NAME

software−wishlist − wishes (and requirements) by TiU LIS Unix for supported software

INTRODUCTION

The intended audience for this document is software developers, as well as people intermediating with software developers.

This document lists wishes and requirements for software, for it to be installed on TiU LIS Unix managed systems. For specific software installations, requirements are decided upon on a case by case basis. However, this list will be used as a guideline.

This list applies to *all* software running on LIS Unix managed systems: the OS vendor supplied kernel, Oracle product installations, PHP and Perl webapplications, in-house developed small scripts, and all other stuff.

The list is sorted: the first wish is a hard requirement always, and is "easy" to fulfill. The last wish would make LIS Unix very happy, but is quite often impossible to fulfill. Of course, the more items your software adheres to, the higher the service level LIS Unix can guarantee, the happier everybody will be.

This document is published at: <http://non−gnu.uvt.nl/pub/uvt−unix−doc/software−wishlist/software−wishlist> .

WISHES AND REQUIREMENTS

1. maintained

The software needs to be maintained. Popular Unix Operating Systems change, expectations of SysAdmins and users change, new ways to attack and abuse software get found and used: the environment of each software package changes, so all software packages will have to change too. For newly fixed software versions and releases, a agreed upon channel should be established to LIS Unix, so that upgrades can get carried out in time.

2. version numbers

Each version of the software package has a number by which it can be easily identified. This helps when referring to specific installations: it should be possible to make it clear exactly about which software one is talking. This could simply be the date it's released, written as YYYYMMDD. The version number could be implicit, e.g. when LIS Unix has access to the version control system used to develop the software a thing like an SVN revision number could be used.

In case the software is offered in a filearchive (e.g. .zip, .tar.gz, .txz), the filearchive must be published on a secured location, e.g. a httpS URL or uploaded to an scp-reachable location (one option is to use https://send.uvt.nl). The archive should be named *name−version.extension*, and should extract to a directory named *name−version/* . E.g. hello−1.2.tar.gz contains the file hello−1.2/INSTALL .

3. readonly install

If the software writes to specific files (state data), these files should be under a common root in the filesystem, separate from the installation root. The software could write to:

```
/var/lib/foobar/*
```

While other installed files are typically:

```
/usr/bin/foobar
/usr/share/foobar/*
```

It should be possible to install the software in such a way that the user ID typically running the software has no way to change the program itself.

4. configuration

As with state data, configuration data should be kept in a specific directory. (Of course, all data the application maintainer might want to change, should be kept in run-time configuration files.) This is needed to separate Unix users having write access to configuration data, from users having write access to the software installation.

Preferably, all configuration should be stored in human-editable text files. If the application needs any passwords (or other secret data like ssh private keys), it should be possible to store these outside the main configuration files.

5. syslog

Logging should go to syslog. Use the libc *syslog*(3) call, or use a wrapper supplied by the scripting language you use. (Python, Perl and PHP all have interfaces for this.) If you'd like to be able to write logging to stuff to STDERR, you might want to choose to wrap you application in *logger*(1). It should be possible to specify which syslog "facility" identifier is used.

This is needed to efficiently deal with logrotates, to flexibly decide which logdata should go where (possibly to a separate loghost, for example), and to be able to plug the software in the LIS Unix maintained logfile monitoring system. Using this monitoring framework, LIS Unix can e.g. redirect loggged error conditions to destinations supplied by the application maintainer.

6. email

If the software sends out emails, the used envelope From-adress as well as header From-adress should be configurable. The software should use *sendmail*(1) to send mail. If this is not possible, the software should talk to localhost:smtp using the SMTP protocol.

7. cronjobs

If the software requires cronjobs to be run, it should not assume the installing user runs "crontab −e". It should be possible to use /etc/cron.d/*appname*, as supported by the *cron*(8) that is shipped with popular GNU/Linux distributions.

8. daemon

Every Unix daemon process is supposed to "deamonize" itself properly, preferably using the libc *daemon*(3) call. If your programming environment does not support this libc call, you'll need to do a number of things manually. You can find a description in <http://www.enderunix.org/documents/eng/daemon.php> .

9. IPv6

Every application that uses the network should work with IPv6. Applications must not fail if IPv6 is present.

10. INSTALL

The software should come with usable installations instructions (typically in a file named INSTALL). (Some software doesn't need such a file: if a usable Makefile or ./configure is shipped, installation can be straightforward. See below.)

11. makefiles

The software should come with Makefiles, which act in a sane way when invoked as "make install". Caspar <http://mdcc.cx/caspar/> makes it easy to add support for this to your software. (For larger projects, you might consider using Automake.)

See the GNU coding standards <http://www.gnu.org/prep/standards/standards.html#Standard-Targets> for a description of what a sane "make install" should do.

12. stow-able

It should be possible to install under a prefix of ones choice. An example of an installation system that does this out of the box is autoconf.

13. FHS

A default install should conform to the FHS <http://www.pathname.com/fhs/>.

14. [dropped]

[this section is dropped (was: LSB).]

15. packageable

The software's build and installation system should have support for DESTDIR. An example of an installation system that does this out of the box is the GNU Autotools.

16. ChangeLog

The software should ship with a NEWS file, or ChangeLog, listing the major changes between releases.

17. packaged

The software is packaged in packages that conform to the Debian Policy <http://www.debian.org/doc/debian-policy/>, so that it integrates nicely with the Debian GNU/Linux Operating System.

18. Debian APT repository

The aforementioned packages are shipped using a Debian APT repository, e.g. either the one running at <https://non-gnu.uvt.nl/>, or the one running at <https://non-free.uvt.nl/>.

19. Debian main

The aforementioned APT repository is the Debian main APT repository. This is the ideal situation.

SEE ALSO

The wishlist for Web Applications <https://doc.uvt.nl/its-external/system-group/web-app/webapp-wishlist.html> (also available from <https://tiu.nu/webapp-wishlist>) these are additional requirements specific to websites and other web-based applications.

The GNU coding standards <http://www.gnu.org/prep/standards/> . The GNU hello software package at http://www.gnu.org/software/hello/ : an example of best practices. The Debian package of GNU Hello <http://packages.debian.org/hello-debhelper> .

The Debian Upstream Guide <http://wiki.debian.org/UpstreamGuide> , explaining how to make live easy for Debian packagers when writing your software.

COPYRIGHT

AUTHORS

Joost van Baal <joostvb@uvt.nl>

Wessel Dankers <wsl@uvt.nl>

VERSION

```
$Id: software-wishlist.pod 46445 2017-03-27 09:21:13Z joostvb $
$URL: https://svn.uvt.nl/its-id/trunk/sources/uvt-unix-doc/software-wishlist/sof
```