XDG Base Directory Specification

Waldo Bastian

<bastian@kde.org>

Ryan Lortie

<desrt@desrt.ca>

Lennart Poettering

<lennart@poettering.net>

Version 0.7

24th November 2010

Table of Contents

Introduction

Various specifications specify files and file formats. This specification defines where these files should be looked for by defining one or more base directories relative to which files should be located.

Basics

The XDG Base Directory Specification is based on the following concepts:

- There is a single base directory relative to which user-specific data files should be written. This directory is defined by the environment variable `$XDG_DATA_HOME`.

- There is a single base directory relative to which user-specific configuration files should be written. This directory is defined by the environment variable `$XDG_CONFIG_HOME`.

- There is a set of preference ordered base directories relative to which data files should be searched. This set of directories is defined by the environment variable `$XDG_DATA_DIRS`.

- There is a set of preference ordered base directories relative to which configuration files should be searched. This set of directories is defined by the environment variable `$XDG_CONFIG_DIRS`.

- There is a single base directory relative to which user-specific non-essential (cached) data should be written. This directory is defined by the environment variable `$XDG_CACHE_HOME`.

- There is a single base directory relative to which user-specific runtime files and other file objects should be placed. This directory is defined by the environment variable `$XDG_RUNTIME_DIR`.

All paths set in these environment variables must be absolute. If an implementation encounters a relative path in any of these variables it should consider the path invalid and ignore it.

Environment variables

`$XDG_DATA_HOME` defines the base directory relative to which user specific data files should be stored. If `$XDG_DATA_HOME` is either not set or empty, a default equal to`$HOME`/.local/share should be used.

`$XDG_CONFIG_HOME` defines the base directory relative to which user specific configuration files should be stored. If `$XDG_CONFIG_HOME` is either not set or empty, a default equal to `$HOME`/.config should be used.

`$XDG_DATA_DIRS` defines the preference-ordered set of base directories to search for data files in addition to the `$XDG_DATA_HOME` base directory. The directories in `$XDG_DATA_DIRS` should be seperated with a colon ':'.

If `$XDG_DATA_DIRS` is either not set or empty, a value equal to /usr/local/share/:/usr/share/ should be used.

`$XDG_CONFIG_DIRS` defines the preference-ordered set of base directories to search for configuration files in addition to the `$XDG_CONFIG_HOME` base directory. The

directories in `$XDG_CONFIG_DIRS` should be seperated with a colon ':'.

If `$XDG_CONFIG_DIRS` is either not set or empty, a value equal to /etc/xdg should be used.

The order of base directories denotes their importance; the first directory listed is the most important. When the same information is defined in multiple places the information defined relative to the more important base directory takes precedent. The base directory defined by `$XDG_DATA_HOME` is considered more important than any of the base directories defined by `$XDG_DATA_DIRS`. The base directory defined by `$XDG_CONFIG_HOME` is considered more important than any of the base directories defined by `$XDG_CONFIG_DIRS`.

`$XDG_CACHE_HOME` defines the base directory relative to which user specific non-essential data files should be stored. If `$XDG_CACHE_HOME` is either not set or empty, a default equal to `$HOME`/.cache should be used.

`$XDG_RUNTIME_DIR` defines the base directory relative to which user-specific non-essential runtime files and other file objects (such as sockets, named pipes, ...) should be stored. The directory MUST be owned by the user, and he MUST be the only one having read and write access to it. Its Unix access mode MUST be 0700.

The lifetime of the directory MUST be bound to the user being logged in. It MUST be created when the user first logs in and if the user fully logs out the directory MUST be removed. If the user logs in more than once he should get pointed to the same directory, and it is mandatory that the directory continues to exist from his first login to his last logout on the system, and not removed in between. Files in the directory MUST not survive reboot or a full logout/login cycle.

The directory MUST be on a local file system and not shared with any other system. The directory MUST by fully-featured by the standards of the operating system. More specifically, on Unix-like operating systems AF_UNIX sockets, symbolic links, hard links, proper permissions, file locking, sparse files, memory mapping, file change notifications, a reliable hard link count must be supported, and no restrictions on the file name character set should be imposed. Files in this directory MAY be subjected to periodic clean-up. To ensure that your files are not removed, they should have their access time timestamp modified at least once every 6 hours of monotonic time or the 'sticky' bit should be set on the file.

If `$XDG_RUNTIME_DIR` is not set applications should fall back to a replacement directory with similar capabilities and print a warning message. Applications should use this directory for communication and synchronization purposes and should not place larger files in it, since it might reside in runtime memory and cannot necessarily be swapped out to disk.

Referencing this specification

Other specifications may reference this specification by specifying the location of a data file as `$XDG_DATA_DIRS`/subdir/filename. This implies that:

- Such file should be installed to `$datadir`/subdir/filename with `$datadir`defaulting to /usr/share.
- A user specific version of the data file may be created in`$XDG_DATA_HOME`/subdir/filename, taking into account the default value for `$XDG_DATA_HOME` if `$XDG_DATA_HOME` is not set.
- Lookups of the data file should search for ./subdir/filename relative to all base directories specified by `$XDG_DATA_HOME` and `$XDG_DATA_DIRS` . If an environment variable is either not set or empty, its default value as defined by this specification should be used instead.

Specifications may reference this specification by specifying the location of a configuration file as `$XDG_CONFIG_DIRS`/subdir/filename. This implies that:

- Default configuration files should be installed to `$sysconfdir`/xdg/subdir/filename with `$sysconfdir` defaulting to /etc.
- A user specific version of the configuration file may be created in`$XDG_CONFIG_HOME`/subdir/filename, taking into account the default value for `$XDG_CONFIG_HOME` if `$XDG_CONFIG_HOME` is not set.
- Lookups of the configuration file should search for ./subdir/filename relative to all base directories indicated by `$XDG_CONFIG_HOME` and `$XDG_CONFIG_DIRS` . If an environment variable is either not set or empty, its default value as defined by this specification should be used instead.

If, when attempting to write a file, the destination directory is non-existant an attempt should be made to create it with permission `0700`. If the destination directory exists already the permissions should not be changed. The application should be prepared to handle the case where the file could not be written, either because the directory was non-existant and could not be created, or for any other reason. In such case it may chose to present an error message to the user.

When attempting to read a file, if for any reason a file in a certain directory is unaccessible, e.g. because the directory is non-existant, the file is non-existant or the user is not authorized to open the file, then the processing of the file in that directory should be skipped. If due to this a required file could not be found at all, the application may chose to present an error message to the user.

A specification that refers to `$XDG_DATA_DIRS` or `$XDG_CONFIG_DIRS` should define what the behaviour must be when a file is located under multiple base directories. It could, for example, define that only the file under the most important base directory should be used or, as another example, it could define rules for merging the information from the different files.