Hiveconf - A UNIX configuration framework

Peter Åstrand,
Cendio Systems AB

Windows registry

- Good
  - Provides simple API to applications
  - Easy to build GUIs
  - Provides storage for configuration values in tree structure

Windows registry

- Bad
  - Data is stored in few, binary files
    - If the ”database” gets corrupted, the entire system is affected.
    - Hard to backup and restore settings
  - Over used: Used for too many things
  - Few data types
  - No real network support

Windows registry

- Bad
  - No support for meta data, ie information about the data type, valid values etc
  - Uses the file system too little: Does not use file permissions, for example
  - No support for alternative backends

UNIX text files

- Good
  - Flexible. Can use common text processing tools like M4, awk etc
  - Safe. If parts of the configuration file is broken, the remaining part can probably be reused
  - Flexible. The administratior can use their favorite tools & editors
  - ”One file per application”

UNIX text files

- Bad
  - Application programmers must re-invent the wheel every time: implement parsing etc
  - No common file format
  - Hard to make changes mechanically
    - Hard to build GUIs, especially if you want to preserve comments and formatting. Example: Samba/SWAT

UNIX text files

- Bad
  - Since applications must handle configuration themselves, only a minimum of features is usually supported. For example, very few applications supports storing configuration using a LDAP server

GConf

- Good
  - Provides API
  - Several data types
  - Support for meta data
  - ”Callbacks” are useful for desktop environment
  - Configuration data is stored (by default) in text files, approximately one per application
  - Support for alternative backends

GConf

- Bad
  - A per-user ”configuration daemon” is required. When it's started, no one else must access the configuration files
    - Lots of troubles when logging in on two machines at the same time, for example
  - Unusable for system services!
  - Default backend is XML, which is not human-friendly, IMHO
  - Does not preserve comments & structure in configuration files

GConf

- Bad
  - Machine-wide namespace
    - Not possible to run multiple instances of the same application with different settings
  - Requires ”schemas”, ie meta data

Hiveconf

- Tries to combine good properties from other systems
  - Provides API for storing configuration in tree structure
  - Easy to make changes mechanically
    - Easy to write GUIs
  - Configuration can be stored in <1, 1 or >1 files per application
  - Support for alternative backends
    - The default backend uses INI-like text files

Hiveconf

- Provides support for meta data, but does not enforce the use of meta data
- Uses the file system
  - Ordinary file permissions can be used
  - Possible future enhancements: Version control using RCS

Hiveconf

- If the API provides a file system like name space, why is a configuration framework needed at all? Why not use the file system directly, and store every parameter in a new file?
  - Bad:
    - Inefficient with many small files (but ReiserFS might help)
    - Impractical: Needs to open lots of files in text editors
    - Might be harder to version control

Hiveconf

- Good:
  - Fine-grained permissions
  - Clean and simple
- Hiveconf has a ”filesystem backend”, which supports storing parameter values in files
- Different parameters can be stored with different backends

Hiveconf

- Current implementation (2003-06-21)
  - Python module
    - Usable for application programmers
  - Python tool (”hivetool”)
    - Useable as a tool for configuring Samba, KDE etc
    - Idea similiar to gconftool
- Future
  - Implement C library
  - More backends
  - World domination...

?