, 30] is easily confused with other kinds of parameters. ***Configuration item*** [4] or ***configuration option*** [21, 31, 32] are sometimes not applicable, for example, “proxy option”, or “language item”. ***Configuration data*** [11] is often used in the context of programmable gate arrays and has a different meaning in that domain.

Definition

A ***configuration file*** is a file containing configuration settings.

Definition

A ***configuration file*** is a file containing configuration settings.
A Web server configuration file:

```
1 port=80 ; comment
2 address=127.0.0.1
```

Task

What are keys? What are configuration values? What is metadata?

Definition

A ***configuration file*** is a file containing configuration settings.
A Web server configuration file:

```
1 port=80 ; comment
2 address=127.0.0.1
```

The configuration values are 80 and 127.0.0.1, respectively.
Other information in the configuration file is metadata for the configuration settings (such as the comment).

Types of Formats

- CSV (comma-separated values)
- semi-structured
- programming language
- document-oriented, literate
- especially made (easy to use vs. easy to implement)

CSV formats

- passwd: 3rd November, 1971
- passwd and group use : as separator
- are difficult to extend (e.g., GECOS)
- only used for legacy reasons
- are replaced one-by-one (e.g., inetd, crontab)

Trends

- away from CSV
- towards general-purpose serialization formats (INI, JSON)
- human-read/writable (YAML, HOCON, TOML)
- programming language as configuration file

Introduce somebody

Task

Talk with someone else about your favourite configuration file format.

Task

Did you implement a configuration file parser and/or invented a new configuration file format?

Task

Explain to everyone about the other person and his/her favourite configuration file format.

Method

S: source code analysis of 16 applications, comprising 50 million lines of code [18]

Q: survey with 672 persons visiting, 162 persons completing the survey [18]

Why are so many formats present?

Q: "In which way have you used or contributed to the configuration system/library/API in your previously mentioned FLOSS project(s)?" [18]

- 19 % persons ($n = 251$) claim to have introduced a configuration file format.
- 29 % implemented a configuration file parser.
- 15 % introduced a configuration system/library/API.
- used external configuration access APIs (34 %).

Abstraction

Requirement

A configuration library must be able to integrate (legacy) systems and must fully support (legacy) configuration files.

How can we deal with the many formats?

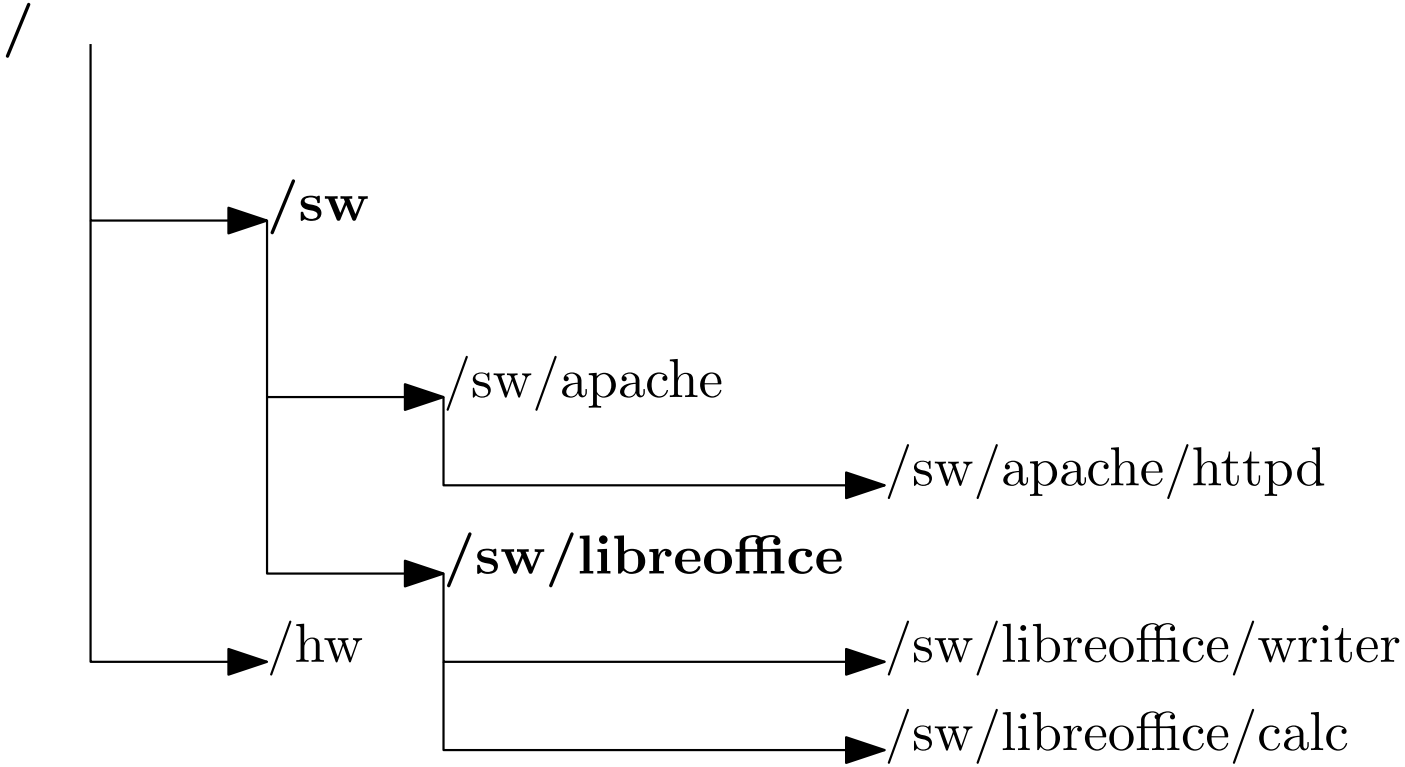
Key-Value

A key-value pair is the simplest generic data structure [23]. While all these formats above have many differences, all of them represent configuration settings as ***key-value pairs*** [12, 13, 21, 29].

For configuration as program you need to execute them first.

Mounting

Mounting integrates a backend into the key database [20]. Hence, Elektra allows several backends to deal with configuration files at the same time. Each backend is responsible for its own subtree of the key database.

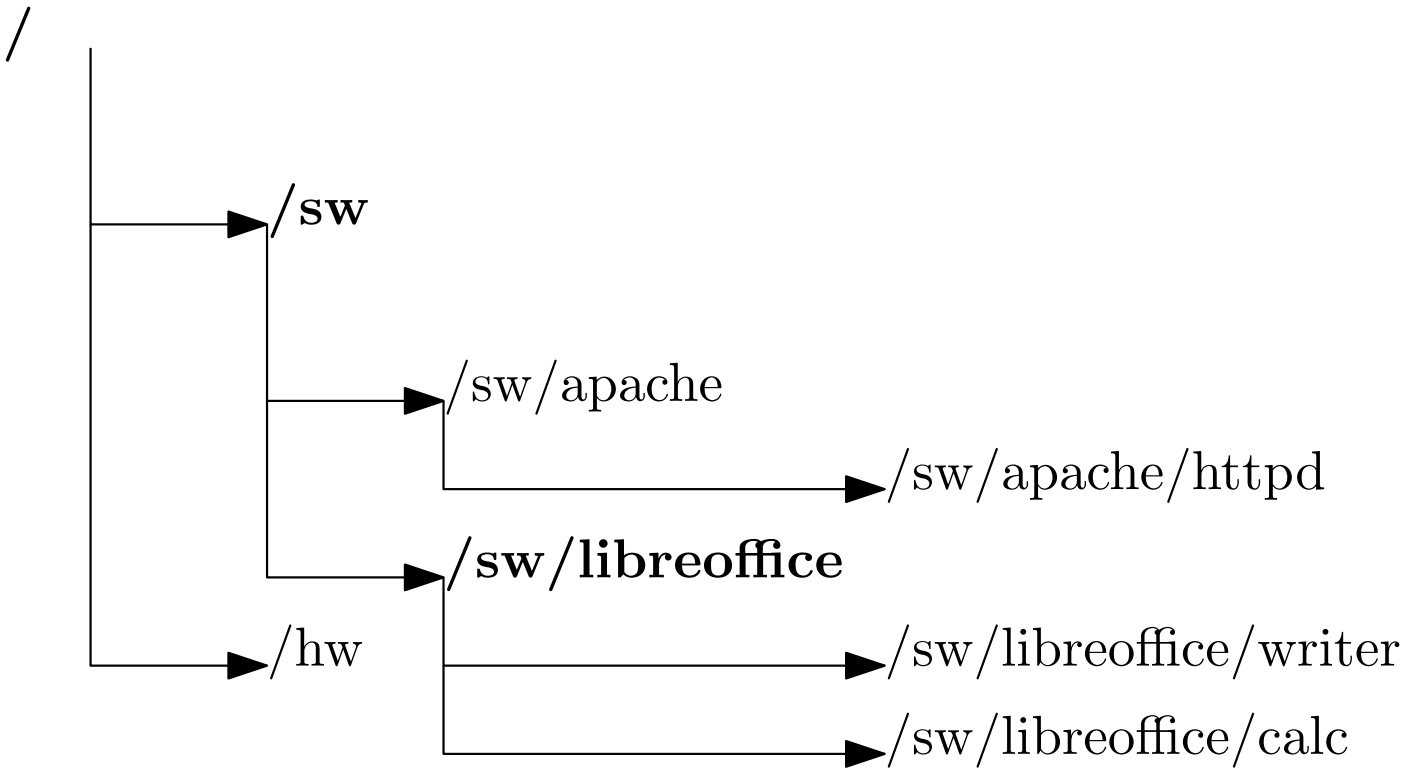


Plugins

Different backends can use different plugins:

`/sw` in the INI file config.ini

`/sw/libreoffice` in the XML file libreoffice.xml



Task

Possible Homework: Implement a storage plugin with existing parser.

Task

Explain your neighbor what mounting is.

Command-line Arguments

1. Configuration File Formats
   - Definitions
   - Formats
   - Abstractions
2. Command-line Arguments
   - Usage and Popularity
   - Semantics
3. Environment Variables
   - Trends
   - Requirements
   - Conclusion

Is there something else?

- configuration files are the most researched of all configuration sources [12]
- but it is neither the most used nor most popular [18]

Q: "Which configuration systems/libraries/APIs have you already used or would like to use in one of your FLOSS project(s)?"

- command-line arguments (92 %, $n = 222$)
- environment variables (79 %, $n = 218$)
- *S:* API `getenv` is used omnipresently with 2,683 occurrences
- configuration files (74 %, $n = 218$))

Q: "What is your experience with the following configuration systems/libraries/APIs?"

- `getenv` (10 %, $n = 198$)
- configuration files (6 %, $n = 190$)
- command-line options (4 %, $n = 210$)
- X/Q/GSettings (41 %, 14 %, 35 %)
- KConfig (21 %)
- dconf (42 %)
- plist (32 %)
- Windows Registry (69 %)

Task

Which configuration source do you use most?

Task

Possible talk: About one of these sources.

- passed by main for a new process via (`int argc, char ** argv`)
- visible from other processes (e.g., via `ps aux`)
- could be passed along to subprocesses but hardly done
- need to be parsed by process
- portability: differences in parsing
- cannot be changed from outside (requires restart, no IPC)

Environment Variables

1. Configuration File Formats
   - Definitions
   - Formats
   - Abstractions
2. Command-line Arguments
   - Usage and Popularity
   - Semantics
3. Environment Variables
   - Trends
   - Requirements
   - Conclusion

Semantics

- are also per-process (`/proc/self/environ`)
- are not visible from other processes
- are automatically inherited by subprocesses
- need to be parsed by process (`[extern] char **environ`) but API is provided (`getenv`)
- cannot be changed from outside (requires restart, no IPC)

Task

What is wrong with the code in the book?

getenv

- is widely standardized, including SVr4, POSIX.1-2001, 4.3BSD, C89, C99 [1],
- is supported by many programming languages, and
- enforces `key=value` convention.

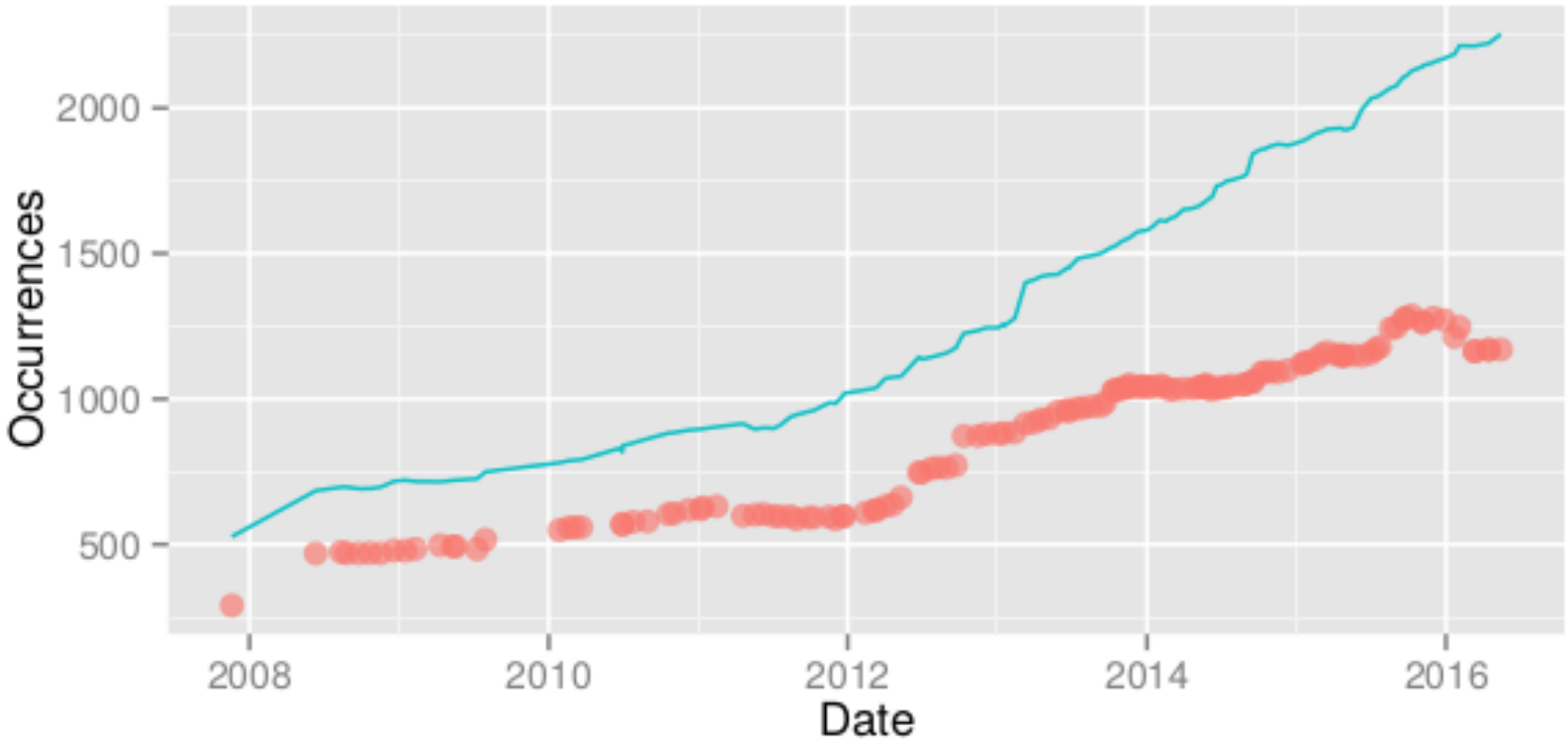
Usage

1. bypassing other configuration accesses (*Q:* 45 %)
2. locating configuration files
3. debugging and testing (*Q:* 55 %, *S:* 1,152, i. e. 43 %)
4. sharing configuration settings across applications (*Q:* 53 %, *S:* 716, i. e. 47 %)
5. for configuration settings unlikely to be changed by a user (*Q:* 20 %)
6. *"even when it is used inside a loop"* (*Q:* 2 %)

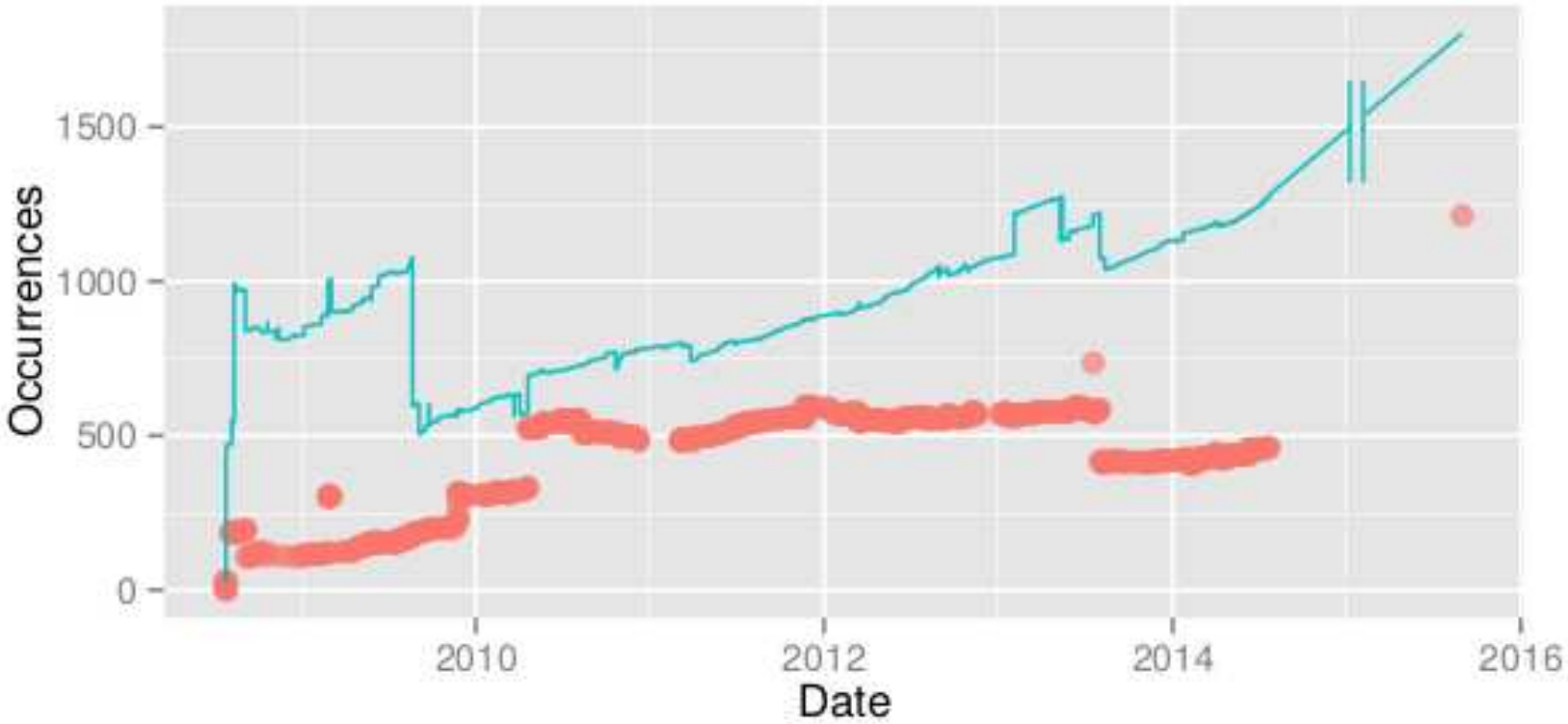
Portability

- no separators for values defined
- case sensitivity problems
- often many environment variables for the same purpose: TMP, TEMP, or TMPDIR
- sometimes one environment variable for different purposes: PATH

Trend Firefox



Trend Chromium

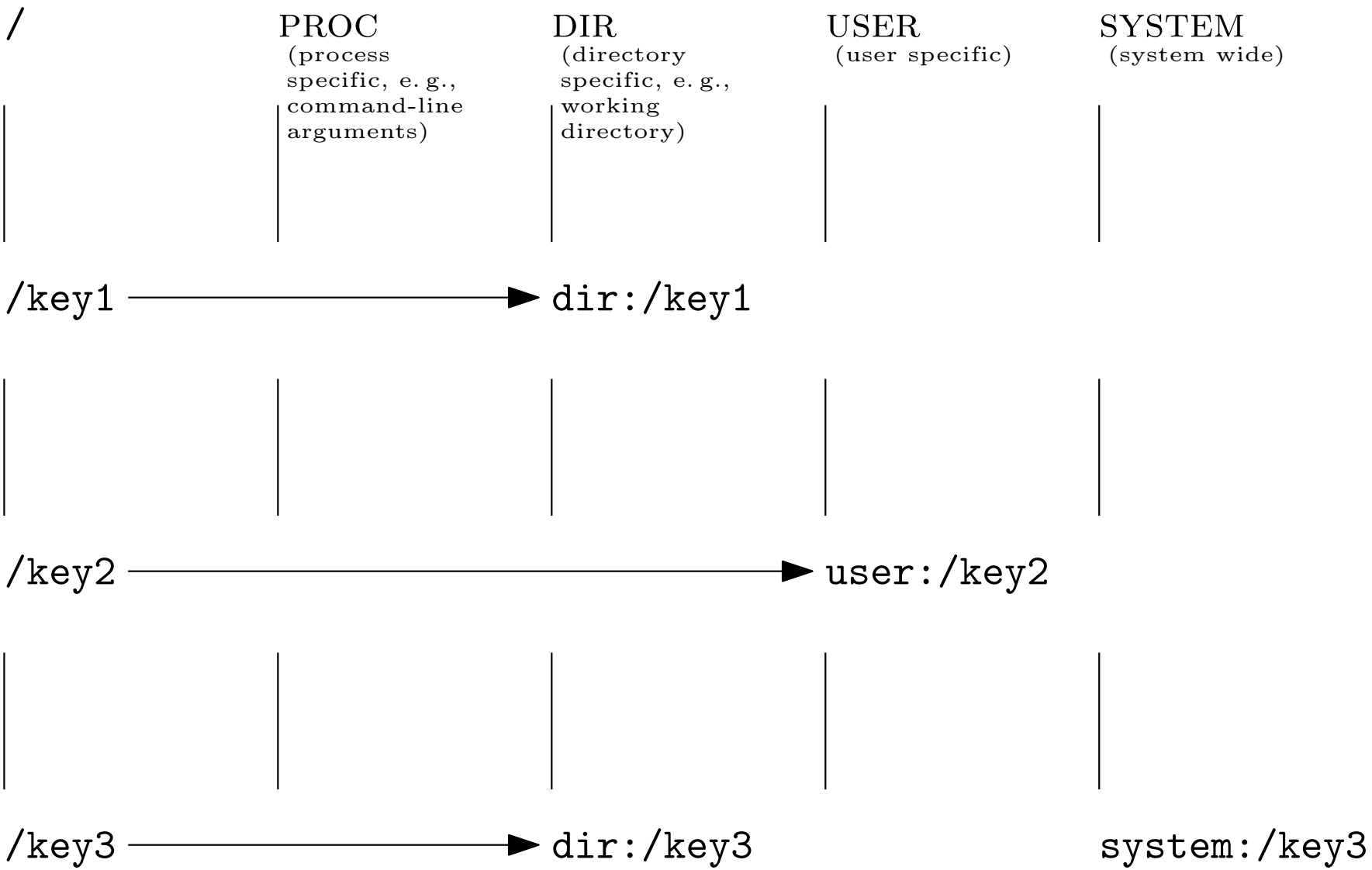


How can we deal with the many sources?

Requirement

A configuration library must support all three popular ways for configuration access: configuration files, command-line options, and environment variables.

Cascading


/
PROC
(process specific, e. g., command-line arguments)
DIR
(directory specific, e. g., working directory)
USER
(user specific)
SYSTEM
(system wide)
/key1
dir:/key1
/key2
user:/key2
/key3
dir:/key3
system:/key3

Task

Discuss the differences of mounting and cascading with your neighbor.

User View

- command-line for trying out configuration settings
- environment variables for configuration settings within a shell
- configuration files for persistent configuration settings

Conclusion

- three different configuration sources widely used
- all three used for different reasons but often for the same configuration settings
- many different configuration file formats
- abstractions: key-value, mounting, and cascading

[1] *getenv(3) Linux User's Manual*, March 2017.

[2] Eric Arnold Anderson. *Researching system administration*. PhD thesis, University of California at Berkeley, 2002.

[3] Paul Anderson. Towards a high-level machine configuration system. In *LISA*, volume 94, pages 19–26, 1994.

[4] Richard Anthony, DeJiu Chen, Mariusz Pelc, Magnus Persson, and Martin Törngren. Context-aware adaptation in DySCAS. *Electronic Communications of the EASST*, 19, 2009. URL `http://gala.gre.ac.uk/5533/`.

[5] Lionel Cons and Piotr Poznanski. Pan: A high-level configuration language. In *LISA*, volume 2, pages 83–98, 2002. URL `http://static.usenix.org/events/lisa02/tech/full_papers/cons/cons_html/`.

[6] Fernando J. Corbató, Fernando H. Saltzer, and C. T. Clingen. Multics: The first seven years. In *Proceedings of the May 16-18, 1972, Spring Joint Computer Conference*, AFIPS '72 (Spring), pages 571–583, New York, NY, USA, 1972. ACM. doi: 10.1145/1478873.1478950. URL http://dx.doi.org/10.1145/1478873.1478950.

[7] Software Productivity Consortium Services Corporation. *Reuse-driven Software Processes Guidebook: SPC-92019-CMC, Version 02.00. 03*. Software Productivity Consortium Services Corporation, 1993.

[8] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wąsowski. Cool features and tough decisions: A comparison of variability modeling approaches. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, VaMoS '12, pages 173–182, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1058-1. doi: 10.1145/2110147.2110167. URL http://dx.doi.org/10.1145/2110147.2110167.

[9] Ian Goldberg, David Wagner, Randi Thomas, and Eric A. Brewer. A secure environment for untrusted helper applications: Confining the wily hacker. In *Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography*, volume 6, pages 1–1, 1996.

[10] Sebastian Günther, Thomas Cleenewerck, and Viviane Jonckers. Software variability: the design space of configuration languages. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, pages 157–164. ACM, 2012. URL `http://dl.acm.org/citation.cfm?id=2110165`.

[11] Peng Huang, William J. Bolosky, Abhishek Singh, and Yuanyuan Zhou. ConfValley: a systematic configuration validation framework for cloud services. In *EuroSys*, page 19, 2015.

[12] Dongpu Jin, Xiao Qu, Myra B. Cohen, and Brian Robinson. Configurations everywhere: Implications for testing and debugging in practice. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 215–224, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2768-8. doi: 10.1145/2591062.2591191. URL http://dx.doi.org/10.1145/2591062.2591191.

[13] Neal Lathia, Kiran Rachuri, Cecilia Mascolo, and George Roussos. Open source smartphone libraries for computational social science. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp '13 Adjunct, pages 911–920, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2215-7. doi: 10.1145/2494091.2497345. URL http://dx.doi.org/10.1145/2494091.2497345.

[14] Zhenkai Liang, V. N. Venkatakrishnan, and R. Sekar. Isolated program execution: an application transparent approach for executing untrusted programs. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, pages 182–191, December 2003. doi: 10.1109/CSAC.2003.1254323.

[15] Kim Mens, Rafael Capilla, Nicolas Cardozo, Bruno Dumas, et al. A taxonomy of context-aware software variability approaches. In *Workshop on Live Adaptation of Software Systems, collocated with Modularity 2016 conference*, 2016.

[16] Sarah Nadi, Thorsten Berger, Christian Kästner, and Krzysztof Czarnecki. Mining configuration constraints: static analyses and empirical results. In *ICSE*, pages 140–151, 2014.

[17] Jeff H. Perkins, Sunghun Kim, Sam Larsen, Saman Amarasinghe, Jonathan Bachrach, Michael Carbin, Carlos Pacheco, Frank Sherwood, Stelios Sidiroglou, Greg Sullivan, Weng-Fai Wong, Yoav Zibin, Michael D. Ernst, and Martin Rinard. Automatically patching errors in deployed software. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 87–102, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-752-3. doi: 10.1145/1629575.1629585. URL http://dx.doi.org/10.1145/1629575.1629585.

[18] Markus Raab and Gergö Barany. *Challenges in Validating FLOSS Configuration*, pages 101–114. Springer International Publishing, Cham, 2017. ISBN 978-3-319-57735-7. doi: 10.1007/978-3-319-57735-7_11. URL http://dx.doi.org/10.1007/978-3-319-57735-7_11.

[19] Markus Raab and Gergö Barany. Introducing context awareness in unmodified, context-unaware software. In *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE,*, pages 218–225. INSTICC, ScitePress, 2017. ISBN 978-989-758-250-9. doi: 10.5220/0006326602180225.

[20] Markus Raab and Patrick Sabin. Implementation of Multiple Key Databases for Shared Configuration. `ftp://www.markus-raab.org/elektra.pdf`, March 2008. Accessed February 2014.

[21] Ariel Rabkin and Randy Katz. Static extraction of program configuration options. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 131–140. IEEE, 2011.

[22] Z. Cliffe Schreuders, Tanya Jane McGill, and Christian Payne. Towards usable application-oriented access controls: qualitative results from a usability study of SELinux, AppArmor and FBAC-LSM. *International Journal of Information Security and Privacy*, 6(1):57–76, 2012.

[23] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *First International Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp 2004*, September 2004. URL `http://elib.dlr.de/7444/`.

[24] Sander van der Burg and Eelco Dolstra. Automating system tests using declarative virtual machines. In *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on*, pages 181–190. IEEE, 2010.

[25] Jilles Van Gurp, Jan Bosch, and Mikael Svahnberg. On the notion of variability in software product lines. In *Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference on*, pages 45–54. IEEE, 2001.

[26] Karina Villela, Adeline Silva, Tassio Vale, and Eduardo Santana de Almeida. A survey on software variability management approaches. In *Proceedings of the 18th International Software Product Line Conference - Volume 1*, SPLC '14, pages 147–156, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2740-4. doi: 10.1145/2648511.2648527. URL http://dx.doi.org/10.1145/2648511.2648527.

[27] Alexander von Rhein, Thomas Thüm, Ina Schaefer, Jörg Liebig, and Sven Apel. Variability encoding: From compile-time to load-time variability. *Journal of Logical and Algebraic Methods in Programming*, 85(1, Part 2):125–145, 2016. ISSN 2352-2208. doi: http://dx.doi.org/10.1016/j.jlamp.2015.06.007. URL `http://www.sciencedirect.com/science/article/pii/S2352220815000577`. Formal Methods for Software Product Line Engineering.

[28] Huai Wang and Wing Kwong Chan. Weaving context sensitivity into test suite construction. In *Automated Software Engineering, 2009. ASE '09. 24th IEEE/ACM International Conference on*, pages 610–614, November 2009. doi: 10.1109/ASE.2009.79.

[29] Tianyin Xu, Jiaqi Zhang, Peng Huang, Jing Zheng, Tianwei Sheng, Ding Yuan, Yuanyuan Zhou, and Shankar Pasupathy. Do not blame users for misconfigurations. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 244–259. ACM, 2013.

[30] Zuoning Yin, Xiao Ma, Jing Zheng, Yuanyuan Zhou, Lakshmi N. Bairavasundaram, and Shankar Pasupathy. An empirical study on configuration errors in commercial and open source systems. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 159–172, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0977-6. doi: 10.1145/2043556.2043572.

[31] Sai Zhang and Michael D. Ernst. Automated diagnosis of software configuration errors. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 312–321, Piscataway, NJ, USA, 2013. IEEE Press. ISBN 978-1-4673-3076-3.

[32] Sai Zhang and Michael D. Ernst. Which configuration option should I change? In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 152–163, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2756-5. doi: 10.1145/2568225.2568251. URL http://dx.doi.org/10.1145/2568225.2568251.