

Skulker Version 2 - Installation Guide

Version: 1.3
Date: 20th December, 2007.

Version History

Version	Date	Author	Changes
1.0	12 th December, 2006.	Simon Edwards	Original.
1.1	8 th June, 2007.	Simon Edwards	Improved documentation.
1.2	22 nd August, 2007.	Simon Edwards	Information regarding protected directory configuration.
1.3	20 th December, 2007.	Simon Edwards	History + html reporting configuration requirements.

Contents

1	INTRODUCTION.....	3
1.1	WHAT IS SKULKER?.....	3
1.2	3
1.3	WHAT IS DIFFERENT IN VERSION 2?.....	4
1.4	WHO SHOULD READ THIS DOCUMENT?.....	4
2	INSTALLATION	5
2.1	RECOMMENDED MINIMUM VERSION	5
2.2	SOFTWARE PACKAGE AVAILABILITY	5
2.3	INSTALLATION INSTRUCTIONS FOR HP-UX (SDUX PACKAGE)	5
2.4	INSTALLATION INSTRUCTIONS FOR SOLARIS (PKG PACKAGE).....	6
2.5	INSTALLATION INSTRUCTIONS FOR LINUX (RPM-BASED DISTRIBUTIONS)	6
2.6	INSTALLATION INSTRUCTIONS FOR LINUX (AUTOPACKAGE ENVIRONMENTS).....	6
2.7	INSTALLATION INSTRUCTIONS FOR LINUX (SLACKWARE-BASED DISTRIBUTIONS)	6
2.8	INSTALLATION INSTRUCTIONS FOR GENERIC UNIX (TP2 PACKAGE).....	7
3	POST INSTALLATION TASKS.....	8
3.1	BASIC SOFTWARE CONFIGURATION	8
3.2	VALIDATE BASE RULE SET.....	13
3.3	INITIAL RULE SET STATISTICS GENERATION	13
3.4	INITIAL DRY RUN	14
3.5	AUTOMATED SCHEDULING	14
3.6	RULE SET CUSTOMISATION	14

1 Introduction

1.1 What is Skulker?

“Skulker” is a tool that has been designed to automatically manage many files that either grow in size or number over a period of time whilst the Operating System runs and applies a series of actions against these files to ensure the disk space they occupy is automatically managed.

Although this may sound a trivial requirement a modern UNIX-like Operating System tends to have a large number of logs files that are written on a regular basis. Since such an Operating system is designed to run for prolonged periods without downtime, unless these logs are managed they can grow to substantial sizes. Skulker relieves the administrator of such machines the requirement for tidying logs manually.

Skulker provides a number of different mechanisms (known as “functions”) for acting on a log file, and works with various compression algorithms if necessary. It has been designed to allow the administrator to add additional mechanisms if designed for particular environments or applications. As of version 2.0, the following functions are provided as part of the tool:

- command - run a specified command
- compress - compress a specified file
- delete - delete a specified file
- deletedir - delete a specified directory
- mailfile - modify a specified mail file
- organise - move specified file
- rotate - rename specified file
- archive - move files into collective archive files
- tail - manipulate specified file

Also supplied are the following application specific functions to act as working examples:

- saswork - Remove specified SAS¹ work directory
- spdswork - Remove specified SAS SPDS temporary file

Skulker provides functionality to ensure that the target files are dealt with only on specific frequencies and also under specific characteristics if necessary (such as size of the file, or the time of last modification).

Further Skulker handles log files that are open for reading in an intelligent manner meaning that applications or daemons do not need to be shutdown for log management to occur - obviously important!

Finally the tool provides options to summarise the effectiveness of the last run, or even all runs ever made, and provides many logging levels to ensure the administrator is always aware of the actions that a given run of the tool performed.

¹ SAS is copyright of the SAS Institute Inc.

1.2 What is Different in Version 2?

Version 1.0.0 was first written in 1999 and still is live in many environments today. At that time many commercial versions of UNIX made little use of Open Source software, meaning that Skulker was written with only commonly available POSIX tools - essentially things like “ksh” (the Korn shell) and “awk”.

The software available in 2006 is quite different - almost all commercial UNIX variants now come with Open Source software, much of which is considered part of any “default” installation. The Perl programming language is inevitably part of that software and given the performance and power of that language is made sense to take advantage of it.

Thus version 2 of Skulker is completely written in Perl - making calls to external programs to handle compression or decompression when necessary. By migrating to a Perl-based solution not only has it been possible to greatly increase the performance of the software, the handling of software configuration has changed as well - all configuration files are now XML based. This means that the syntax has been simplified whilst making use of commonly used file formats.

Apart from supporting the “organise” method the functionality of the software is fairly similar to the first version. The major difference is the handling of files that might be open in a sensible manner. This is a major advantage compared to the previous version.

Finally the summaries provided of disk space reclaimed by running the utility are now completely accurate. The first version of the software estimated this reclaim by comparing the space reported by the disk free command (“df” or equivalent) before any changes and after all changes was complete. On a system with significant activity this is invariably incorrect. Version 2 computes this information for every file that is changed or removed and so is independent of other activity running on the host at the same time.

Version 2.0 also has a DB style database which contains the attributes of when each rule is run, and the run characteristics of the last run, or all runs - thus allowing reasonably detailed reporting information to be extracted - independently of actually running the software.

1.3 Who Should Read this Document?

This document is aimed at System Administrators wish to install the product in the default configuration - which contains the necessary rules to manage standard operating system files.

This “out-of-the-box” configuration is a suitable starting point for many configurations.

2 Installation

2.1 Recommended Minimum Version

The recommended minimum installation of Skulker V2 is “0.2.2”. Despite the low version number the software is stable, performs well and has no known bugs. The low version number is simply because some features desired by the authors are not yet included.

2.2 Software Package Availability

The software is available in a range of package formats, including the following:

- SDUX - SDUX Package format (for HP-UX)
- Pkg - SVR4 Package format (for Solaris)
- RPM - RedHat Package (RPM-based Linux distributions)
- tp2 - TP2 package (OS neutral) package
- Autopackage - Linux Autopackage (distribution neutral) package
- Slackpkg - Slackware package format (for Slackware Linux)
- Debian - Suitable for Linux distributions using Debian-based packages

Specific installation instructions for various package formats and UNIX types can be found below. If no specific package is available for your platform, please consider using the “tp2” package which is a packaging infrastructure also making use of Perl. For information regarding TP2 see the following web page:

<http://www.linuxha.net/tp2>

All documentation and latest versions of this software are available from the following web site:

<http://www.linuxha.net/skulker2>

2.3 Installation Instructions for HP-UX (SDUX Package)

The HP-UX package format will be default install the software into the following directory structure:

```
/opt/skulker2
```

If a customer specific package is available the path might be different and commonly is:

```
/opt/customer/skulker2
```

Installation of the software is a matter of running the “swinstall” command once the “.sdux” file has been copied to a given directory. For example if the software “skulker2-0.0.8,0-0-8,HP-UX.sdux” has been downloaded to “/tmp” locally the software can be installed using:

```
# swinstall -s /tmp/skulker2-0.0.8,0-0-8,HP-UX.sdux skulker2
```

In many environments the software might already have been placed in a local depot, and in that instance the following should suffice:

```
# swinstall -s host:/depot skulker2
```

2.4 Installation Instructions for Solaris (Pkg Package)

Details to be added soon!

2.5 Installation Instructions for AIX (Installp Package)

Details to be added soon!

2.6 Installation Instructions for Linux (RPM-based distributions)

The RPM package format will be default install the software into the following directory structure:

```
/opt/skulker2
```

Installation of the software is a matter of running the “rpm” command once the “.rpm” file has been copied to a given directory. For example if the software “skulker2-0.2.2-1.noarch.rpm” has been downloaded to “/tmp” locally the software can be installed using:

```
# rpm -Uvh /tmp/skulker2-0.2.2-1.noarch.rpm
```

Notice that since Skulker 2 is written in Perl the package is not architecture specific and can be installed on any hardware platform.

2.7 Installation Instructions for Linux (Autopackage environments)

Autopackage is a distribution neutral packaging toolset that supports command line and GUI installations. Installation of an autopackage is a matter of making the package file executable and then running it.

For example if the file has been downloaded as “/tmp/skulker2-0.2.2.x86.package” then the following commands could be used:

```
# cd /tmp
# chmod +x skulker2-0.2.2.x86.package
# ./skulker2-0.2.2.x86.package
```

If the Autopackage toolset is not installed the script will automatically install it as necessary.

2.8 Installation Instructions for Linux (Slackware-based distributions)

Assuming the package has been copied as “/tmp/skulker2-0.4.0-i486-1.tgz” then the following commands can be used to install it:

```
# cd /tmp
# installpkg skulker2-0.4.0-i486-1.tgz
```

2.9 Installation Instructions for Linux (Debian-based distributions)

Assuming the Skulker2 package is available in the current directory as “skulker2_0.4.0.deb” then simply issue the following command as root:

```
# dpkg --install skulker2_0.4.0.deb
```

2.10 Installation Instructions for Generic UNIX (TP2 Package)

TP2 is a particularly useful packaging toolset that works across different UNIX platforms, including Linux, HP-UX, Solaris and AIX. Because of the cross-platform nature of Skulker it makes sense to provide it as a TP2 package.

If you wish to consider using this packaging format the software, installation and usage instructions can be found via the following link:

http://linuxha.net/wcodemgr3/index.cgi/project_home?tp2

If the package has been downloaded in this format, say to “/tmp/skulker2+0.2.2.tp2”, then it can be installed in the following two steps:

```
# tp2 repos --repos /tmp
# tp2 install --namespace root --pkg skulker2 --verbose --repos /tmp
```

3 Post Installation Tasks

3.1 Basic Software Configuration

The software package deployed has been designed to require minimal changes before it can be run. The first step if this is a new installation is to create a directory for history and statistic information. Some files in this directory will grow over time, but should never amount of more than 1-2 megabytes. Create this directory as:

```
# mkdir /opt/skulker2/var
```

If Skulker has been installed elsewhere you might instead need to enter:

```
# mkdir /opt/customer/skulker2/var
```

Prior to running the software the only step is validate that the location of the log files is sensible for the current environment. By default this will default to “/var/adm” - which may or may not be appropriate for your environment. To change it edit the following file:

```
/opt/skulker2/cfg/cfg.xml
```

If you have a customer-specific package it is likely instead to reside in:

```
/opt/customer/skulker2/cfg/cfg.xml
```

In this file near the top a “variables” section can be found, which by default will look similar to:

```
<variables
    utilsdir="%INST/utils"
    emailprefixdir="%INST/email"
    logsdire="/var/adm"
/>
```

To generate log files in another directory simply change the “logsdire” setting. For example if you wish to generate them in “/tmp” then use:

```
<variables
    utilsdir="%INST/utils"
    emailprefixdir="%INST/email"
    logsdire="/tmp"
/>
```

3.2 Configure Default Logging Options

The configuration file by default has the following section defined for logging options:

```
<logging level="5"
    dir="/var/adm"
    file="skulker-%D.log"/>
<!--
    If file == NULL then use STDOUT for logging details.
-->
```

The logging level of 5 is the most important one here. A level of “5” means that by default Skulker will not actually do anything - just attempt to simulate [as best it can] the changes that would result if

the rules were actually run. This is the safest option but might be somewhat confusing. Many environments change this to “3” - a good level of logging whilst still actually making changes.

The default name of the logs generated is probably best left as is - a separate log file for each day is most sensible if the logs are to be digested in any way.

3.3 Symbolic Link Handling

Symbolic links are one of the most useful tools in UNIX - and one of the most dangerous. For example consider the following link in a users home directory:

```
/home/usera/bin -> /usr/bin
```

Now imagine if the administrator generated a rule that looked for setuid binaries under /home and deleted them automatically via a Skulker rule. Such a link would result might result in the search including the contents of "/usr/bin" - obviously deleting all setuid programs here would cause a big problem!

For this reason Skulker allows the administrator some control on how Symbolic links are dealt with when scanning directories. The options available are:

follow_skip	Links are not followed - only the link name is included in the list of items to check. This is the default setting.
follow_fast	Follow all symbolic links - even if the destination directory has already been processed. Might cause Skulker to abort if the same file is found multiple times [and hence it not recommended].
follow	Follow symbolic links - though all results are only ever processed once. This requires more resources when dealing with large directory structures.

3.4 Compression Types Configuration

Skulker integrates with compressed files in a consistent manner across all support rule actions. The most obvious interface to compression is via the "compress" rule type - though compressed files are dealt with implicitly in many other circumstances.

For example it is possible to "tail" compressed files - both as source and destination files. Rotation also takes account of compressed file extensions when performing rotations as well.

The available compression types that Skulker can handle is determined by the "compression" section of the main configuration file. A typical example might be:

```
<compression>
  <type name="bzip2"
    extension=".bz2"
    read="/usr/local/bin/bunzip2 -c"
    write="/usr/local/bin/bzip2 -f"/>
  <type name="compress"
    extension=".Z"
    read="/usr/bin/zcat"
    write="/usr/bin/compress -f"/>
  <type name="gzip"
    extension=".gz"
    read="/usr/contrib/bin/gunzip -c"
    write="/usr/contrib/bin/gzip -f"/>
</compression>
```

For a given machine please remove entries that are not considered relevant [or available]. Also ensure that the full paths for the executables are correct for the machine in question.

Notice that the “read” and “write” commands specified must be the commands which read from standard input and write to standard output respectively. If a given compression type does not support this type of functionality it can not be used in Skulker.

There are no limits on the number of different compression types that can be configured - though at least one should be used. Also ensure that the default type configured matches one of the available compression types.

3.5 Protected Directories Configuration

Skulker manipulates and deletes files in order to save disk space. It should be obvious that such a tool, which must typically run as “root”, could cause major problems if an incorrectly configured rule deleted the wrong files - such as device files, binary programs, libraries or even the kernel image!

In an effort to mitigate against such possibilities the concept of “protected directories” has been included in versions from 0.3.1 onwards. To make use of this facility one or more entries are configured as part of the “protected_dirs” section of the configuration file. A typically section [this one taken from a HP-UX configuration file] would look similar to the following:

```
<protected_dirs>
  <dir   path="/usr"
        recursive="yes"
        override="no"/>
  <dir   path="/sbin"
        recursive="yes"
        override="no"/>
  <dir   path="/lib"
        recursive="yes"
        override="no"/>
  <dir   path="/stand"
        recursive="yes"
        override="no"/>
  <dir   path="/boot"
        recursive="yes"
        override="no"/>
  <dir   path="/dev"
        recursive="yes"
        override="yes"/>
  <dir   path="/etc"
        recursive="yes"
        override="yes"/>
</protected_dirs>
```

Further directories can be added if necessary. Apart from the path that is protected two attributes should be present:

- **recursive** - Can be set to “yes” or “no”. If “yes” then all sub-directories specified are also assumed to be protected as well. For example in the above example “/usr” has it’s “recursive” attribute set to “yes”, meaning that “/usr”, “/usr/lib”, “/usr/bin”, “/usr/local/X11/bin” are all examples of directories that are protected. When set to “no” just the entries in the directory itself are protected.
- **override** - This is not currently made use of. However in later releases it will determine whether the settings for a directory can be over-ruled explicitly for rules. Typically this should be set to “no” unless there are good reasons for changing this.

3.6 Rule Default Configuration Options

The “defaults” section provides many settings which help ensure that Skulker works in a sensible manner when processing command line arguments or even rules themselves. The next section covers defaults for the rules, first one covers command line handling.

The top of the configuration file typically contains settings relevant to command line argument handling:

```
<defaults>
  <rules      directory="%INST/rules"
             file="MAIN.xml"/>
  <status    file="%INST/var/status.info"/>
  <tests     directory="%INST/tests"/>
  <functions  directory="%INST/functions"
             list="archive delete compress tail rotate saswork
                 spdswork command deletedir mailfile organise"/>
```

Each of the elements present are:

rules	The “rules” settings determine the directory where all rules files exist, and the default rules file to run. Typically these do not need to be changed since they refer to the installation directory and “MAIN.xml” which loads in rule configuration files dependent on the current Operating System.
status	The “file” attribute determines the complete path to the file which contains the statistics of running Skulker on the current machine. The default location is a sub-directory of “var” under the installation directory. The amount of storage required for the statistics is quite small and is determined only by the number of rules configured rather than the number of times Skulker is run or number of files processed. This can be changed but at present a typical amount of storage used is only 20 Kb.
tests	This element has a single attribute “tests”, which defines where the self-test routines can be found. The default should not be changed.
functions	The “directory” attribute determines where Skulker looks for rules code to auto-load when it is started. Again this does not need to be changed. The “list” attribute determines the module names to load. This might need to be changed if the user adds additional custom modules and wishes to run rules against them.

3.7 Per Rule Action Defaults

The remaining entries in the “defaults” section refer to defaults for each of the rules. The standard installation tries to ensure these are “sensible” for almost all environments, though they can be changed if required.

For information on the available defaults and what and how these can be changed please view the associated “Skulker Configuration Guide”.

4 First Time Actions

4.1 Validate Base Rule Set

Once the basic log setting has been changed, if necessary the next step is to validate the syntax for the base rules for the current architecture. Typically this will first require setting the “SKULKERV2_CFG” variable:

```
# export SKULKERV2_CFG=/opt/skulker2/cfg/cfg.xml
```

Again if the software is installed in a customer specific location use:

```
# export SKULKERV2_CFG=/opt/customer/skulker2/cfg/cfg.xml
```

```
# /opt/skulker2/bin/skulkerv2 --action=parse --logfile=-
```

Again for a specific customer the path might be:

```
# /opt/customer/skulker2/bin/skulkerv2 --action=parse --logfile=-
```

This command will scroll through lots of information, but the last two lines generated should be:

```
2006/04/11 12:54:34 LOG      Rule parsing stage complete.
2006/04/11 12:54:34 LOG      Sucessfully opened Status file.
```

4.2 Initial Rule Set Statistics Generation

If the parsing works OK, then the next step is to generate the default “statistics” entries for all the rules. This can be done using the “generate” action - for example:

```
/opt/mbnatools/skulker2/bin/skulkerv2 --action=generate --logfile=-
```

One possible problem is that the directory for the statistics information does not exist. In this instance run the following command:

```
# mkdir /opt/skulker2/var
```

[Again the “customer” directory may need to be added to the above path]

Once the “skulkerv2” command completes it should generate output ending with something similar to:

```
2006/04/11 14:04:58 LOG      Generated rule entry for hpux-wtmplogs:1000
2006/04/11 14:04:58 LOG      Generated rule entry for hpux-wtmplogs:1010
2006/04/11 14:04:58 LOG      Generated rule entry for hpux-wtmplogs:1020
2006/04/11 14:04:58 LOG      Generated rule entry for hpux-wtmplogs:1030
2006/04/11 14:04:58 LOG      Generated rule entry for hpux-wtmplogs:1040
2006/04/11 14:04:58 LOG      Generated rule entry for hpux-wtmplogs:1050
2006/04/11 14:04:58 LOG      Generated rule entry for skulker2:1000
2006/04/11 14:04:58 LOG      Generated rule entry for skulker2:1010
2006/04/11 14:04:58 LOG      Status records added: 71
2006/04/11 14:04:58 LOG      Status records validated/added successfully.
```

The actual rules shown and count will differ depending on architecture.

4.3 Initial Dry Run

Prior to actually configuring the software to run automatically it is often useful to perform a “dry-run” to ensure the files actions are indeed inline with expectations. To do this simply run the following command (assuming the SKULKERV2_CFG environment variable is still defined):

```
# ./skulkerv2 --action=skulk --loglevel=5 --ignore
```

This will append to the standard log file rather than show much output to the screen. Please note that even in this case some screen output will be given if attached to a terminal. Check the log file, which in the standard case will be named:

```
/var/adm/skulker-YYYYMMDD.log
```

The amount of text generate will be significant - but because the “loglevel” was set to 5 *no actual changes to data took place*. The results should give you confidence to deploy the software automatically.

4.4 Automated Scheduling

Given that each rule has a minimum defined periods which they can be run, the user is responsible for defining a suitable frequency for scheduling the invocation of Skulker.

The most common manner of invocation is simply once over-night. However sometimes certain rules might need to be run on a more regular basis, and so if necessary the administrator can specify a non-default rules file on the command line.

For example consider the following crontab entry (for “root”) which runs the default ruleset every night at 02:25:

```
25 2 * * * /opt/mbnatools/skulker2/bin/skulkerv2 \  
--cfg /opt/mbnatools/skulker2/cfg/cfg.xml --loglevel=3 \  
--actions=generate,skulk,summary
```

Because no environment variables are set the “--cfg” option above is used to define the configuration for the tool. The “--actions” option is used to ensure a summary report is also written to the log following the main “skulk” action.

4.5 Rule Set Customisation

Once the base installation has been validated it is typical to add rules specific to each host [for managing logs related to the currently running application]. The easiest way to do this is to create a file in the rules directory called:

```
hostonly-hostname.xml
```

Once this has been done the rules will be run using the default configuration file. For information on the rules configuration see the document “Skulker V2 : Rule Configuration Guide”.

5 HTML Report Generation Support

5.1 Introduction

From version 0.4.1 of Skulker 2 it is possible to take statistical information stored in the history database [if configured] and generate HTML reports from them. Depending on your environment it is likely that the relevant components are not installed.

When attempting to generate a html report without the necessary software the following error will be recorded in the logfile:

```
2007/12/19 16:23:16 ERROR Unable to create histgraphs: No graphing software
[GD::Graph] available.
```

The lack of this software does not impinge any other functionality of Skulker 2 - and thus completing the steps in the rest of this section is only necessary if you wish to generate HTML graphs.

5.2 Installation of Additional Software

To enable html generation the following CPAN Perl modules must be installed on each host:

GD::Graph
GD::Text::Align

These modules can be compiled from source, via CPAN or via packages for the host environment. Once installed html reporting will be available.